1-1-2015

Frontiers In Operations Research For Overcoming Barriers To Vehicle Electrification

Mahyar Movahed Nejad
Wayne State University,

FRONTIERS IN OPERATIONS RESEARCH FOR OVERCOMING BARRIERS TO VEHICLE ELECTRIFICATION

by

MAHYAR MOVAHED NEJAD

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2015

MAJOR: INDUSTRIAL ENGINEERING

Approved by:

Advisor Date

DEDICATION

To my parents Maryam and Ali, and my brother Mehran.

ACKNOWLEDGMENTS

I owe a debt of gratitude to the many remarkable people I have had the fortune of knowing during my PhD studies. First, I would like to express my deepest appreciation to my advisor Dr. Chinnam for his carefully-tuned mentorship, technical advise, encouragement, and support. I am thankful for everything he has taught me without which this dissertation would have not been completed.

I am also grateful to my dissertation committee members Dr. Phillips, Dr. Murat, and Dr. Chelst for their constructive suggestions and helpful comments on my research. I am greatly thankful to Dr. Monplaisir, Dr. Grosu, and Dr. Fotouhi for their support during my PhD studies.

I want to thank the faculty and the staff in the department for all of their help and assistants. I am also grateful to the incredible group of friends I have made in Michigan who made these years much more enjoyable.

I would like to express my deepest gratitude to my parents Maryam and Ali, and my brother Mehran for their endless love and support. I must also express my heartfelt gratitude to my brilliant and loving Lena. I am grateful to have her in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Electric vehicles (EVs) hold many promises including diversification of the transportation energy feedstock and reduction of greenhouse gas and other emissions. However, achieving large-scale adoption of EVs presents a number of challenges resulting from a current lack of supporting infrastructure and difficulties in overcoming technological barriers. This dissertation addresses some of these challenges by contributing to the advancement of theories in the areas of network optimization and mechanism design.

To increase the electric driving range of plug-in hybrid electric vehicles (PHEVs), we propose a power-train energy management control system that exploits energy efficiency differences of the electric machine and the internal combustion engine during route planning. We introduce the Energy-Efficient Routing problem (EERP) for PHEVs, and formulate this problem as a new class of the shortest path problem. The objective of the EERP is to not only find a path to any given destination, but also to identify the predominant operating mode for each segment of the path in order to minimize the fuel consumption. We prove that the EERP is NP-complete. We then propose two exact algorithms that find optimal solutions by exploiting the transitive structure inherent in the network. To tackle the intractability of the problem, we proposed a Fully Polynomial Time Approximation Scheme (FPTAS).

From a theoretic perspective, the proposed two-phase approaches improve the state-of-the-art to optimally solving shortest path problems on general constrained multi-graph networks. These novel approaches are scalable and offer broad potential in many network optimization problems. In the context of vehicle routing, this is the first study to take into account energy efficiency difference of different operating modes of PHEVs during route planning, which is a high level power-train energy management procedure.

Another challenge for EV adoption is the inefficiency of current charging systems. In addition, high electricity consumption rates of EVs during charging make the load management of micro grids a challenge. We proposed an offline optimal mechanism for scheduling and pricing of electric vehicle charging considering incentives of both EV owners and utility companies. In the offline setting, information about future supply and demand is known to the scheduler. By considering uncertainty about future demand, we then designed a family of online mechanisms for real-time scheduling of EV charging. A fundamental problem with significant economic implications is how to price the charging units at different times under dynamic demand. We propose novel bidding based mechanisms for online scheduling and pricing of electric vehicle charging. The proposed preemption-aware charging mechanisms consider incentives of both EV drivers and grid operators. We also prove incentive-compatibility of the mechanisms, that is, truthful reporting is a dominant strategy for self-interested EV drivers. The proposed mechanisms demonstrate the benefits of electric grid load management, revenue maximization, and quick response, key attributes when providing online charging services.

Another challenge for vehicle routing problems, including the EV routing, is how to efficiently incorporate information from Intelligent Transportation Systems (ITS) in route planning. The quickly expanding ITS coverage around the world can be a key enabler for efficient vehicle route planning and for reducing the effects of traffic congestion on travel times. ITS provides valuable information for a time-dependent road network, such as time-varying travel times for traversing road segments at high resolu- tion (e.g., one minute). Routing algorithms must exploit these traffic information feeds efficiently, both to plan the route in advance and to update it en route. In general, an efficient routing algorithm should strike a balance among preprocessing time, query time, optimality gap, and storage/processor memory requirements. In addition, the scalability of the routing algorithm for handling large-scale road networks while maintaining reasonable response times is an important property. Depending on the form of implementation of the routing application, however, some of the aforementioned features may be prioritized over others. We focus on large-scale deterministic time-dependent transportation networks. The need for fast responses to ITS information puts the speed-up techniques for shortest path problems (SPP) on time-dependent networks at the heart of computa- tional needs for routing. In addition, a vast majority of vehicle routing navigation systems, whether built-in or portable, lack the ability to rely on online servers and must compute the route in a stand-alone mode with limited hardware processing/memory capacity. This last aspect is the primary focus of this study to design computationally efficient yet effective hierarchical search strategies and algorithms to solve the time-dependent shortest path problem (TDSP).

1.1 Organization

The rest of this dissertation is organized as follows. In Chapter 2, we present optimal routing for plug-in hybrid electric vehicles. In Chapter 2.6, we introduce novel online scheduling and pricing mechanisms for lectric vehicle charging. In Chapter 3.7, we present hierarchical time-dependent shortest path algorithms for vehicle routing under intelligent transportation systems. In Chapter 5, we summarize our results and present possible directions for future research.

CHAPTER 2: OPTIMAL ROUTING FOR PLUG-IN HYBRID ELECTRIC VEHICLES

2.1 Introduction

Electrified vehicles promise to enable diversification of the transportation energy feedstock that can reduce the dependence on petroleum for transport, lessen greenhouse gas and other emissions, and provide more sustainable transportation. The governments of the U.S., European Union, China, Japan, Korea and others are aggressively promoting vehicle electrification objectives, and the major automobile companies of the world are being challenged by environmentally conscious consumers and governments to produce affordable electrified vehicles. Several companies have accepted the challenge, and more models of electrified vehicles (hybrid vehicles, plug-in hybrid vehicles, pure battery electric vehicles) are being introduced every year around the world [57].

Hybrid vehicles have become increasingly popular in the automotive marketplace in the past decade. The most common type is the electric hybrid, which consists of an internal combustion engine (ICE), a battery, and at least one electric machine (EM). Hybrids are built in several configurations including series and parallel. In a series PHEV, only the EM is connected to the wheel and can operate the vehicle, while its ICE produces electricity as a generator. In a parallel PHEV, both EM and ICE are connected to the wheel, thus, the vehicle can be operated on EM, ICE, or both. In this study, we consider parallel plug-in hybrid electric vehicles, and refer them by the term PHEVs.

While pure battery electric vehicles (BEVs) are desired for their significant reduction in emissions, their deployment presents a host of challenges resulting from a current lack of supporting infrastructure: charging stations are relatively sparse, charging takes considerable time, and currently the driving range of BEVs is significantly limited. PHEVs partially address these concerns by allowing the vehicle to be operated in an all electric mode, internal combustion mode, or a combination, mitigating the range anxiety associated with BEVs. When the batteries are depleted, the ICEs of PHEVs work as a backup, providing a driving range comparable to conventional internal combustion vehicles. Unlike standard hybrid vehicles, PHEVs also offer the ability to be recharged from an external electrical outlet. However, the unique capability of hybrids and PHEVs to operate in multiple modes (electric, gasoline, or hybrid mode) brings about new challenges to their routing problems. This is unlike routing algorithms for pure BEVs that employ just one operating mode (i.e., the electric mode).

Literature offers a number of routing algorithms for BEVs ([116, 28, 129]). However, unlike BEVs, the routing algorithms for PHEVs should also account for the significant energy efficiency differences of different operating modes and recommend the predominant mode of operation for each road segment during route planning. The energy efficiency differences are also a function of the vehicle (e.g., payload) and road segment features (e.g., speed limits, terrain geometry). Given that ICEs tend to be most efficient when operating at steady highway speeds of 45-65 MPH, the electric mode is relatively efficient on city roads with lower speed limits [104]. Given that the all-electric range is limited, the routing algorithm should explicitly exploit these energy efficiency differences for the different road segments in planning the route.

The strategy to control the energy among these multiple energy sources of a hybrid vehicle is termed "Powertrain Energy Management". An overview of this area is provided by [122]. Currently, powertrain energy management control algorithms for PHEVs are mostly "static" and try to utilize the charge within the battery as soon as possible, without explicit consideration for real opportunities that might be present within the route to best utilize the battery charge. This is also attributable to the fact that there exist no route planning algorithms for PHEVs, and automotive original equipment manufacturers (OEMs) are yet to develop "dynamic" energy management control algorithms that account for the complete route plan in controlling the vehicle powertrain operating modes.

Given that powertrain energy management systems take control actions in the order of few milliseconds and trip travel times can span minutes to several hours, it is not practical for routing algorithms to explicitly optimize the actions to be taken by the energy management system along the route. Instead, routing algorithms for PHEVs should consider segmenting the roads into segments with relatively uniform energy efficiency conditions and identify the predominant operating mode (electric mode or gasoline mode) for each road segment and delegate the actual energy management to lower level control algorithms. In this study, we propose routing algorithms for PHEVs that find an optimal path along with the predominant operating mode for road segments of the path.

Our proposed routing algorithms assume (without loss of generality) that the arcs of the road network have been pre-segmented into short-distance sections with uniform energy efficiency conditions for the different operating modes (electric mode or gasoline mode). The task of the routing algorithms is to identify the optimal route as well as the predominant operating mode for each segment. While PHEVs can occasionally operate in a hybrid mode where both power sources (EM as well as the ICE) can be employed simultaneously to drive the vehicle, the power split policies are managed by powertrain energy management systems, which form a lower level control than routing. Therefore, the routing algorithms for any given road segment to be traversed can only consider a single operating mode (all electric or all gasoline). In the rest of this study, we refer to the two vehicle modes as *electric mode*, when operated by a battery driven EM, and *gasoline mode*, when operated by the ICE (irrespective of the type of petroleum derived fuel employed by the engine).

The cost to drive the vehicle in electric mode, at least in the U.S., is currently several times cheaper than using the gasoline mode ([115, 124]). Therefore, the objective of routing algorithms for PHEVs is to minimize the cost of using the gasoline mode through the planned route. In doing so, the routing algorithm should explicitly consider the battery state of charge (SOC) at the beginning of the trip (once depleted, the rest of the trip should only employ the gasoline mode).

In this study, we propose energy-efficient routing algorithms for PHEVs considering battery SOC constraints. The decision is to not only select the route (i.e., the road segments), but also the vehicle's operating mode for each road segment (i.e., gasoline or electric).

2.1.1 Our Contribution

We address the problem of Energy-Efficient Routing (EERP) for PHEVs by designing exact and approximation algorithms for solving it. The primary contributions of this study include introducing a new class of the shortest path problems for the energy-efficient routing of PHEVs, and designing two exact algorithms and a fully polynomial time approximation scheme (FPTAS) to solve the EERP. We propose to represent the fuel consumption network as a multigraph, so that the alternative fuel consumptions (gasoline or electricity) are considered. Our proposed algorithms to solve the EERP consider general networks, which makes them suitable for road networks that are not acyclic, ordered, etc. We first model the problem as an Integer Program, and present the hardness proof of the EERP. In the absence of solution methods for solving the EERP, we design two exact pseudopolynomial algorithms Exact-EER-I and Exact-EER-II that find the exact solution with minimum gasoline consumption, with time complexity of $O(C(|A| + |V|^2))$ and $O(C(|A| + |V| \log(|V|)))$, respectively, where C is the battery SOC at the beginning of the trip. We then design an approximation scheme for the EERP, called FPTAS-EER. In addition, we prove that the proposed FPTAS-EER is a fully polynomial time approximation scheme. We analyze the properties of Exact-EER-I, Exact-EER-II, and FPTAS-EER, and conduct extensive experiments. The results show significant energy savings for PHEVs over simplistic routing algorithms and current practice.

2.1.2 Related Work

The EERP can be considered as a class of the shortest path problem (SPP). The SPP is at the heart of general network problems. There is extensive literature on different classes of the SPP and their applications in vehicle routing problems ([9, 2, 6]).

We focus on studies on SPP that are closely related to our work, in particular, those on the resource constrained shortest path problem (RCSPP). The resource constrained shortest path problem is defined as follows:

Definition 1 (Resource Constrained Shortest Path Problem)**.** *Given a graph in which* *each arc is characterized by cost and a set of resource consumptions, find the shortest path from O to D such that the total consumption of each resource along the path is less than or equal to a specified value.*

Unlike the SPP, which is solvable in polynomial time, the RCSPP is NP-hard. For a recent survey on exact solution approaches for the RCSPP we refer the reader to [23]. The solution methods for the RCSPP can be briefly classified into path ranking methods (e.g., [118]), Lagrangian relaxation methods (e.g., [84]), and node labelling methods (e.g., [109]). RCSPP has been studied extensively, however, here we only focus on a special case of RCSPP that considers one resource consumption. This problem is refereed to as the restricted shortest path problem (RSPP) or weighted constrained shortest path problem. The RSPP is closer to our introduced problem since it has one resource. Note that we consider the battery SOC as a resource, and we assume that there is plenty of gasoline to complete the trip in just gasoline mode.

The RSPP is defined as follows:

Definition 2 (Restricted Shortest Path Problem)**.** *Given a graph in which each arc is characterized by length and transition time, find the shortest path from O to D such that the total delay of the path is less than or equal to a specified value.*

The RSPP is suggested to be NP-hard [38], even if the graph is acyclic, and all length and transition times are positive [27]. However, the RSPP becomes NP-Complete if the graph does not contain negative length cycles, and can be solved in pseudopolynomial time [109]. For a survey on exact and approximate solution approaches for the RSPP and the RCSPP we refer the reader to [39]. [134] was the first to introduce a polynomial-time approximation scheme (PTAS) for the RSPP on acyclic graphs. [53] improved the results of [134] by deriving an FPTAS with time complexity of $O((|A||V|/\epsilon)\log\log(U/L))$ on acyclic graphs, where U and L are upper and lower bounds on the optimal solution. [103] proposed a PTAS for the RSPP, which uses a Dijkstra-based algorithm with time complexity of $O((|A||V|/\epsilon + (|V|^2/\epsilon)\log(|V|^2/\epsilon))\log\log(U/L))$. [45] proposed a PTAS for the RSPP from a source to all destinations with the time complexity of $O(|V|/\epsilon(|A| + |V|\log(|V|)))$.

However, their approach relaxes the time delay instead of the length, that is, they allow for an error in the delay bound. In general, rounding for the delay is easier since the bound is known. [70] reduced the complexity of the results by [53] to $O(|A||V|(1/\epsilon + \log\log(U/L)))$ by proposing an FPTAS. [29] further improved this time complexity to $O(|A||V|/\epsilon)$. However, their FPTAS method works only on acyclic graphs, where nodes have a topological order. [14] proposed two approximation algorithms for the RSPP based on rounding delays instead of lengths. Their approach has the same worst time complexity as that proposed by [45]. In simulation, however, their algorithms run one order of magnitude faster on the average case. Finding several paths with different length-delay trade-offs was investigated by [24], where they approximated the Pareto curves. [8] proposed an approximation approach for the RSPP with close to linear running time. However, his approach only works for undirected graphs, and it is randomized. In addition, the approach achieves a $(1+\epsilon)$ approximation in both parameters (i.e., delay and length). For various applications of the RSPP, the reader is referred to [69], [112], [71], and [123].

In the RSPP, the goal is to solve the shortest path by selecting the arcs based on all of their attributes (length and delay). However, in our introduced problem, the EERP, the decisions are not only about choosing arcs but also about choosing the mode of operation (gasoline or electric mode) for each arc. In addition, our proposed exact and approximation algorithms work on general graphs, and do not require any assumptions regarding the network, such as, acyclicity and predetermined order of the nodes in the network.

Several researchers investigated the powertrain management for hybrids and PHEVs from an energy management control perspective (e.g., [49, 86]). However, none of them have addressed the energy-efficient routing of PHEVs during route planning, which is a higher level energy management process for PHEVs. In the absence of energy-efficient routing algorithms for PHEVs, we design exact and approximation routing algorithms that consider general graphs.

2.1.3 Organization

The rest of the study is organized as follows. In Section 2.2, we introduce the energy-efficient routing problem for PHEVs. In Section 2.3, we describe our proposed exact algorithms for solving the EERP. In Section 2.4, we propose an FPTAS for solving the EERP, and characterize its properties. In Section 2.5, we evaluate the proposed algorithms by extensive experiments on the Southeast Michigan road network. In Section 2.6, we conclude the study and present possible directions for future research.

2.2 Energy-Efficient Routing Problem for PHEV

PHEVs can run on both electric and gasoline modes as long as there is adequate charge in the battery and the gasoline tank is not empty, respectively. This feature necessitates routing algorithms to incorporate decision on vehicle's operating mode in path planning in order to enhance fuel economy and reduce emissions. The decision in energy-efficient routing problems for PHEVs involves not only choosing road segments to traverse, but also choosing vehicle's predominant operation mode for each segment. As a result, energy-efficient routing algorithms considering battery SOC constraints should be designed to find a path to any given destination while minimizing the fuel consumption of PHEVs.

We model the *road network* as a directed graph $G_R = (V, A_R)$, where V is the set of nodes and A_R is the set of arcs. Arc $(i, j) \in A_R$ represents the road segment connecting nodes i and j, where $i, j \in V$. Since each arc in the network can be traversed by the vehicle using either the gasoline-based engine or the electrical motor, we associate with it the gasoline consumption (in gallons) or battery charge consumption (in Watt hours). Taking into account these two parameters, we model the *fuel consumption network* as a directed multigraph $G = (V, A)$ of multiplicity two. The multiplicity represents the maximum number of edges from one node to another. The graph G is induced by the road segment network graph G_R in the sense that for each arc (i, j) in G_R, there are two arcs $(i, j)^e$ and $(i, j)^g$ in G that correspond to the choice of traversing the arc (i, j) in G_R using electrical power and gasoline power, respectively. The set of all arcs in G of these two

Table 2.1: Notation

V	Set of nodes $\{1, \ldots, N\}$
A^e	Set of all arcs $(i,j)^e \in A$
A^g	Set of all arcs $(i,j)^g \in A$
w_{ij}^e	Battery charge consumption when traversing arc (i,j)
w_{ij}^g	Gasoline consumption when traversing arc (i,j)
O	Origin node
D	Destination node
C	Available battery SOC at O
Γ	Total gasoline consumption
E	Total battery charge consumption

types is denoted by A. We associate with each arc $(i,j)^e \in A$ a weight w_{ij}^e representing the battery charge consumption when traversing arc $(i,j) \in A_R$. Similarly, we associate with each arc $(i,j)^g \in A$ a weight w_{ij}^g representing the gasoline consumption when traversing arc $(i,j) \in A_R$. We denote by A^e the set of all arcs $(i,j)^e \in A$ and by A^g the set of all arcs $(i,j)^g \in A$. We define two decision variables x_{ij} and y_{ij} as follows:

$$x_{ij} = \begin{cases} 1 & \text{if the vehicle passes through arc } (i,j) \text{ in gasoline mode,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

$$y_{ij} = \begin{cases} 1 & \text{if the vehicle passes through arc } (i,j) \text{ in electric mode,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

The vehicle has an available battery SOC C and needs to travel from an origin node $O \in V$ to a destination node $D \in V$. The total gasoline consumption Γ on a path P in G_R from O to D is given by $\Gamma = \sum_{(i,j) \in P} w_{ij}^g x_{ij}$. Similarly, the total battery charge consumption E on a path P in G_R from O to D is given by $E = \sum_{(i,j) \in P} w_{ij}^e y_{ij}$. Table 2.1 summarizes the notation used throughout the study.

We define the problem of Energy-Efficient Routing (EERP) for PHEVs as follows.

Definition 3 (Energy-Efficient Routing Problem (EERP))**.** *Given a fuel consumption net-*

work $G = (V, A)$ and a PHEV with available battery SOC C, find a path from an origin node $O \in V$ to a destination node $D \in V$ such that the total gasoline consumption Γ is minimized.

We formulate the EERP as an Integer Program as follows:

$$\text{Minimize} \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}^{g} x_{ij} \tag{2.3}$$

Subject to:

$$\sum_{i=1}^{N}(x_{Oi} + y_{Oi}) = 1 \tag{2.4}$$

$$\sum_{i=1}^{N}(x_{iD} + y_{iD}) = 1 \tag{2.5}$$

$$\sum_{i=1}^{N}(x_{ij} + y_{ij}) - \sum_{k=1}^{N}(x_{jk} + y_{jk}) = 0, \forall j; j \notin \{O, D\} \tag{2.6}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}^{e} y_{ij} \leq C \tag{2.7}$$

$$x_{ij} + y_{ij} \leq 1, \forall i, j \tag{2.8}$$

$$x_{ij}, y_{ij} = \{0, 1\}, \forall i, j. \tag{2.9}$$

The objective function represents the total gasoline consumption for a trip from O to D. Without loss of generality, we assume that the cost of electricity is lower than that of gasoline, hence, the vehicle should run in electric mode as much as possible to minimize the fuel cost. Constraint (2.4) ensures that there is an outgoing arc in the final path from O. Constraint (2.5) guarantees that there is an incoming arc to D in the final path. Connectivity of the path from O to D is ensured by constraints (2.6). Constraint (2.7) guarantees that the battery charge consumption does not exceed the available battery SOC of the vehicle. Constraints (2.8) ensure that for each pair of nodes at most one vehicle operating mode is selected. Constraints (2.9) represent the integrality requirements for the decision variables. The solution to the EERP is represented as a tuple of matrices $(\mathbf{X}, \mathbf{Y})$, where $\mathbf{X} = [x_{ij}]_{i,j \in V}$ and $\mathbf{Y} = [y_{ij}]_{i,j \in V}$.

We now define the EERP for a PHEV along a given path called EERP-PATH, as follows:

Definition 4 (EERP-PATH). *Find operating modes for a PHEV with available battery SOC C that must travel along a fixed path P such that the total gasoline consumption Γ is minimized.*

This problem considers a fixed route P consisting of a sequence of nodes $1, \ldots, N$. For each arc, there are two choices for a PHEV to operate: electric mode or gasoline mode. We formulate the problem as an Integer Program as follows:

$$\text{Maximize } \sum_{i=1}^{N-1} -w^g_{i,i+1} x_{i,i+1} \tag{2.10}$$

Subject to:

$$\sum_{i=1}^{N-1} w^e_{i,i+1} y_{i,i+1} \leq C \tag{2.11}$$

$$x_{i,i+1} + y_{i,i+1} = 1, \forall i = 1, \ldots, N-1 \tag{2.12}$$

$$x_{i,i+1}, y_{i,i+1} = \{0, 1\}, \forall i = 1, \ldots, N-1. \tag{2.13}$$

The objective function represents the total saving of gasoline consumption for the path $P = (1, \ldots, N)$. Constraint (2.11) guarantees that the battery charge consumption does not exceed the available battery SOC of the vehicle. Constraints (2.12) ensure that for each arc on the path, only one vehicle operating mode is selected. Constraints (2.13) represent the integrality requirements for the decision variables. The objective function is considered to be maximized since in the hardness proof of the problem, presented in Theorem 1, we rely on a reduction to the multiple-choice knapsack problem, which is a maximization problem.

Theorem 1. *EERP-PATH is NP-complete.*

Proof. Proof. To show that the EERP-PATH is NP-complete we consider its decision version, called EERP-PATH-D. EERP-PATH-D is defined as follows: Given a fuel consumption network $G = (V, A)$ and a path P, a PHEV with available battery SOC C, and a given bound Q on gasoline consumption, is there a solution such that the total gasoline consumption Γ is at most Q?

The first step is to prove that the EERP-PATH-D is in NP by showing that given a certificate $(\mathbf{X}, \mathbf{Y})$, it can be decided in polynomial time that $(\mathbf{X}, \mathbf{Y})$ is a solution to the problem or not. Given a path P, battery SOC C, and gasoline consumption Q, it is easy to check if $(\mathbf{X}, \mathbf{Y})$ is a solution. This involves computing two sums and comparing the results against Q and C, that is, checking $\sum_{(i,j)\in P} w^g_{ij} x_{ij} \leq Q$ and $\sum_{(i,j)\in P} w^e_{ij} y_{ij} \leq C$. This requires an amount of time linear in the number of arcs in G. Thus, the EERP-PATH-D is in NP.

The second step is to find a polynomial-time reduction from the decision version of the 0-1 multiple-choice knapsack problem (denoted here by MCKP-D), which is known to be NP-complete [64]. The MCKP-D problem is defined as follows: Given k mutually disjoint sets $\{S_1, \ldots, S_k\}$ of items to pack in a knapsack of capacity C, where each item $j \in S_i$ has a value v_{ij} and a weight w_{ij}, is there a subset T with k items where each item $s_{ij} \in T$ is from one set S_i with total weight at most C (i.e., $\sum_{i=1}^{k} \sum_{j\in S_i} \sum_{s_{ij}\in T} w_{ij} \leq C$), such that the corresponding profit is at least R (i.e., $\sum_{i=1}^{k} \sum_{j\in S_i} \sum_{s_{ij}\in T} v_{ij} \geq R$)?

We consider an instance of the EERP-PATH-D on a given path $P = (1, \ldots, N)$, and a battery SOC of C. We then use a one-to-one mapping between the items in $\{S_1, \ldots, S_k\}$ and the arcs in P, where P consists of $N-1$ arcs. For each arc $(i, i+1) \in P$ in the EERP-PATH-D, there is a corresponding set of items S_i in MCKP-D. That means, there are $k = N-1$ mutually disjoint sets of arcs. We consider a specific version of MCKP-D, where each set has two items. Each arc $(i, i+1) \in P$ has two choices $(w^e_{i,i+1}, 0)$ and $(0, -w^g_{i,i+1})$, corresponding to two items in its set in MCKP-D. The first choice, $(w^e_{i,i+1}, 0)$, represents traversing the arc by electric mode, which corresponds to an item of size $w_{i1} = w^e_{i,i+1}$ and value $v_{i1} = 0$. The second choice, $(0, -w^g_{i,i+1})$, represents traversing the arc by gasoline mode, which corresponds to an item of size $w_{i2} = 0$ and value $v_{i2} = -w^g_{i,i+1}$. MCKP-D selects one of these choices for each arc to maximize the sum of values satisfying the capacity constraint, where the total battery charge consumption of arcs in $\{S_1, \ldots, S_{N-1}\}$ is $\sum_{i=1}^{N-1} \sum_{j\in S_i} \sum_{s_{ij}\in T} w_{ij}$, and the total gasoline consumption is $\sum_{i=1}^{N-1} \sum_{j\in S_i} \sum_{s_{ij}\in T} v_{ij}$.

The Yes/No answer to the EERP-PATH-D instance corresponds to the same answer to the MCKP-D instance. If there is a Yes answer to the EERP-PATH-D instance, it means we can find such a set T of arcs in path P that satisfies $\sum_{i=1}^{N-1}\sum_{j\in S_i}\sum_{s_{ij}\in T} w_{ij} \leq C$ and $\sum_{i=1}^{N-1}\sum_{j\in S_i}\sum_{s_{ij}\in T} v_{ij} \geq -Q$. Then, this subset T is also a solution to the MCKP-D instance. As a result, the answer to the MCKP-D instance must also be Yes. Conversely, if there is a No answer to the EERP-PATH-D instance, it means there is no set T that satisfies $\sum_{i=1}^{N-1}\sum_{j\in S_i}\sum_{s_{ij}\in T} w_{ij} \leq C$ and $\sum_{i=1}^{N-1}\sum_{j\in S_i}\sum_{s_{ij}\in T} v_{ij} \geq -Q$. As a result, the answer to the MCKP-D instance must also be No.

This reduction from the MCKP-D to the EERP-PATH-D can be done in polynomial time. Therefore, the EERP-PATH-D is NP-complete. Since the EERP-PATH-D is the decision version of the EERP-PATH it follows that the EERP-PATH is also NP-complete. □

In the following, we show that the EERP is NP-complete.

Theorem 2. *EERP is NP-complete.*

Proof. Proof. To show that the EERP is NP-complete we consider its decision version, called EERP-D. EERP-D is defined as follows: Given a fuel consumption network $G = (V, A)$, a PHEV with available battery SOC C, and a given bound Q on gasoline consumption, is there a path in G from an origin node $O \in V$ to a destination node $D \in V$ such that the total gasoline consumption Γ is at most Q?

The first step is to prove that the EERP-D is in NP by showing that given a certificate $(\mathbf{X}, \mathbf{Y})$, it can be decided in polynomial time that $(\mathbf{X}, \mathbf{Y})$ is a solution to the problem or not. Given a path P, battery SOC C, and gasoline consumption Q, it is easy to check if $(\mathbf{X}, \mathbf{Y})$ is a solution. This involves computing two sums and comparing the results against Q and C, that is, checking $\sum_{(i,j)\in P} w_{ij}^{g} x_{ij} \leq Q$ and $\sum_{(i,j)\in P} w_{ij}^{e} y_{ij} \leq C$. This requires an amount of time linear in the number of arcs in G. Thus, EERP-D is in NP.

The second step is to find a polynomial-time reduction from the decision version of the 0-1 multiple-choice knapsack problem. We consider an instance of the EERP-D on a given path. Such instance is the same decision problem as the EERP-PATH-D. As in Theorem 1,

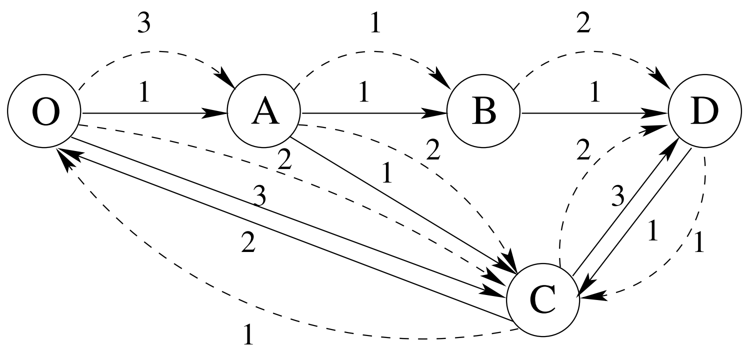

Figure 2.1: EERP: Illustrative example

we proved that there is such a polynomial time reduction for an instance of MCKP-D to EERP-PATH-D, thus, there exits a polynomial time reduction for an instance of MCKP-D to an instance of EERP-D.

Therefore, EERP-D is NP-complete as its special case EERP-PATH-D is NP-complete. Since EERP-D is the decision version of EERP it follows that EERP is also NP-complete.

$\square$

2.2.1 Illustrative Example

To illustrate the EERP, we present an example shown in Figure 2.1. The multigraph in the example consists of two kinds of arcs: solid arcs and dotted arcs. The solid arcs correspond to arcs in set A^g (i.e., arcs corresponding to gasoline mode), while the dotted arcs correspond to arcs in set A^e (i.e., arcs corresponding to electrical mode). The values on the solid arcs correspond to w^g_{ij}, while the values on the dotted arcs correspond to w^e_{ij}. We assume that the battery has SOC $C = 3$ units. The goal is to find a path from O to D minimizing the gasoline consumption.

A naïve approach to solve the EERP is to use a greedy-based method, where we first find a path with minimum gasoline consumption from O to D. The path can be found by using Dijkstra's algorithm on only gasoline consumption network. In this example, the selected path is $((O, A), (A, B), (B, D))$. Then, the naïve approach solves the EERP on the given path by first running on electric mode until the SOC is fully depleted and then operating only on gasoline mode for the rest of the path. This approach may result in a path far from the optimal in terms of fuel economy. In this example, the result of this approach is the path $((O,A)^e,(A,B)^g,(B,D)^g)$ with a gasoline consumption given by $w^g_{AB}+w^g_{BD}$, that is 2 units. However, the optimal solution is the path $((O,A)^g,(A,B)^e,(B,D)^e)$ with a gasoline consumption given by w^g_{OA}, that is 1 unit.

2.3 Exact Algorithms

In this section, we propose two algorithms that find the exact solution to the EERP. The algorithms, called Exact-EER-I and Exact-EER-II, find the path from O to D that leads to the minimum gasoline consumption while satisfying the battery SOC constraint.

The fuel consumption network, considered in this study, is a general graph without any assumptions on the order of the nodes. As a result, simple dynamic programming fails to obtain an optimal solution. To address this problem, we propose two-phase approaches that find exact solutions to the EERP. For the first phase, we propose a dynamic programming recurrence to find an initial gasoline consumption from O to all the nodes for a specific battery SOC. However, the obtained results do not guarantee the optimality of the solution. This is due to the fact that for any battery SOC, the gasoline consumption to reach a node from O is calculated in each iteration of dynamic programming, where those values may have effect on gasoline consumption to reach other nodes on that iteration. Since the nodes do not have any order, the gasoline consumption of some nodes may be decreased by using the just updated values, where the dynamic programming cannot capture these updates during that iteration. To obtain the optimal solution, we propose the second phase, where we update the results obtained by the first phase in each iteration. Note that using only the first phase leads to an error from the optimal value, and this error cannot be bounded.

2.3.1 Basic Algorithm: Exact-EER-I

In this subsection, we propose an exact algorithm, Exact-EER-I, to solve the EERP. Exact-EER-I consists of two phases in each iteration.

Exact-EER-I employs a dynamic programming table Γ with $C+1$ columns and $|V|$

rows. We denote by $\Gamma(j,c)$ the minimum gasoline consumption for the subproblem that considers finding a path from O to j when the available battery SOC is not greater than c. Exact-EER-I initializes $\Gamma(j,0)$ to be the minimum gasoline consumption between nodes O and j, for all j, when there is no battery charge available. Note that $\Gamma(j,0)$ can be obtained using Dijkstra's algorithm on the gasoline consumption network. Then, the dynamic programming table is calculated using a two-phase approach for each iteration (column) $c = 1, \ldots, C$ as outlined next.

In the first phase of iteration c, to find $\Gamma(j,c)$ for a battery SOC c, Exact-EER-I computes whether using the electric mode on any of the incoming arcs to node j leads to less gasoline consumption. Given the additional battery SOC unit, the following recurrence function investigates the possibility of gasoline consumption savings from employing the electric mode for all arcs $(i,j) \in A$ that earlier employed gasoline mode:

$$\Gamma(j,c) = \min_{\forall (i,j) \in A} \{\Gamma(i, c - w^e_{ij}), \Gamma(j, c-1)\}. \tag{2.14}$$

The recurrence compares the value of two cases: First, it uses the minimum gasoline consumption of reaching node i with electricity capacity $c - w^e_{ij}$, and then uses electricity to traverse (i,j). Second, it uses the minimum gasoline consumption to reach node j with electricity capacity $c-1$. The minimum between $\Gamma(i, c-w^e_{ij})$ and $\Gamma(j, c-1)$ gives an initial value of $\Gamma(j,c)$.

We denote by $\Delta(i,j)$ the minimum gasoline consumption between nodes i and j without consuming electricity. Computing $\Delta(i,j)$, $\forall i,j \in V$, is equivalent to computing all-pairs shortest paths on (V, A^g). This can be done in $O(|V|^2 \log(|V|) + |V||A|)$ using Johnson's algorithm.

In the second phase of iteration c, we investigate the possibility of further gasoline consumption savings for node j by leveraging any new savings observed in the first phase (i.e., $\Gamma(i,c)$ for all nodes that observed new savings in the first phase) coupled with pre-computed optimal all gasoline paths from these promising nodes to j (i.e., $\Delta(i,j)$). Exact-EER-I updates $\Gamma(j,c)$ by considering only gasoline consumption on the path $(i \to j)$, for

Algorithm 1 Exact-EER-I (G, OD, C)

```
1: $\Delta(i, j) \leftarrow$ Shortest path on $A^g$ from $i$ to $j$, $\forall i, j \in V$
2: for all nodes $j \in V$, $j \neq O$ do
3:     $\Gamma(j, 0) \leftarrow \Delta(O, j)$
4: end for
5: $\Gamma(O, 0) \leftarrow 0$
6: for all $c = 1$ to $C$ do
7:     for all nodes $j \in V$ do
8:         $\Gamma(j, c) \leftarrow \Gamma(j, c-1)$
9:         for all arcs $(i, j) \in A$ do
10:            if $w^e_{ij} \leq c$ then
11:                $\Gamma(j, c) = \min\{\Gamma(j, c), \Gamma(i, c - w^e_{ij})\}$
12:            end if
13:        end for
14:    end for
15:    for all nodes $j \in V$ do
16:        for all nodes $i \in V$, $i \neq j$ do
17:            $\Gamma(j, c) = \min\{\Gamma(j, c), \Gamma(i, c) + \Delta(i, j)\}$
18:        end for
19:    end for
20: end for
21: $\Gamma^* = \Gamma(D, C)$
22: Find $(\mathbf{X}^*, \mathbf{Y}^*)$ by looking backward
23: Find $P^*$ from $(\mathbf{X}^*, \mathbf{Y}^*)$
24: Output: $\Gamma^*, (\mathbf{X}^*, \mathbf{Y}^*), P^*$
```

all $i \in V$, as follows:

$$\Gamma(j, c) = \min_{\forall i \in V}\{\Gamma(j, c), \Gamma(i, c) + \Delta(i, j)\}. \tag{2.15}$$

$\Gamma(j, c)$ is updated by considering the path from O to node i obtained from the first phase, and then the path from node i to j using only gasoline mode, for all i. The minimum among all these paths from O to j gives the optimal value of $\Gamma(j, c)$ for SOC c.

Exact-EER-I is given in Algorithm 1. The Exact-EER-I algorithm has three input parameters, a multigraph $G\,(V, A)$, an OD pair, and the battery SOC C. The algorithm has three output parameters: Γ^*, the optimal gasoline consumption; $(\mathbf{X}^*, \mathbf{Y}^*)$ the optimal solution; and P^* the optimal path. We assume that all-pairs shortest paths, $\Delta(i, j)$, are precomputed.

Exact-EER-I starts by setting $\Delta(i,j)$ to the minimum gasoline consumption between nodes i and j (line 1). Then, it sets $\Gamma(j,0)$ to $\Delta(O,j)$ for all $j \neq O$ (lines 2-3) and $\Gamma(O,0)$ to zero (line 4). The algorithm then updates the dynamic programming table for each column $c \leq C$ (lines 5-13). For each capacity c, the algorithm first finds $\Gamma(j,c)$ in the first phase (lines 6-10), then updates $\Gamma(j,c)$ in the second phase (lines 11-13). Note that the second phase uses the precomputed values of $\Delta(i,j)$.

Once the algorithm reaches column C of the table and finishes the two phases, it finds Γ^* (line 14). At the end, Exact-EER-I finds $(\mathbf{X}^*, \mathbf{Y}^*)$, and P^* by back-tracking the optimal path from $\Gamma(D,C)$. We note that a key novel characteristic of the algorithm is that the second phase calculations are only dependent on results from the first phase and pre-computed all gasoline all to all shortest paths. This enables the algorithm to identify optimal solutions for general graphs in just two phases for any given column (i.e., battery SOC) of the dynamic programming table.

Exact-EER-I solves the EERP optimally in $O(C(|A| + |V|^2))$, where $|V|$ is the number of nodes, $|A|$ is the number of arcs, and C is the battery SOC. The algorithm builds a table, where the rows are the nodes and the columns are the possible capacities. For each node, all its adjacent nodes are analyzed. As a result, for all nodes the computation takes $O(|A|)$. In addition, for each node j, the algorithm updates the minimum gasoline consumption using the all-pair shortest paths from all nodes $i \in V$. This takes $O(|V|^2)$ for all nodes. Since in general, the capacity C is not bounded by a polynomial in the number of arcs in G, the Exact-EER-I algorithm is a pseudo-polynomial time algorithm.

Theorem 3. *The Exact-EER-I algorithm finds the optimal solution to the EERP.*

Proof. Proof. To prove that $\Gamma(D,C)$ computed by Exact-EER-I is optimal, from the principle of optimality, we need to show that $\Gamma(j,c)$ is optimal, for every node j and battery SOC c. We assume $\Gamma^*(j,c)$ is the optimal solution. We consider an arc $(i,j) \in A$ that is in the optimal path. The proof has two cases.

In case one, in the optimal path, arc (i,j) is assumed to be traversed by electric mode, then $\Gamma^*(j,c) = \Gamma(i, c - w^e_{ij})$. As a result, we need to check if $\Gamma(i, c - w^e_{ij})$ is also optimal.

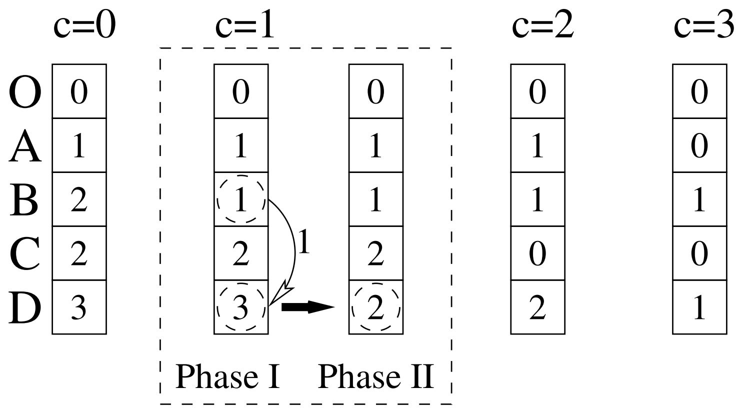

Figure 2.2: Exact-EER-I: Illustrative example

The proof is by contradiction. If $\Gamma(i, c - w^e_{ij})$ is not optimal, then there would be a better solution $\Gamma'(i, c - w^e_{ij}) < \Gamma(i, c - w^e_{ij})$. Given that arc (i, j) is traversed by electric mode, the gasoline consumption of passing through (i, j) is zero. Then, using $\Gamma'(i, c - w^e_{ij})$, we have:

$$\Gamma'(j, c) = \Gamma'(i, c - w^e_{ij}) < \Gamma(i, c - w^e_{ij}) = \Gamma^*(j, c).$$

Thus, $\Gamma'(j, c) < \Gamma^*(j, c)$, which contradicts the fact that $\Gamma^*(j, c)$ is the optimal solution. Therefore, $\Gamma(i, c - w^e_{ij})$ is optimal.

In case two, in the optimal path, arc (i, j) is assumed to be traversed by gasoline mode, hence, $\Gamma(j, c) = \Gamma(i, c) + \Delta(i, j)$, where i is the first node in the path such that $\Gamma(i, c)$ is not optimal. Since $\Gamma(i, c)$ is not optimal and given that node i is the first node where optimality of $\Gamma(i, c)$ is compromised, based on case one, the mode of arrival to node i could not have been electric and has to be gasoline mode. That means, there exists node h in the path, where $\Gamma^*(i, c) = \Gamma(h, c) + \Delta(h, i)$ and $\Gamma(h, c) = \Gamma^*(h, c)$, based on our assumption that i is the first node that $\Gamma(i, c)$ is not optimal. However, $\Gamma(j, c) = \Gamma^*(h, c) + \Delta(h, j) = \Gamma^*(j, c)$, where $\Delta(h, j) = \Delta(h, i) + \Delta(i, j)$ contains an optimal path from i, contradicting the possibility that $\Gamma(i, c)$ is not optimal. This proves the second case.

We conclude that $\Gamma^*(j, c) = \Gamma(j, c)$, and that this property is maintained for all j and c. $\square$

Illustrative Example

To illustrate how the Exact-EER-I works, we present an example in Figure 2.2. We use the same fuel consumption network as in Figure 2.1. This figure shows a dynamic programming table, where there is a row for each node of the network, and a column for each battery SOC $c = 0, 1, 2, 3$.

We only illustrate the results of the calculations for both phases of the column for a battery SOC of one unit (i.e., $c = 1$). This is due to the fact that only for this column the values of the second phase are different than those of the first phase. For $c = 0$, since there is no battery SOC available, the values of the dynamic programming table are initialized using the costs from the precomputed all to all shortest paths of the gasoline consumption network (i.e., $\Delta(i, j)$).

For brevity, we illustrate the calculations for only the cells that have changed their values for column $c = 1$ (these cells are highlighted by the dotted circles). In the first phase, when we examine the cost of reaching node B from the origin, the first phase identifies the opportunity to traverse from node A to node B by electric mode using the unit charge available for the column. This reduces the gasoline cost for the sub-path $O \to B$ from 2 to 1 unit, represented by the path $((O, A)^g, (A, B)^e)$. No other opportunities arise in the first phase for any other node. When we examine the cost of reaching node D from the origin, in the second phase, the algorithm recognizes the opportunity to reduce gasoline consumption by employing the just updated optimal sub-path $O \to B$ (in phase one), and traversing from B to D using all gasoline mode (i.e., $\Delta(B, D)$), reducing the cost from 3 units in column $c = 0$ represented by the path $((O, A)^g, (A, B)^g, (B, D)^g)$ to 2 units in column $c = 1$ represented by the path $((O, A)^g, (A, B)^e, (B, D)^g)$. The rest of the columns are calculated to find the optimal value of $\Gamma^*(D, 3) = 1$ which represents the path $((O, A)^g, (A, B)^e, (B, D)^e)$. Note that the number of such updates are very large in actual networks to guarantee optimality. To make the point, we choose to present this highly stylized example with limited updates.

2.3.2 Improved Algorithm: Exact-EER-II

In this subsection, we propose another exact algorithm, Exact-EER-II, to solve the EERP with better time complexity than Exact-EER-I. The algorithm also eliminates the need for pre-computing all-pairs shortest paths for the gasoline network, which is a very time consuming task.

Similar to Exact-EER-I, Exact-EER-II also employs a two-phase approach for updating each column of the dynamic programming table $\Gamma(j, c)$. In fact, the first phase of the algorithm is exactly identical to the first phase procedure of Exact-EER-I. However, Exact-EER-II uses a priority queue to impose an order for the nodes to be explored for the update process in the second phase to guarantee optimality of the solution.

To consider paths that use gasoline in traversing the last arc, Exact-EER-II uses a priority queue $\mathcal{Q}$ to impose an order for the nodes based on their obtained gasoline consumption in the first phase.

In the second phase, Exact-EER-II uses the values based on the priority queue, $\mathcal{Q}$, on all nodes to reduce the number of calculations while guaranteeing optimality. The priority queue explicitly recognizes the fact that no node that is lower in $\mathcal{Q}$ can improve the highest priority node through an all gasoline path from the lower priority node to the highest priority node. The priority queue $\mathcal{Q}$ for the nodes is based on $\Gamma(j, c)$ resulting from the first phase. The node j with the smallest total gasoline consumption in $\mathcal{Q}$ is extracted, and its $\Gamma(j, c)$ is finalized with the optimal gasoline consumption for that iteration. The algorithm then updates the values of gasoline consumption for all nodes connected to j remaining in $\mathcal{Q}$ by considering traversing arcs from j by gasoline consumption. The keys of the remaining nodes in $\mathcal{Q}$ are updated accordingly as follows:

$$\Gamma(k, c) = \min_{\forall (j,k) \in A} \{\Gamma(k, c), \Gamma(j, c) + w^{g}_{jk}\}. \tag{2.16}$$

Exact-EER-II is given in Algorithm 2. The Exact-EER-II algorithm has three input parameters, a multigraph $G\,(V, A)$, an OD pair, and the battery SOC C. The algorithm has three output parameters: Γ^{*}, the optimal gasoline consumption; $(\mathbf{X}^{*}, \mathbf{Y}^{*})$ the optimal solution; and P^* the optimal path.

Exact-EER-II builds a dynamic programming table where rows are nodes and columns are battery capacities from 0 to C (lines 1-19). The algorithm initializes $\Gamma(j,c)$ for each node j and capacity c (lines 4-9). Then, it updates $\Gamma(j,c)$ if traversing an arc $(i,j) \in A$ using electric mode reduces the total gasoline consumption to reach j (lines 10-12). Therefore, for each node j the algorithm finds its adjacent node i, and reduces $\Gamma(j,c)$ if traveling arc (i,j) in electric mode is beneficial.

By the end of this step, the dynamic programming table contains minimum gasoline consumption to reach node j with capacity c while traversing the last arc (i,j) in electric mode. The obtained gasoline consumption to reach node j is added to a priority queue, $\mathcal{Q}$, to impose an order for the nodes in the second phase (lines 13-14).

Exact-EER-II uses a priority queue, $\mathcal{Q}$, to find the optimal gasoline consumption (lines 15-19). In the priority queue, each node j has a key associated with it which represents the total gasoline consumption from O to j, $\Gamma(j,c)$, considering the maximum battery SOC c. There are three operations associated with priority queue $\mathcal{Q}$: $\mathcal{Q}$.enqueue(), $\mathcal{Q}$.updateKey(), and $\mathcal{Q}$.extractMin(). $\mathcal{Q}$.enqueue() inserts a node in $\mathcal{Q}$ and assigns to it the greatest possible key (i.e., key $= +\infty$). $\mathcal{Q}$.updateKey(j, K) updates the value of the key of node $j \in \mathcal{Q}$ to K if it is less than j's current key. $\mathcal{Q}$.extractMin() extracts the node with the smallest key from $\mathcal{Q}$.

The node with the smallest key (i.e., the node with the smallest total gasoline consumption) is always the first node to be extracted from $\mathcal{Q}$ and its $\Gamma(j,c)$ is finalized in the DP table. In this step, the algorithm updates the values of gasoline consumption by considering all paths to reach node j. Then, the keys of the other nodes in $\mathcal{Q}$ are updated accordingly. Once the algorithm reaches column C of the table and finishes the two phases, it finds Γ^* (line 20). At the end, Exact-EER-II finds $(\mathbf{X}^*, \mathbf{Y}^*)$, and P^* by backtracking $\Gamma(D,C)$.

Exact-EER-II solves EERP optimally in $O(C(|A|+|V|\log(|V|)))$, where $|V|$ is the number of nodes, $|A|$ is the number of arcs, and C is the battery SOC. This is due to the fact that the algorithm builds a table, where the rows are the nodes and the columns are the

Algorithm 2 Exact-EER-II (G, OD, C)

```
 1: for all $c = 0$ to $C$ do
 2:     Create an empty priority queue $\mathcal{Q}$
 3:     for all nodes $j \in V$ do
 4:         if $c = 0$ then
 5:             $\Gamma(j, c) \leftarrow \infty$
 6:         else
 7:             $\Gamma(j, c) \leftarrow \Gamma(j, c - 1)$
 8:         end if
 9:         if $j = O$ then
10:             $\Gamma(O, c) \leftarrow 0$
11:         end if
12:         for all arcs $(i, j) \in A$ do
13:             if $w^e_{ij} \le c$ then
14:                 $\Gamma(j, c) = \min\{\Gamma(j, c), \Gamma(i, c - w^e_{ij})\}$
15:             end if
16:         end for
17:         $\mathcal{Q}$.enqueue($j$)
18:         $\mathcal{Q}$.updateKey($j, \Gamma(j, c)$)
19:     end for
20:     while $\mathcal{Q}$ is not empty do
21:         $(j, \Gamma(j, c)) = \mathcal{Q}$.extractMin()
22:         for all arcs $(j, k) \in A$ and $k \in \mathcal{Q}$ do
23:             $\Gamma(k, c) = \min\{\Gamma(k, c), \Gamma(j, c) + w^g_{jk}\}$
24:             $\mathcal{Q}$.updateKey($k, \Gamma(k, c)$)
25:         end for
26:     end while
27: end for
28: $\Gamma^* = \Gamma(D, C)$
29: Find $(\mathbf{X}^*, \mathbf{Y}^*)$ by looking backward
30: Find $P^*$ from $(\mathbf{X}^*, \mathbf{Y}^*)$
31: Output: $\Gamma^*, (\mathbf{X}^*, \mathbf{Y}^*), P^*$
```

possible battery SOC capacities. For each node, all its visited adjacent nodes are analyzed. This step takes $O(|A|)$. Inserting all nodes in $\mathcal{Q}$ takes $O(|V|)$. In addition, for each node, $\mathcal{Q}$.extractMin() takes $O(\log(|V|)$. Using a Fibonacci heap for implementing the priority queue, $\mathcal{Q}$.updateKey() and $\mathcal{Q}$.enqueue() take $O(1)$ for each node. Therefore, the time complexity of the second step of Exact-EER-II is $O(\sum_{j=1}^{|V|} \log|V| + \text{degree}(j) + 1 * \text{degree}(j)) = O(|A| + |V|\log(|V|))$, where degree($j$) is the number of outgoing arcs of node j. Since in general the capacity C is not bounded by a polynomial in the number of arcs in G, the Exact-EER-II algorithm is a pseudo-polynomial time algorithm.

Theorem 4. *The Exact-EER-II algorithm finds the optimal solution to the EERP.*

Proof. Proof. The proof has two cases. The first case is the same as that of Theorem 3. Here, we only present the proof for the second case, where the optimal path to reach j is to traverse the last arc using gasoline mode. The priority queue $\mathcal{Q}$ in phase two of Exact-EER-II considers all paths from O to j that can lead to improvement using gasoline mode on the last arc. The minimum value among these paths gives the optimal value for $\Gamma(j, c)$. □

2.4 A Fully Polynomial Time Approximation Scheme for EERP: FPTAS-EER

The time complexity of our proposed Exact-EER-I and Exact-EER-II algorithms depends on battery SOC C thus, they are pseudopolynomial. In this section, we propose a fully polynomial-time approximation scheme (FPTAS) for EERP, called FPTAS-EER, to eliminate the dependency of time complexity on the battery SOC. Since EERP is an NP-complete problem, an FPTAS is by far the strongest approximation result that can be achieved unless $P = NP$ [131].

Definition 5 (FPTAS)**.** *A minimization problem has an FPTAS if for every instance I and for every $\epsilon > 0$, it finds a solution S for I in time polynomial in the size of I and in $\frac{1}{\epsilon}$ that satisfies:*

$$S(I) \leq (1 + \epsilon)S^*(I),$$

where $S^*(I)$ is the optimal value of a solution for I.

All the FPTASs proposed in the literature are based on dynamic programming formulations ([136]). There are two methods for transforming an exact dynamic programming-based algorithm into an FPTAS: rounding and scaling, and reducing the state space iteratively. The rounding and scaling method, first introduced by [117], rounds the input data reducing the size of the dynamic programming table. The rounding and scaling method has been extensively used in the design of polynomial approximation schemes. The method of reducing the state space was first introduced by [58]. Our proposed FPTAS is based on the rounding and scaling method, where the rounding is applied to gasoline consumption weights.

Our proposed FPTAS-EER finds a near-optimal solution in polynomial time in the size of the input and $\frac{1}{\epsilon}$, where ϵ is a given error parameter. In order to design such an FPTAS, we need to propose an algorithm that finds an optimal solution using the rounded values, then the solution is rounded back to the original values with some bounded error. Given that battery SOC constraints are hard, we approximate the gasoline consumption in our FPTAS algorithm. We first present an exact algorithm similar to Exact-EER-II that uses the rounded values of gasoline consumption, and then, we propose our FPTAS algorithm. Finally, we analyze the properties of our proposed FPTAS.

2.4.1 Exact-EER-GC Algorithm

In this subsection, we propose a two-phase exact algorithm, called Exact-EER-GC (the name is derived from Exact-EER Gasoline Constrained), similar to Exact-EER-II that uses the rounded values of gasoline consumption. However, Exact-EER-GC determines the path with the minimum gasoline consumption less than a given gasoline capacity Q and with the total battery charge consumption satisfying the battery SOC constraint C.

Exact-EER-GC employs a dynamic programming table E with $Q + 1$ columns and $|V|$ rows. We denote by $E(j, q)$ the optimal battery charge consumption for the subproblem that considers finding the path from O to j where the available gasoline consumption is not greater than q.

In the first phase, Exact-EER-GC finds the initial value of $E(j, q)$ for each node j and gasoline consumption q. Exact-EER-GC uses the following dynamic programming recurrence to find if traversing an arc $(i, j) \in A$ using gasoline reduces the total electricity

Algorithm 3 Exact-EER-GC (G, OD, C, Q): Exact Algorithm

```
1: $E^* \leftarrow \infty$
2: for all $q = 0$ to $Q$ do
3:    Create an empty priority queue $\mathcal{Q}$
4:    for all nodes $j \in V$ do
5:       if $q = 0$ then
6:          $E(j,q) \leftarrow \infty$
7:       else
8:          $E(j,q) \leftarrow E(j,q-1)$
9:       end if
10:      $E(O,q) \leftarrow 0$
11:      for all arcs $(i,j) \in A$ do
12:         if $w^g_{ij} \leq q$ then
13:            $E(j,q) = \min\{E(j,q), E(i,q-w^g_{ij})\}$
14:         end if
15:      end for
16:      $\mathcal{Q}$.enqueue($j$)
17:      $\mathcal{Q}$.updateKey($j, E(j,q)$)
18:   end for
19:   while $\mathcal{Q}$ is not empty do
20:      $(j, E(j,q)) = \mathcal{Q}$.extractMin()
21:      for all arcs $(j,k) \in A^e$ and $k \in \mathcal{Q}$ do
22:         $E(k,q) = \min\{E(k,q), E(j,q) + w^e_{jk}\}$
23:         $\mathcal{Q}$.updateKey($k, E(k,q)$)
24:      end for
25:   end while
26:   if $E(D,q) \leq C$ then
27:      $E^* = E(D,q)$
28:      $\Gamma^* = q$
29:      break;
30:   end if
31: end for
32: Find $(\mathbf{X}^*, \mathbf{Y}^*)$ by looking backward
33: Find $P^*$ from $(\mathbf{X}^*, \mathbf{Y}^*)$
34: Output: $\Gamma^*, E^*, (\mathbf{X}^*, \mathbf{Y}^*), P^*$
```

consumption $E(j,q)$ to reach j:

$$E(j,q) = \min_{\forall (i,j) \in A} \{E(i, q - w^g_{ij}), E(j, q-1)\}. \tag{2.17}$$

To consider paths that use electricity in traversing the last arc, Exact-EER-GC uses a priority queue $\mathcal{Q}$ to impose an order for the nodes based on their obtained electricity consumption in the first phase.

The node with the smallest total electricity consumption (e.g., j) is always the first node to be extracted from $\mathcal{Q}$ and its $E(j,q)$ is finalized in the dynamic programming table. Then, the keys of all nodes connected to j remaining in $\mathcal{Q}$ are updated accordingly as follows:

$$E(k,q) = \min\{E(k,q), E(j,q) + w_{jk}^{e}\}. \tag{2.18}$$

Exact-EER-GC is given in Algorithm 3. The Exact-EER-GC algorithm has four input parameters, a multigraph $G\,(V, A)$, an OD pair, the battery SOC C, and a given gasoline capacity Q. The algorithm has three output parameters: E^{*}, the electricity consumption; $(\mathbf{X}^{*}, \mathbf{Y}^{*})$ the optimal solution; and P^{*} the optimal path. The algorithm is similar in structure to Exact-EER-II with the exception that Γ is replaced by E and the weights corresponding to gasoline consumption are changed to the weights corresponding to battery charge consumption and vice versa. Exact-EER-GC has also an additional input parameter, the gasoline capacity Q.

Exact-EER-GC builds a dynamic programming table, where rows are nodes and columns are gasoline consumption (lines 2-23). The algorithm initializes $E(j,q)$ for each node j and gasoline consumption q (lines 5-9). Then, it updates $E(j,q)$ if traversing an arc $(i,j) \in A$ using gasoline reduces the total electricity consumption to reach j (lines 10-12). Therefore, for each node j the algorithm finds its adjacent node i, and reduces $E(j,q)$ if traveling arc (i,j) with gasoline is beneficial.

By the end of this step, the dynamic programming table contains minimum electricity consumption to reach node j with gasoline consumption q while traversing the last arc (i,j) with gasoline. The obtained gasoline consumption to reach node j is added to a priority queue, $\mathcal{Q}$, to impose an order for the nodes in the second phase (lines 13-14).

Exact-EER-GC uses a priority queue, $\mathcal{Q}$, to find the optimal electricity consumption (lines 15-19). In the priority queue, each node j has a key associated with it which represents the total electricity consumption from O to j, $E(j,q)$, considering the maximum gasoline consumption q. The usage of $\mathcal{Q}$ is the same as in Exact-EER-II and we will not describe it here.

The node with the smallest key (i.e., the node with the smallest total electricity consumption) is always the first node to be extracted from $\mathcal{Q}$ and its $E(j, q)$ is finalized in the DP table. In this step, the algorithm updates the values of electricity consumption by considering all paths to reach node j. Once the final values are determined, the algorithm checks if the obtained $E(D, q)$ satisfies the battery SOC constraint (lines 20-23), if not, E^* has its initial value which is $+\infty$. Based on the determined value q, Γ^* is set to q (the optimal gasoline consumption). At the end, the algorithm finds $(\mathbf{X}^*, \mathbf{Y}^*)$, and P^* by backtracking the optimal solution.

Exact-EER-GC solves EERP optimally in $O(Q(|A| + |V| \log(|V|)))$, where $|V|$ is the number of nodes, $|A|$ is the number of arcs, and Q is the given gasoline capacity. The derivation of the time complexity of Exact-EER-GC is similar to that presented for Exact-EER-II and it will not be presented here. Note that Q is a given value for the gasoline capacity and it is not known a priori. By controlling the given gasoline capacity value, the algorithm finds a path with minimum gasoline consumption satisfying the battery SOC constraint.

2.4.2 FPTAS-EER Algorithm

In this subsection, we propose our FPTAS algorithm, called FPTAS-EER. As noted earlier, FPTAS-EER rounds the arc weights on the gasoline consumption network, iteratively, to reduce the size of the dynamic programming table. The iterative procedure Exact-EER-GC is used as a subroutine on the rounded problems, to tighten the lower and upper bounds, L and U, respectively. The iterative procedure terminates when $U/L < 2$. Once the iterative procedure terminates, the final problem with properly updated arc weights for the network is solved one last time using Exact-EER-GC, which satisfies the guaranteed optimality bound. The details follow.

The gasoline consumption weights w_{ij}^{g} for all arcs $(i,j) \in A$ are updated as follows:

$$w_{ij}'^{g} \leftarrow \left(\left\lfloor \frac{w_{ij}^{g}|V|}{Q\delta} \right\rfloor + 1 \right) \frac{Q\delta}{|V|}, \tag{2.19}$$

where δ is an adaptive factor for the approximation, set to $\sqrt{U/L} - 1$, and Q is an arc weight rounding parameter selected to lie between L and U and chosen based on the results of Exact-EER-GC, as follows: $Q = \sqrt{LU/(1+\delta)}$. During the procedure of closing the gap to less than L, δ is calculated based on the new values of L and U.

In each iteration of FPTAS-EER, the number of columns in the dynamic programming table are determined by δ. When δ is large, there are fewer columns, and Exact-EER-GC is faster. Given that the bounds are updated in every iteration, they tighten faster initially with larger δ. As the gap decreases, δ becomes smaller, to increase precision. After reaching a gap less than L, δ is set to ϵ.

A key property of our proposed FPTAS is that, it rounds up the weights of the arcs. Therefore, there is no arc with a new weight of zero. This property leads to optimal results using Exact-EER-GC such that after rounding back to the original values, the results are within the bounded error.

FPTAS-EER is given in Algorithm 4. The FPTAS-EER algorithm has four input parameters, a multigraph $G(V, A)$, an OD pair, the battery SOC C, and a given ϵ. The algorithm has three output parameters: $\hat{\Gamma}$, $(\hat{\mathbf{X}}, \hat{\mathbf{Y}})$, and $\hat{P}$, where $\hat{\Gamma}$ is the near-optimal gasoline consumption for the path, $(\hat{\mathbf{X}}, \hat{\mathbf{Y}})$ is the near-optimal solution, and $\hat{P}$ is the near-optimal path.

The FPTAS-EER starts by setting the lower bound on gasoline consumption to 1 (line 1). The upper bound on gasoline consumption is set to the gasoline consumption obtained by computing the shortest path on the gasoline consumption network (line 2). This upper bound can be found by using Dijkstra's algorithm on gasoline consumption network in $O(|A| + |V| \log |V|)$.

Then, FPTAS-EER iterates to reduce the gap between the lower and upper bounds to less than L (lines 3-13). In each iteration, the gasoline consumption Q is updated

Algorithm 4 FPTAS-EER(G, OD, C, ϵ)

```
1: $L \leftarrow 1$
2: $U \leftarrow$ gasoline consumption determined as a shortest path from $O$ to $D$
3: **while** $U > 2L$ **do**
4:     $\delta = \sqrt{U/L} - 1$
5:     $Q = \sqrt{LU/(1+\delta)}$
6:     **for all** arcs $(i, j) \in A$ **do**
7:         $w'^{g}_{ij} \leftarrow \lfloor \frac{w^{g}_{ij}|V|}{Q\delta} \rfloor + 1$
8:     **end for**
9:     $Q' \leftarrow \lfloor |V|/\delta \rfloor + 1$
10:    $(E, (\mathbf{X}, \mathbf{Y}), P)$ =Exact-EER-GC$(G(V, A'), OD, C, Q')$
11:    **if** $E \leq C$ **then**
12:        $U \leftarrow (1+\delta)Q$
13:    **else**
14:        $L \leftarrow Q$
15:    **end if**
16: **end while**
17: **for all** arcs $(i, j) \in A$ **do**
18:     $w'^{g}_{ij} \leftarrow \lfloor \frac{w^{g}_{ij}|V|}{\epsilon L} \rfloor + 1$
19: **end for**
20: $Q' \leftarrow \lfloor |V|/\epsilon \rfloor + 1$
21: $(\hat{E}, (\hat{\mathbf{X}}, \hat{\mathbf{Y}}), \hat{P})$ =Exact-EER-GC$(G(V, A'), OD, C, Q')$
22: Calculate $\hat{\Gamma}$ based on original value of $\hat{P}$
23: **Output:** $\hat{\Gamma}, (\hat{\mathbf{X}}, \hat{\mathbf{Y}}), \hat{P}$
```

based on the current lower and upper bounds (line 5). The algorithm uses an adaptive parameter δ set to $\sqrt{U/L} - 1$ (line 4). FPTAS-EER rounds the gasoline consumption of each arc (lines 6-7). Then, it calls Exact-EER-GC using the rounded values w'^{g}_{ij} stored as A', and a gasoline capacity Q' as the input parameters (line 9). If there is no feasible solution, then the lower bound is updated to Q, otherwise the upper bound is updated to $(1+\delta)Q$. Finding a feasible solution by Exact-EER-GC means that with a given total gasoline consumption Q', we have $E(D, Q') \leq C$.

At the end of the while loop iterations, FPTAS-EER finds a lower bound and an upper bound on gasoline consumption, where the gap between them is less than the lower bound. FPTAS-EER then rounds the gasoline consumption of each arc based on the latest lower bound and ϵ (lines 14-15). Finally, FPTAS-EER calls Exact-EER-GC to solve the problem using the rounded values based on the obtained lower bound stored as A' (line 17).

2.4.3 Properties of FPTAS-EER

In this subsection, we analyze the properties of the proposed FPTAS-EER. We first prove that our proposed approximation algorithm is an FPTAS, that is, for every fixed ϵ, its running time is polynomial in the size of the input and in $\frac{1}{\epsilon}$. We then compare the time complexity of FPTAS and Exact-EER-II.

Theorem 5. *The FPTAS-EER algorithm is an FPTAS.*

Proof. Proof. To prove that the algorithm is FPTAS, we need to show that the solution determined by the algorithm is in a $(1+\epsilon)$ neighborhood of the optimal, and that the time complexity of the algorithm is polynomial in the size of the input and in $\frac{1}{\epsilon}$.

First, we show that the solution obtained by FPTAS-EER is within $(1+\epsilon)$ of the optimal solution. Let Γ^* be the optimal gasoline consumption, and Γ be the obtained solution by FPTAS-EER. FPTAS-EER finds a lower bound L and an upper bound U, where $L \leq \Gamma \leq U$. If $U \leq (1+\epsilon)L$, then Γ is within $(1+\epsilon)$ of the optimal solution Γ^*. If $U > (1+\epsilon)L$, we can select Q such that $L < Q < U(1+\epsilon)^{-1}$. Using Exact-EER-GC with selected Q can improve the bounds such that either U is decreased to $Q(1+\epsilon)$ (line 11 in Algorithm 4) or L is increased to Q (line 13 in Algorithm 4). The while loop (lines 3-13) continues to reduce the ratio of U/L below a constant (here 2). Then, FPTAS-EER calls Exact-EER-GC with rounded values for arcs' gasoline consumption for all arcs in A based on the determined lower bound L. Since FPTAS-EER replaces the arcs' gasoline consumption by $w'^{g}_{ij} \leftarrow (\lfloor \frac{w^{g}_{ij}|V|}{\epsilon L} \rfloor + 1)\frac{\epsilon L}{|V|}$, for each arc, we have $|w^{g}_{ij} - w'^{g}_{ij}| \leq \frac{\epsilon L}{|V|}$. That means, there is an error of at most $\frac{\epsilon L}{|V|}$ for each arc. In the worst case, where the number of arcs in the path is $|V|-1$, the total error is $\frac{\epsilon L(|V|-1)}{|V|}$, which is less than ϵL. Since $L \leq \Gamma^*$, we have $\epsilon L < \epsilon \Gamma^*$. Therefore, Γ is within $(1+\epsilon)$ of the optimal solution Γ^*.

We now show that the time complexity of FPTAS-EER is polynomial in the number of nodes, the number of arcs, and $\frac{1}{\epsilon}$. The time complexity of FPTAS-EER is given by the time complexity of its three major parts. In the first part, the upper bound U is determined by finding the shortest path from O to D on gasoline consumption network using Dijsktra's algorithm (line 2) in $O(|A|+|V|\log|V|)$. In the second part, the algorithm reduces the gap between the upper and lower bounds (lines 3-13). The running time of this step depends on the number of iterations and the time complexity of Exact-EER-GC in each iteration. The time complexity of Exact-EER-GC in iteration i is $O(|V|/\delta_i(|A| + |V| \log |V|))$, where $|V|$ is the number of rows and $|V|/\delta_i$ is the number of columns based on the rounding. For each node, Exact-EER-GC checks all its adjacent nodes such that $\sum_{j \in V} \sum_{i;(i,j) \in A} 1 = |A|$, and needs $O(|V| \log |V|)$ for the updates. The algorithm adapts parameter δ based on the gap between the upper bound and the lower bound in each iteration i such that if the gap is large, the algorithm applies a coarse approximation, whereas if the gap becomes narrower, the algorithm applies a finer approximation. That is, the algorithm reduces the gap faster by coarse approximations, and then applies more precise approximations. The total run time of the second step is $\sum_i |V|/\delta_i(|A| + |V| \log |V|) = (|V|^2 \log |V| + |A||V|) \sum_i 1/\delta_i$, where $\sum_i 1/\delta_i$ is $O(1)$. As a result, the second part of the algorithm has time complexity $O(|V|^2 \log |V| + |A||V|)$. In the third part, the algorithm calls Exact-EER-GC using the rounded values based on the determined lower bound. The time complexity of this step is $O((|A| + |V| \log |V|)|V|/\epsilon)$, where $|V|$ is the number of rows and $|V|/\epsilon$ is the number of columns based on the rounding. Thus, the time complexity of the algorithm is $O(((|V|^2 \log |V| + |A||V|)/\epsilon)$. This concludes that the algorithm is FPTAS.

□

In the following, we compare the performance of FPTAS-EER and Exact-EER-II. The time complexity of FPTAS-EER is less than Exact-EER-II if and only if $\frac{|V|}{\epsilon} < C$. This is driven from the time complexity of the Exact-EER-II algorithm and the FPTAS-EER algorithm, which are $O(C(|A|+|V| \log(|V|)))$ and $O(((|V|^2 \log |V|+|A||V|)/\epsilon)$, respectively. This indicates that in a setting where $\frac{|V|}{\epsilon} \geq C$, it is better to use the pseudo polynomial algorithm Exact-EER-II in terms of execution time. In addition, Exact-EER-II obtains optimal results. Note that if C is bounded and $|V|$ is not, it is best to use Exact-EER-II instead of FPTAS-EER.

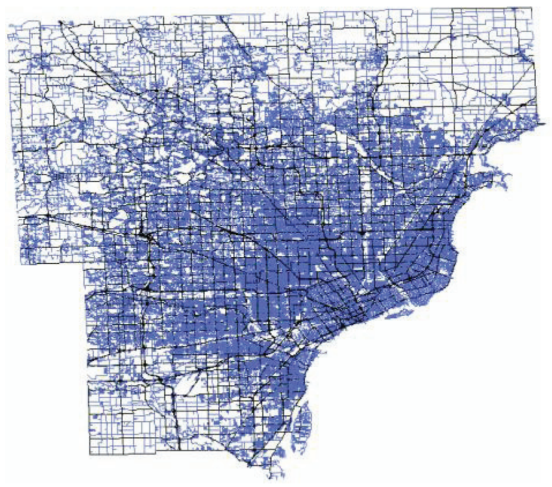

Figure 2.3: Southeast Michigan road network: 465,938 arcs and 168,806 nodes

2.5 Experimental Results

We extract real road network features of Southeast Michigan from data provided by [88] using ArcGIS Desktop 10.1. Figure 2.3, shows the full road network of Southeast Michigan. The extracted data from Southeast Michigan map consists of 465,938 arcs and 168,806 nodes along with their longitudes and latitudes, speed limits, travel time, distance of road segments, etc.

The proposed algorithms are implemented in C++, and the experiments are conducted on an Intel 3.3GHz with 48GB RAM. In this section, we describe the experimental setup and analyze the experimental results.

2.5.1 Generating Multigraph Road Networks

Given the absence of gasoline and battery consumption data for any production PHEV for all the arcs of Southeast Michigan, we adopt the following procedure for generating this data. For estimating the gasoline consumption for different arcs of the network for a hypothetical PHEV, we rely on fuel economy (mpg) by speed (mph) plots available in the public domain for different vehicles [21]. In particular, we employ the following expression that estimates the gasoline mode fuel economy (mpg) as a function of posted speed limit (PSL) in mph for any road segment: $45 - 0.015(PSL - 45)^2$. The gasoline consumption for the segment can then be calculated readily based on the length of the segment. For estimating the battery charge consumption rate during electric mode of the PHEV (kWh/mile), we rely on battery consumption rate (kWh/mile) by speed (mpg) data posted by actual users of hybrid vehicles in public domain sites (e.g., [107]). We employ the following expression to estimate the electric mode battery charge consumption rate (kWh/mile) as a function of PSL for any road segment: $0.18581 + 0.00321(PSL) - 0.00011(PSL)^2 + 0.0000014(PSL)^3$. The total battery charge consumption for the segment can then be calculated readily based on the length of the segment.

2.5.2 Analysis of Results

We compare the performance of Exact-EER-I, Exact-EER-II, and FPTAS-EER on different network sizes. In subsection 2.5.2, we present the results for Exact-EER-II on the whole Southeast Michigan network. In subsection 2.5.2, we present the results for all the proposed algorithms on a small network selected from Southeast Michigan network. The reason that we present the results for large and small networks separately is that Exact-EER-I is not able to find the optimal results in reasonable amount of time due to its large memory requirements, and the setting for Southeast Michigan network is such that $\frac{|V|}{\epsilon} < C$ for the chosen range of C, therefore, as discussed in Section 2.4, we only present the results for Exact-EER-II. We present the performance of Exact-EER-I and FPTAS-EER on a smaller network within Southeast Michigan network.

Large scale networks.

We evaluate the performance of Exact-EER-II on actual Southeast Michigan road network with all its 465,938 arcs and 168,806 nodes using 3,000 randomly generated OD pairs. To analyze effects of OD pairs distance on the proposed algorithm, our tests are executed on

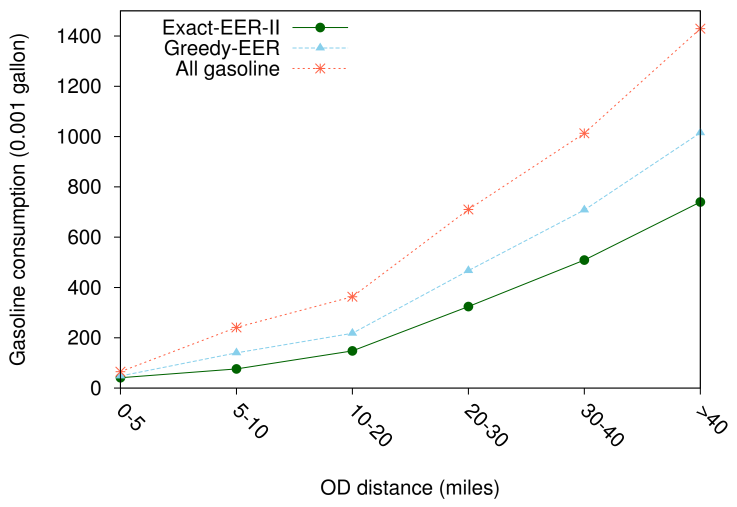

Figure 2.4: Gasoline consumption (0.001 gallon)

six different classes of OD pairs distance: less than 5 miles, 5 to 10 miles, 10 to 20 miles, 20 to 30 miles, 30 to 40 miles, and more than 40 miles (the longest distance is 59 miles). Each class consists of 500 random OD pairs. The available battery SOC (at the start of the trip) in the classes are 200 Wh, 1000 Wh, 2000 Wh, 3000 Wh, 4000 Wh, and 5000 Wh, respectively. The selected battery SOCs are realistic for PHEVs in production currently available in the market.

Note that, if the OD distance is within all electric range and the battery is at its full charge, the algorithm only selects electric mode for all the arcs in the path to optimize fuel economy. In fact, there is no point in minimizing gasoline consumption in this scenario. It is for this reason that we choose smaller available battery SOC for shorter distances.

We compare the performance of our proposed algorithm, Exact-EER-II, with that of a greedy based algorithm called Greedy-EER. In addition, we present the minimum gasoline consumption without considering electric mode for the selected OD pairs. The Greedy-EER algorithm incorporates a greedy approach to solve the EERP as follows. It first finds the route with minimum gasoline consumption without considering the electric mode. Then, the algorithm selects to run the vehicle in electric mode along the established route until the battery SOC is depleted, and then reverts to the gasoline mode for the rest of the trip. Several production vehicles are known to currently employ this greedy approach. While the

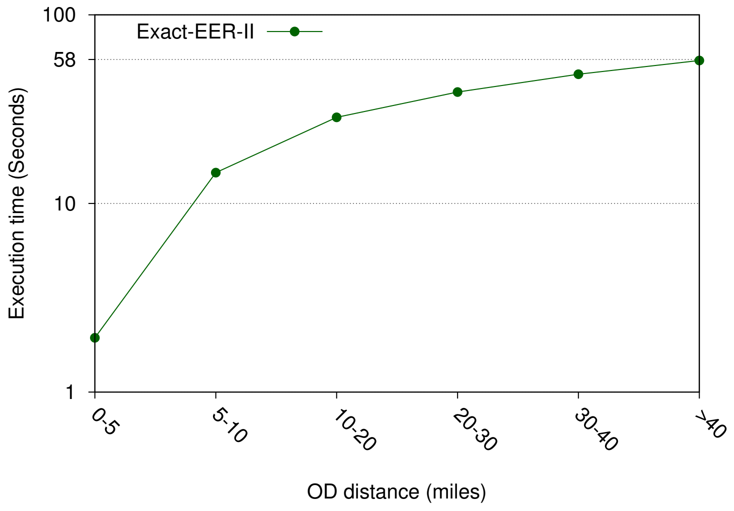

Figure 2.5: Exact-EER-II Execution time (Seconds)

energy management control algorithms of production PHEVs do not currently have access to the route plan, they try to greedily claim as many miles as possible in the electric mode before being forced to switch to the gasoline mode.

Figure 2.4 shows the average gasoline consumption in 0.001 gallon units obtained by the algorithms. The results show that Greedy-EER can be far from the optimal solution provided by Exact-EER-II. With increase in OD distance and battery SOC, the gap between Greedy-EER and Exact-EER-II increases. This is due to the fact that Exact-EER-II takes into account the energy efficiency differences of the vehicle operating modes in jointly selecting the path, and with a higher SOC the vehicle can have more choices on the operating mode in order to reduce gasoline consumption.

Figure 2.5 presents the execution time of the Exact-EER-II. The execution time of Exact-EER-II depends on the battery SOC, which is higher for longer OD distances. It is worth mentioning that CPLEX 12 solver cannot find the optimal solution for the EERP for any of the selected OD pairs even after 3600 seconds. Since the execution time of Greedy-EER is negligible, we do not present it here.

In Figure 2.6 and 2.7, we compare the gasoline consumption obtained by Exact-EER-II and Greedy-EER. Figure 2.6 shows the gasoline consumption ratio of Greedy-EER over the optimal solution obtained by Exact-EER-II. In the less than five miles class, the ratio

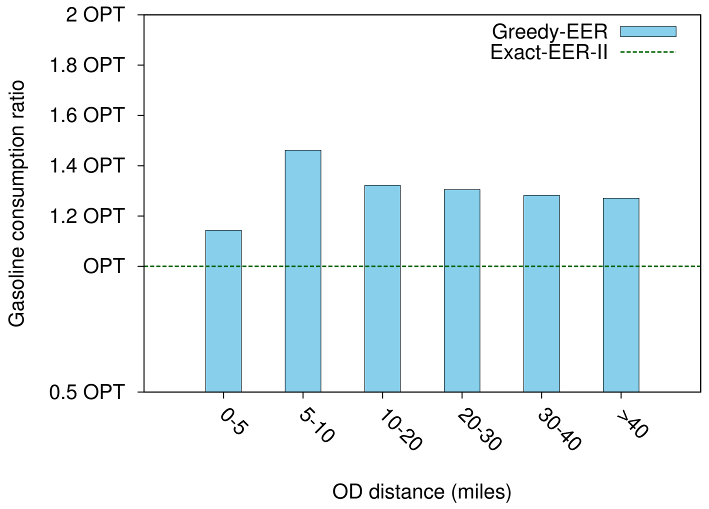

Figure 2.6: Gasoline consumption ratio (OPT: consumption obtained by Exact-EER)

is 1.14, which is the lowest ratio among all classes of OD pairs. This is due to fact that the battery SOC has its lowest value (i.e., 200 Wh) leading to using electric modes on a few arcs in the selected path. As a result, the obtained results by both algorithms are close. However, for the class of 5-10 miles, where the battery SOC is 1000, this ratio increases to 1.46. This is due to the fact that Exact-EER-II uses electric mode on the selected path more efficiently, and there are more arcs that are traversed by the electric mode compared to the total number of arcs along the path. For the remaining OD pair classes, the ratio is higher than 1.27, which shows the significant energy efficiency of our proposed algorithm. Figure 2.7 shows the difference of gasoline consumption obtained by Exact-EER-II compared to that of the Greedy-EER. This value can be interpreted as the gasoline saving when using our proposed algorithm. It is clear that with higher battery SOC, Exact-EER-II is able to save more gasoline.

For all the above results, we conclude that Exact-EER-II not only provides energy efficient solutions, but also obtains them in a reasonable amount of time. However, the execution time of Exact-EER-II depends on the available battery SOC.

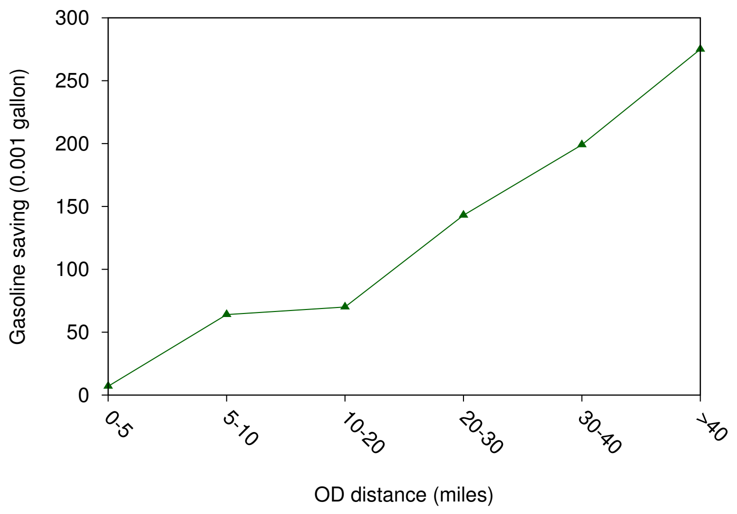

Figure 2.7: Gasoline saving of Exact-EER-II compared to Greedy-EER

Small networks.

We evaluate the performance of the proposed algorithms in the generated multigraph based on a selected small network with 2,000 nodes extracted from Southeast Michigan data provided by [88] using ArcGIS Desktop 10.1.

We compare the performance of our proposed algorithms, Exact-EER-I, Exact-EER-II, and FPTAS-EER. In addition, we present the results of the Greedy-EER (as we explained in the previous subsection), and the minimum gasoline consumption only on the gasoline consumption network, called All gasoline. We select randomly 100 OD pairs by considering several battery SOC between 50,000 to 250,000 mWh. For FPTAS-EER, we set $\epsilon = 0.1$.

Figure 2.8 shows the average gasoline consumption in 0.001 gallon units obtained by FPTAS-EER, Greedy-EER, and All gasoline algorithms. For all the instances, FPTAS-EER obtains the optimal solution, the same as that obtained by both Exact-EER-I and Exact-EER-II. Note that the gap between gasoline consumption obtained by Greedy-EER and the optimal gasoline consumption is low due to the small range of battery SOC and the size of the network.

We perform sensitivity analysis for the battery SOC parameter. Figure 2.9 presents the execution time of our proposed algorithms with several values of battery SOC. This sensitivity analysis on the available battery SOC C, clearly shows that unlike Exact-EER-I

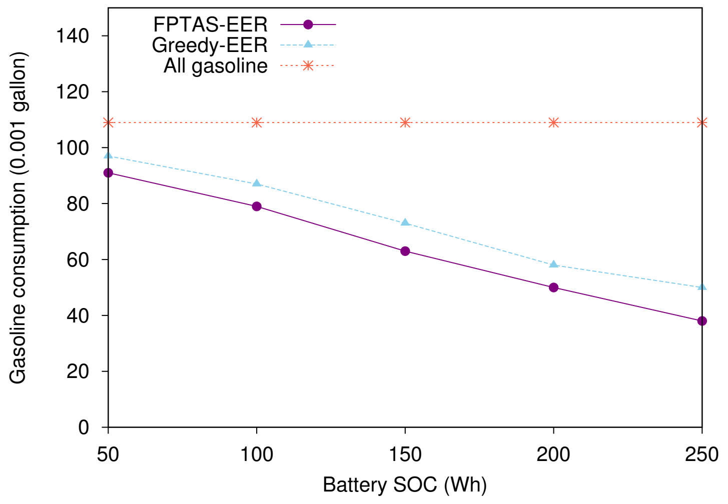

Figure 2.8: Effect of battery SOC on the gasoline consumption of the algorithms

and Exact-EER-II, the performance of the FPTAS-EER does not depend on C. However, the execution time of both exact algorithms increases by increasing the battery SOC, since they have pseudopolynomial time complexity in terms of the battery SOC. As we expected based on the time complexity, Exact-EER-I has a higher execution time than that of Exact-EER-II. In addition, FPTAS-EER performs much faster than Exact-EER-II when the network size is small.

From all the above results, we conclude that applying our proposed energy-efficient routing algorithms over the current static energy management systems can lead to significant fuel consumption savings (reaching over 25% for OD distances exceeding 5 miles). For example, as shown in the results, our proposed algorithms can save over 0.25 gallon of gasoline compared to the currently employed static algorithm for one long trip (more than 40 miles). Note that this improvement depends on the type of production of PHEVs, terrain geometry, traffic dynamics, payload, etc. These features can be captured into our proposed multigraph fuel consumption network as a prepossessing step for the routing algorithms. We suggest that based on the size of the network and the SOC, one can incorporate Exact-EER-II or FPTAS-EER for energy-efficient route planning of PHEVs.

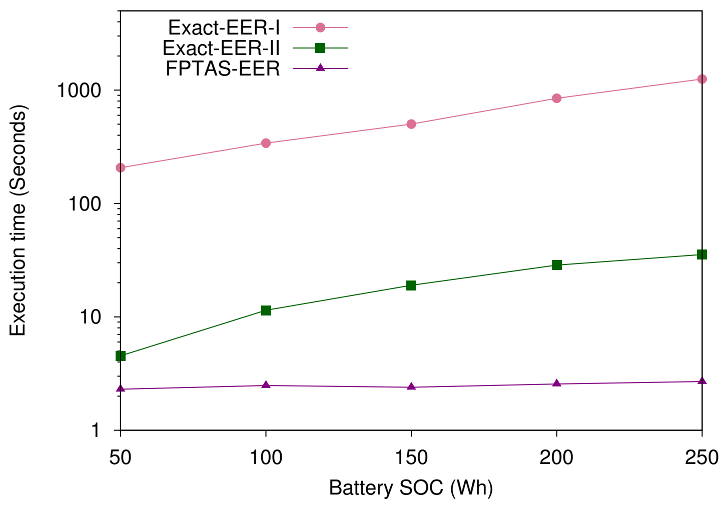

Figure 2.9: Effect of battery SOC on the execution time of the algorithms

2.6 Conclusion

In this study, we introduced the Energy-Efficient Routing problem (EERP) for Plug-in Hybrid Electric Vehicles. We presented the hardness proof of the EERP. We then proposed two exact pseudopolynomial algorithms and an FPTAS algorithm to solve the EERP.

From an algorithmic perspective, the proposed two-phase approaches improve the state of the art in optimally solving shortest path problems on general constrained multi-graph networks. In the context of vehicle routing, this is the first study to take into account energy efficiency difference of different operating modes of PHEVs during route planning, which is a high level power-train energy management procedure. Experimental evaluations of the proposed algorithms on Southeast Michigan road network demonstrate significant fuel economy improvement potential (exceeding 25% improvement in fuel efficiency over currently common greedy methods for trips exceeding 5 miles in distance). In addition, the results show the computational efficiency and accuracy of the proposed algorithms.

CHAPTER 3: ONLINE SCHEDULING AND PRICING FOR ELECTRIC VEHICLE CHARGING

3.1 Introduction

Electrified vehicles promise to enable diversification of the transportation energy feedstock for reduced dependence on petroleum for transport, improve public health by lessening greenhouse gas emissions, and stimulate economic growth through the development of new technologies and industries. Widespread adoption of electrified vehicles is in alignment with sustainable transportation objectives in its social, economic, and environmental perspectives. Automotive companies are being challenged by environmentally conscious consumers and governments to produce affordable electrified vehicles [56]. Several companies from around the world have accepted the challenge and more models of plug-in EVs (plug-in hybrid electric vehicles and pure battery electric vehicles) that can be charged from the electric grid are being introduced [57]. Unlike standard hybrid vehicles, plug-in hybrid EVs (PHEV) also offer the ability to be recharged from an external electrical outlet. While pure battery electric vehicles (BEV) currently offer limited driving range, PHEVs carry an internal combustion engine besides an electric motor to overcome driving range issues.

Achieving large-scale adoption of EVs presents a number of challenges resulting from a current lack of supporting technologies/infrastructure and difficulties in overcoming technological barriers. Currently, EV drivers face long vehicle charging cycles times. In addition, they may also face long waiting times and uncertainty over availability of charging facilities. As EV usage for daily commute increases, the consideration of the ability to recharge these vehicles both in and away from base locations (e.g., residential locations) becomes more important. For example, some EV drivers may want to recharge their EVs at their destination locations such as workplaces, where their vehicles are parked for an extended duration. For electric utility companies, the concern is high electricity consumption of EVs that makes the load management of micro grids a challenge. For example, EVs require up to three times the maximum current demand of a typical home and can overload micro grids [30]. The impact on the grid is especially critical during peak grid demand hours. The existing electricity infrastructure may not be adequately designed to satisfy the surge in power demand under these situations.

While the utility companies will in the long-run work to address capacity shortages, they can benefit from the development of scheduling and pricing mechanisms for EV charging that are cost effective while providing good service. They seek to deploy mechanisms that lead to balanced network load over time. One way to reach a better load balance is dynamic and preemption-aware scheduling. However, the problem of efficient scheduling and fair pricing of EV charging services is challenging, especially as both EV drivers and power providers can be seen as self-interested parties (drivers are interested in minimizing their costs and maximizing convenience, whereas utility companies would like to maximize their profits). When an EV is available for charging over an extended period (e.g., 8 AM to 4 PM), charging mechanisms can service that request (i.e., provide the charge) either in one continuous time slot, or in several discrete shorter time slots. The charging interruption may be due to arrival of other urgent requests or the need for grid load balancing and necessitates preemption of scheduled requests.

Electric utility companies can also choose to sell their unallocated capacity in an auction platform, where EV users can obtain charging units at lower prices. This is a win-win scenario for both providers and users, which allows providers to increase revenues while users obtain charging units at lower prices. Such auction platforms will be of particular interest for PHEV users since they are not faced with range anxiety associated with pure BEVs (i.e., fear that a vehicle will run out of battery charge enroute). When the battery within a PHEV is depleted, the ICE works as a backup, providing a driving range comparable to conventional internal combustion engine vehicles.

EV drivers that are not in urgent need of charging but looking for bargains can also benefit from the auction-based platform. In the rest of this study, we use the term EVs for users of electric vehicles interested in using this platform. EV users in urgent need for charging and not willing to risk preemption can either bid high on the platform or use a conventional charging platform (which is outside the scope of this work).

In this study, we propose the first preemption-aware online mechanism for scheduling and pricing EV charging in an auction-based platform. Users represent EV drivers whose charging requests arrive dynamically over time, at which point they name their own price (place their bids) for a certain amount of charge by their departure. Our goal is to assure that the micro grid capacity constraints are not exceeded, and those users who value the electricity the most are allocated and scheduled. We allow preemption of the requests to manage load and revenue of the provider.

This is a competitive environment where EV drivers compete for the limited supply of an electric utility provider. These EV drivers are strategic users who are self-interested, meaning that they are interested in maximizing their own utility. Different users may have different time constraints and willingness to pay. These users act strategically to maximize their own utility, and they may misreport their preferences if this is in their best interest. Our goal is to design model-free mechanisms (i.e., we make no assumptions about future demand) that incentivises users to reveal their true preferences. Our proposed mechanisms consist of a scheduling algorithm and a dynamic pricing scheme for EVs charging management considering realtime demand.

3.1.1 Our Contribution

We introduce the problem of preemption-aware online scheduling and pricing (OSAP) for EV charging. OSAP problem involves the realtime scheduling of requests released over time (i.e., EVs that require a certain amount of charge by their departure) sharing a scarce resource (i.e., electricity that is limited), and given uncertainty about future arrivals. We first propose an integer program to find the optimal schedule for the offline version of the problem, where all information about future supply and demand is known to the scheduler. We then propose an optimal offline mechanism using the proposed off-line scheduler and the

VCG (Vickrey-Clarke-Groves) pricing scheme. In addition, we design a family of online mechanisms that solve the OSAP problem, where the requests arrive dynamically over time. The mechanisms are model-free, make no assumption about future demand, and are invoked when a user places a new request or additional electricity capacity becomes available. We prove that all our proposed mechanisms are incentive-compatible. This property incentivizes the EV users to report their preferences truthfully. We perform extensive experiments and show that our proposed online mechanisms are able to find near optimal solutions while satisfying the incentive-compatibility property.

3.1.2 Related Work

Research on different decision problems related to electric vehicles has attracted a great deal of attention in the past few years. Such research include, forecasting the EV market share [44], designing energy-efficient routing of PHEVs [119], and proposing battery-swapping polices [73, 3]. [65] proposed a hybrid simulation approach to estimate the evolution of EV market shares. [15] investigated the impact of collaboration on the adoption of EVs among commercial fleets using concepts from cooperative game theory. [68] proposed a framework for optimizing the driving range by minimizing the sum of battery price, electricity cost, and range limitation cost as a measurement of range anxiety. [127] studied the problem of returning electrical load to the grid, known as vehicle-to-grid, to reduce stress on the grid during peak times by injecting power back into the grid.

Automatic scheduling of EV charging has been studied from different points of view and considered different applications. [19] proposed a coordinated charging scheduler in order to minimize the power losses and to maximize the grid load factor. [128] proposed a load flow method for the problem of charging multiple EVs. However, strategic behavior of users (i.e., systematic manipulation of the system to gain unfair advantage) remains possible in their settings, where users misreport their preferences in order to receive preferential charging, leading to inefficient schedules that are not based on true users' requests. [36] proposed a decentralized algorithm to optimally schedule EV charging by exploiting the elasticity of EV loads to fill the valleys in electric load profiles. [60] investigated offline and online EV charging scheduling problems from a user's perspective by jointly considering the aggregator's revenue and users' demands and costs. [130] proposed a lottery-based solution for EV scheduling in order to ensure a level of fairness in the resulting scheduling in which a lottery system decides whether to charge a vehicle or not. However, none of these studies considered strategic users. In addition, they did not consider pricing.

Pricing EV charging is another line of research. [124] investigates the incentives of EV drivers in making charging decisions with different electricity tariffs. In addition, he compares the cost and emissions impacts of these charging patterns to the ideal case of charging controlled by the system operator. [135] developed optimal electricity storage control policies to manage charging and discharging activities for PHEVs. Their proposed models capture the impact of the charging and discharging activities on real-time electricity prices. [32] proposed a charging coordination model considering a spatial price component in order to analyze the loads from price-based EV fleet charging while at the same time accounting for distribution grid constraints. [7] proposed a distributed dynamic pricing mechanism for the charging of PHEVs in a smart grid architecture. Once again, none of the above mentioned studies considered strategic users.

There is an extensive body of literature on mechanism design in scheduling that considers strategic users; the reader is referred to [54] for a survey. Mechanism design theory has been employed in designing incentive-compatible mechanisms in several areas including spectrum auctions and cloud computing. In spectrum auctions, a government or a primary license holder sells the right to use a specific frequency band in a specific area using auction-based mechanisms (e.g., [138, 62]). In cloud computing, a cloud provider offers computing services as commodities. Amazon Elastic Compute Cloud (Amazon EC2) offers auction-based cloud services through its spot instance, where users can bid on spare Amazon EC2 virtual machine instances. Several mechanisms have been designed for cloud auction markets (e.g., [67, 137]). Problems arising from each area have their own specific characteristic leading to fundamentally different problems. The unique characteristics of EV charging, e.g., allowing preemption and online setting, brings about new challenges in designing market mechanisms, and the existing mechanisms fail when applied to the EV charging problem. We focus on studies on online mechanism design, where users arrive dynamically over time. Online mechanism design is an important topic in the multi-agent and economics literature. The reader is referred to [101] for an introduction to online mechanisms. One line of research in designing online mechanisms is to develop online variants of Vickrey-Clarke-Groves (VCG) mechanisms [43, 102]. These studies focus on Bayesian-Nash incentive compatibility. However, the focus of this study is on the stronger concept of incentive compatible dominant-strategies. In addition, these studies are model based, and they rely on a model of future availability of supply and demand, while our proposed mechanisms are model-free. [52] and [106] considered model-free settings. [106] proposed an incentive-compatible mechanism for online scheduling of jobs on a single machine. [52] studied the problem of online scheduling of a single, re-usable resource over a finite time period. They proved the incentive-compatibility of their proposed mechanisms and derived lower bound competitive ratios. [66] considered multi-unit demand, and proposed an online auction model. In their model, however, the auctioneer must respond to each request immediately before considering other requests.

A number of studies have considered scheduling of EV charging with strategic users. [126] proposed a model based online mechanism for pure electric vehicle charging. They introduced the use of pre-commitment in order to guarantee incentive-compatibility of their proposed mechanism. In such a setting, when a mechanism precommits to a request, the request is neither preempted nor canceled. [41] proposed an online auction protocol for EV charging. In order to satisfy the incentive-compatibility property, their mechanism allows burning of allocated electricity to some PHEVs. They showed that their proposed mechanism provides higher allocative efficiency than a fixed price system. [114] proposed an online mechanism with strategic EV drivers allowing burning units. [113] proposed an online mechanism for multi-unit demand and studied its application for charging PHEVs. They proposed two truthful allocation algorithms based on a greedy online assignment algorithm. Their allocation algorithms also allow occasional burning of allocated electricity to some PHEVs in order for their mechanisms to be incentive compatible. [42] proposed a two-sided online mechanism for advance reservations of charging, where EV users and providers can specify their preferences on time slots and number of units per time slot. Overall, our setting is more complex in comparison with the above-mentioned studies by jointly considering strategic users, allowing preemptions, and being model-free. The proposed preemption-aware scheduling and pricing mechanisms are also compatible with the load balancing objectives of utility providers. All these properties of the proposed mechanisms make them more compatible with the real-world settings.

3.2 Online Scheduling and Pricing Problem

In this section, we model the online scheduling and pricing (OSAP) problem for an electric utility provider that is providing charging service for EV users in a competitive environment. The utility provider is assumed to carry a limited electricity capacity C^t for EV charging during a discrete interval (of arbitrary choice) but the capacity might vary randomly from interval to interval by time of the day, $t \in \mathcal{T}$. Users compete for this limited supply while arriving dynamically over time at discrete intervals. User i requests l_i units of charge over a specified discrete interval $[a_i, d_i]$ and is willing to pay a maximum price of v_i if the service is completed on time. In this study, we consider that one unit of charging requires a unit of time, thus, users are requesting the charging units in terms of the amount of time that their EVs require to be charged. User i's bid (request) is denoted by $\beta_i = (a_i, l_i, d_i, v_i)$. For example, bid $(2, 1, 7, \$15)$ represents a user requesting 1 unit of charging, where the request arrives at time 2, expires at time 7, and her maximum price for the charging service is \$15.

The utility provider is able to (re)schedule the charging services for the different users at the arrival of any new user bid and/or change in available capacity.

We denote by $X_{N\times T}$ the charging schedule for all N users in the set $\mathcal{U}$ of users and T number of time intervals in the problem horizon, where x_i^t is 1 if user i's EV is scheduled for charging at time t, and 0, otherwise. Vector $X_i = (x_i^0, \ldots, x_i^T)$ represents the charging schedule for user i over time. Since the preemption of service is allowed, user i's charging might be completed over different intervals with interruptions. We denote by Π_i user i's payment for receiving the charging service.

Each user i is characterized by a *valuation function* V_i defined as follows:

$$V_i(X_i) = \begin{cases} v_i & \text{if} \quad \sum_{t=a_i}^{d_i} x_i^t \geq l_i \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where X_i is the charging schedule of user i. We denote by W the *social welfare*, which is defined as the sum of users' valuations (i.e., the set of users with active service requests): $W = \sum_{i \in \mathcal{U}} V_i(X_i)$.

Given this setting, the problem of online scheduling and pricing of EV charging is to find a charging schedule and charging prices for users such that the total social welfare is maximized.

We denote by $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_N)$ the vector of requests of all N users, and by $\boldsymbol{\beta}_{-i}$ the vector of all requests except user i's request (i.e., $\boldsymbol{\beta}_{-i} = (\beta_1, \ldots, \beta_{i-1}, \beta_{i+1}, \ldots, \beta_N)$). We quantify user i's benefit through a *quasi-linear utility function* defined as the difference between the value she receives and the payment charged to her:

$$U_i(\boldsymbol{\beta}) = V_i(X_i) - \Pi_i \tag{3.2}$$

The users are self-interested, that is, they want to maximize their own utility. It may be beneficial for them to manipulate the service system and gain unfair advantage through untruthful reporting. A user can declare a higher value in the hope to increase the likelihood of obtaining her requested charging service. Strategic behaviors of such users may hinder other qualified users, leading to reduced revenue and reputation of the provider. With the increase in the number of EVs requiring charging, the potential for systematic manipulation will become a significant concern for utility providers. Our goal is to design incentive-compatible (strategy-proof) mechanisms that solve the OSAP problem and discourage users from gaming the system through untruthful reporting.

The utility provider is also self-interested and wants to maximize its profit. In this setting, our goal is to give incentives to the utility provider to fulfill the entire request of a user rather than a partial allocation. In doing so, the utility provider receives payment from a user only if it provides her entire requested charging units. Note that in the absence of such setting, the utility provider can maximize its profit by greedily allocating charging units only to the users with the highest value per unit of charging at any time leading to the fractional OSAP problem (i.e., users are willing to pay for any fraction of their received request). Although, such strategy would result in higher profit for the utility provider, it does not consider the incentives of the users who want their entire requested charging units.

In an online setting, where complete information about future demand and supply is not available, designing an optimal mechanism is not possible. However, in an offline setting of the OSAP problem (SAP problem), we assume that such information is available, and thus, designing an optimal mechanism is possible. In the next section, we propose an optimal offline mechanism for the SAP problem that is used as a benchmark for evaluating the performance of our proposed online mechanisms.

3.3 Optimal Offline Mechanism

In this section, we propose an optimal offline strategy-proof mechanism for SAP, which considers that the information on all the future requests as well as supply is known *a priori*. A set $\mathcal{U}$ of N users submit their requests for the planning horizon of interest. We denote by $\hat{\beta}_i = (\hat{a}_i, \hat{l}_i, \hat{d}_i, \hat{v}_i)$ user i's declared request and valuation. Note that $\beta_i = (a_i, l_i, d_i, v_i)$ is user i's request and true valuation. Users are rational in the sense that they do not want to pay more than their valuation for their requests. A well-designed mechanism should incentivize users to participate. Such a property of a mechanism is called individual rationality and is defined as follows:

Definition 6 (Individual rationality)**.** *A mechanism is* individually-rational *if for every user i with true request β_i and the set of other requests $\boldsymbol{\beta}_{-i}$, we have $U_i(\beta_i, \boldsymbol{\beta}_{-i}) \geq 0$.*

In other words, a mechanism is individually-rational if a user can always achieve as much utility from participation as without participation. However, such mechanisms do not always incentivize users to report their requests truthfully. Our goal is to design a mechanism that is strategy-proof, i.e., a mechanism that incentivizes users to reveal their true requests.

Definition 7 (Strategy-proofness [101]). *A mechanism is* strategy-proof *(or incentive compatible) if* $\forall i \in \mathcal{U}$ *with a true request declaration* β_i *and any other declaration* $\hat{\beta}_i$*, and* $\forall \hat{\boldsymbol{\beta}}_{-i}$*, we have that* $U_i(\beta_i, \hat{\boldsymbol{\beta}}_{-i}) \geq U_i(\hat{\beta}_i, \hat{\boldsymbol{\beta}}_{-i})$.

The strategy-proofness property implies that truthful reporting is a dominant strategy for the users. As a result, it never pays off for any user to deviate from reporting her true request, irrespective of the actions of the others.

Our first proposed strategy-proof mechanism is optimal, and is based on the Vickrey-Clarke-Groves (VCG) mechanism. An optimal schedule with VCG payments provides a strategy-proof mechanism [132, 17, 50]. We define our proposed optimal VCG-based mechanism for SAP as follows:

Definition 8 (VCG-SAP mechanism). *The VCG-SAP mechanism consists of a scheduling function* $\mathcal{S}$ *and a payment function* Π*, where*

i) $\mathcal{S}$ *is an optimal scheduling function maximizing the social welfare, such that* $X_i = \mathcal{S}_i(\hat{\boldsymbol{\beta}})$*, and*

ii) $\Pi_i(\hat{\boldsymbol{\beta}}) = \sum_{j \in \mathcal{U} \setminus \{i\}} V_j(\mathcal{S}_j(\hat{\boldsymbol{\beta}}_{-i})) - \sum_{j \in \mathcal{U} \setminus \{i\}} V_j(\mathcal{S}_j(\hat{\boldsymbol{\beta}}))$,

such that $\sum_{j \in \mathcal{U} \setminus \{i\}} V_j(\mathcal{S}_j(\hat{\boldsymbol{\beta}}_{-i}))$ *is the optimal social welfare obtained when user i is excluded from participation, and* $\sum_{j \in \mathcal{U} \setminus \{i\}} V_j(\mathcal{S}_j(\hat{\boldsymbol{\beta}}))$ *is the sum of all users' valuations in the optimal solution except user i's value.*

Overall, we first identify the winning users and their optimal charging schedules. The prices are then determined based on the VCG pricing scheme.

In order to find the optimal scheduling function, we propose an Integer Program (IP) and define the decision variables over time $t \in \mathcal{T}$ as follows:

$$x_i^t = \begin{cases} 1 & \text{if a charging unit is allocated to user } i \text{ at } t, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

$$y_i = \begin{cases} 1 & \text{if any charging unit is allocated to user } i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

In addition, we define indicator parameters as follows:

$$\delta_i^t = \begin{cases} 1 & \text{if } a_i \leq t \leq d_i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

To maintain optimality, the solution should either fully service any particular request or not provide any service. The feasibility of the schedule to user i is indicated by δ_i^t. This indicator parameter ensures that the requested units are scheduled within time window $[a_i, d_i]$, if we choose to service the request.

The problem that needs to be solved to identify the winning bids and their optimal charging schedule can be formulated as an integer program (called SAP-IP), as follows:

$$\text{Maximize} \sum_{i\in\mathcal{U}} v_i \cdot [(\sum_{t\in\mathcal{T}} \delta_i^t x_i^t) - (l_i - 1)y_i] \tag{3.6}$$

Subject to:

$$\sum_{t\in\mathcal{T}} x_i^t \leq l_i,\ \forall i \in \mathcal{U} \tag{3.7}$$

$$\sum_{i\in\mathcal{U}} \delta_i^t x_i^t \leq C^t,\ \forall t \in \mathcal{T} \tag{3.8}$$

$$x_i^t \leq y_i, \forall i \in \mathcal{U},\ \forall t \in \mathcal{T} \tag{3.9}$$

$$x_i^t = \{0,1\}, \forall i \in \mathcal{U},\ \forall t \in \mathcal{T} \tag{3.10}$$

$$y_i = \{0,1\}, \forall i \in \mathcal{U} \tag{3.11}$$

$$\delta_i^t = \{0,1\}, \forall i \in \mathcal{U},\ \forall t \in \mathcal{T} \tag{3.12}$$

The objective function is to maximize the sum of all N users valuations. Only the values of the users who receive their complete charging requests are considered in the objective function. However, their allocation might be completed over different intervals (with interruptions) as long as they are within their requested time interval. Constraints (3.7) ensure that each user is serviced at most the requested amount. Constraints (3.8) guarantee that the allocation does not exceed the available capacity for any given time. Constraints (3.10) and (3.12) represent the integrality requirements for the decision variables and indicator parameters.

Once solved, SAP-IP finds the winning bids and their optimal charging schedule. The charging prices are then determined based on the VCG pricing scheme that also employs SAP-IP as a subroutine.

The execution time of VCG-SAP becomes prohibitive for large instances of the SAP problem. However, in an online setting, we do not have information about future bid requests or the capacity fluctuations, and thus, we resort to designing fast online mechanisms providing approximate solutions for the OSAP problem. Our goal is to design such online incentive-compatible mechanisms that solve the OSAP problem effectively. The VCG-SAP mechanism will be used in our experiments purely as a benchmark for assessing the performance of the proposed online mechanisms.

3.4 Strategy-proof Online Mechanisms

In this section, we propose incentive-compatible mechanisms (called MOSAP) for the OSAP problem. The goal of the mechanisms is to compute an efficient schedule even if $\hat{\beta}_i \neq \beta_i$ and calculate payments that incentivize users to report their true requests. To obtain a strategy-proof mechanism, the scheduling function $\mathcal{S}$ must be monotone, and the payment function Π must be based on the critical payment [85]. In the following, we define the properties that our proposed mechanisms need to satisfy in order to guarantee strategy-proofness.

We define monotonicity in terms of the following preference relation $\succeq$ on the set of requests. A request $\hat{\beta}'_i = (\hat{a}'_i, \hat{l}'_i, \hat{d}'_i, \hat{v}'_i)$ is more preferred (i.e., $\hat{\beta}'_i \succeq \hat{\beta}_i$) if $\hat{a}'_i \leq \hat{a}_i$, $\hat{l}'_i \leq \hat{l}_i$, $\hat{d}'_i \geq \hat{d}_i$, and $\hat{v}'_i \geq \hat{v}_i$ for user i. That means the request $\hat{\beta}'_i$ is more preferred than $\hat{\beta}_i$ if user i requests less amount of charging units, submits an earlier request, a later deadline, and a higher value. In our setting, for obvious reasons, users have no incentive to report an earlier arrival (i.e., $\hat{a}_i \leq a_i$) or a later deadline (i.e., $\hat{d}_i \geq d_i$) than their true arrival time and true deadline.

The monotonicity property indicates that any winning user who receives her requested charging units by declaring a request $\hat{\beta}_i$ will still be a winner if she requests a more preferred request. A user is a winner if her charging request is accepted and scheduled within her specified time interval. In the following, we describe the monotonicity property.

Definition 9 (Monotonicity). *A scheduling function $\mathcal{S}$ is* monotone *if it selects user i with $\hat{\beta}_i$ as her declared request, then it also selects user i with a more preferred request $\hat{\beta}'_i$, i.e., $\hat{\beta}'_i \succeq \hat{\beta}_i$.*

In addition to a monotone scheduling function $\mathcal{S}$, any strategy-proof mechanism has a payment rule Π satisfying the critical payment property such that the payment of any

Algorithm 5 MOSAP-X(Event, X, Π)

```
1: $t \leftarrow$ Current time
2: $\mathcal{Q}^t \leftarrow \{\hat{\beta}_i | i \in \mathcal{U}$, $i$'s request has not completed yet$\}$
3: $\mathcal{H}^t \leftarrow \{\hat{\beta}_i | i \in \mathcal{U}$, $i$'s request can be completed at $t\}$
4: if $\mathcal{Q}^t = \emptyset$ or $C^t = 0$ then
5:     return
6: end if
7: $X^t \leftarrow$ MOSAP-X-SCH$(X, t, \mathcal{Q}^t, C^t)$
8: $X \leftarrow X \cup X^t$
9: $\Pi^t \leftarrow$ MOSAP-X-PAY$(t, \mathcal{H}^t, X^t, C^t)$
10: $\Pi \leftarrow \Pi \cup \Pi^t$
11: return $X, \Pi$
```

user i, must be independent of her request. In the following, we describe the critical payment property.

Definition 10 (Critical payment). *Let $\mathcal{S}$ be a monotone scheduling function, then for every $\hat{\beta}_i$, there exists a unique value v_i^c, called* critical payment*, such that $\forall \hat{\beta}_i' \succeq (\hat{a}_i, \hat{l}_i, \hat{d}_i, v_i^c)$, $\hat{\beta}_i'$ is a winning declaration, and $\forall \hat{\beta}_i' \prec (\hat{a}_i, \hat{l}_i, \hat{d}_i, v_i^c)$ is a losing declaration. $\Pi_i = v_i^c$ if user i wins, and $\Pi_i = 0$, otherwise.*

In the following, we propose three different mechanisms for the OSAP problem. Since the three mechanisms are similar in structure, we present them as variants of a generic mechanism, called MOSAP-X, where X will be replaced with I, II and III to specify each of the three mechanisms. These mechanisms prioritize users with different metrics such that in each mechanism the selection of the winning users and their schedule and payment might be different than those obtained by other mechanisms based on their given priority. MOSAP-I gives higher priority to users with higher values. However, MOSAP-II gives higher priority to users with higher value per unit of charge, while MOSAP-III determines the priority by taking into account both the value and the partial allocation.

MOSAP-X is given in Algorithm 5, which is an event handler, that is, it is invoked when a new user request arrives or available charging capacity changes. Our proposed mechanisms take as input an event, the current schedule set X, and the payment set Π.

MOSAP-X uses the following four variables defined as:

$\lambda_i^t = \sum_{a_i \leq \tau < t} x_i^\tau$; allocated amount to user i before time t

$\mathcal{Q}^t$: the set of feasible requests of the users that have not been scheduled completely yet (active requests). Formally, $\mathcal{Q}^t \leftarrow \{\hat{\beta}_i | i \in \mathcal{U}, t \leq d_i \wedge \lambda_i^t < l_i \wedge l_i - \lambda_i^t \leq d_i - t\}$;

$\mathcal{H}^t$: the set of requests that can be completed at time t. Formally, $\mathcal{H}^t \leftarrow \{\hat{\beta}_i | i \in \mathcal{U}, t \leq d_i \wedge \lambda_i^t < l_i \wedge \lambda_i^t + 1 \geq l_i\}$

C^t: the available charging capacity at time t.

Considering $\mathcal{Q}^t$, if the mechanism finds a better request than a current allocated request, it will preempt the allocated request with the intention of resuming its allocation at a later time. As a result, all active requests are in set $\mathcal{Q}^t$.

In lines 1 to 3, MOSAP-X sets the current time to t and initializes $\mathcal{Q}^t$ and $\mathcal{H}^t$. Then, it proceeds only if new resources and/or requests are available. MOSAP-X determines the scheduling by calling MOSAP-X-SCH.

The scheduling function MOSAP-X-SCH returns X^t, the set of users who would receive their requested charging units at time t (line 7). The mechanism then updates the overall scheduling set X using the newly determined set X^t (line 8). Then, the mechanism determines the payment of users in X^t by calling MOSAP-X-PAY. The payment function MOSAP-X-PAY returns set Π^t containing the payment of users at time t (line 9). The mechanism updates the overall payment set Π using the newly determined set Π^t (line 10). Finally, the mechanism returns the schedule and payment sets.

Our proposed scheduling algorithm MOSAP-X-SCH is given in Algorithm 6. We consider three algorithm variants for scheduling, MOSAP-I-SCH, MOSAP-II-SCH, and MOSAP-III-SCH. We define a metric called the *priority metric* for each algorithm.

MOSAP-X-SCH algorithm allocates the charging capacity to users in decreasing order of their priority metrics. We define the priority metrics of MOSAP-X-SCH as follows:

1) MOSAP-I-SCH: $f_i = \hat{v}_i$; 2) MOSAP-II-SCH: $f_i = \frac{\hat{v}_i}{\hat{l}_i}$; and 3) MOSAP-III-SCH: $f_i = \frac{(\lambda_i^t+1)\hat{v}_i}{\hat{l}_i}$. The priority metric for MOSAP-I-SCH gives higher priority to users with higher values. MOSAP-II-SCH considers the value per unit of charge as the priority metric.

Algorithm 6 MOSAP-X-SCH($X, t, \mathcal{Q}^t, C^t$)

```
1: $X^t \leftarrow \emptyset$
2: **for all** $i|\hat{\beta}_i \in \mathcal{Q}^t$ **do**
3:     $\lambda_i^t = \sum_{a_i \le \tau < t} x_i^\tau$
4:     $f_i = \hat{v}_i$, for MOSAP-I-SCH; or
       $f_i = \frac{\hat{v}_i}{\hat{l}_i}$, for MOSAP-II-SCH; or
       $f_i = \frac{(\lambda_i^t+1)\hat{v}_i}{\hat{l}_i}$, for MOSAP-III-SCH
5: **end for**
6: Sort all $\hat{\beta}_i \in \mathcal{Q}^t$ in non-increasing order of $f_i$
7: **for all** $\hat{\beta}_i \in \mathcal{Q}^t$ in non-increasing order of $f_i$ **do**
8:     **if** $C^t > 0$ **then**
9:         $C^t = C^t - 1$
10:        $x_i^t = 1$
11:    **else**
12:        **break**;
13:    **end if**
14: **end for**
15: **if** $C^t = 0$ **then**
16:     **for all** $\hat{\beta}_i \in \mathcal{Q}^t$ for which $x_i^{t-1} = 1$ and $x_i^t = 0$ **do**
17:         Preempt user $i$'s request
18:     **end for**
19: **end if**
20: $X^t \leftarrow (x_0^t, \dots, x_N^t)$
21: **Output:** $X^t$
```

MOSAP-III-SCH gives higher priority to the users who have already received a partial allocation of their charging requests.

MOSAP-X-SCH sorts all requests in non-increasing order of priority metrics, f_i (line 6). Then the algorithm schedules the units requested by the sorted users in $\mathcal{Q}^t$ while resources last (lines 7-14). The mechanism uses this ordering for scheduling since the provider is interested in users who want to pay more. MOSAP-X-SCH tries to maximize the sum of the reported values of the users who get their charging units. By allowing preemption, MOSAP-X-SCH allocates charging units to users with higher priority while interrupting the allocation of users who are already allocated and have lower priority than the selected requests at the current time. The lower-priority request is suspended and is resumed as soon as possible (lines 15-19). Such a request is resumed when its priority value compared to those of other active requests is high enough to be selected. Since such a request has already

Algorithm 7 MOSAP-X-PAY($t, \mathcal{H}^t, X^t, C^t$)

```
1: **for all** $i|\hat{\beta}_i \in \mathcal{H}^t$ **do**
2:     $\lambda_i^t = \sum_{a_i \le \tau < t} x_i^\tau$
3:     $f_i = \hat{v}_i$, for MOSAP-I-PAY; or
       $f_i = \frac{\hat{v}_i}{\hat{l}_i}$, for MOSAP-II-PAY; or
       $f_i = \frac{(\lambda_i^t + 1)\hat{v}_i}{\hat{l}_i}$, for MOSAP-III-PAY
4: **end for**
5: **for all** $i|\hat{\beta}_i \in \mathcal{H}^t \wedge x_i^t = 1$ in non-increasing order of $f_i$ **do**
6:     $q = -1$;
7:     $\bar{X} \leftarrow$ MOSAP-X-SCH($t, \mathcal{H}^t \setminus \hat{\beta}_i, C^t + 1$)
8:     **for all** $\hat{\beta}_j \in \mathcal{H}^t \cap \{\hat{\beta}_j | \bar{x}_j^t = 1 \wedge x_j^t = 0\}$ in non-increasing
               order of $f_j$, where $f_j < f_i$ **do**
9:         $q = j$;
10:        **break**;
11:    **end for**
12:    **if** $q$ **then**
13:        $\Pi_i \leftarrow f_q$, for MOSAP-I-PAY and MOSAP-III-PAY; or
           $\Pi_i \leftarrow f_q \hat{l}_i$, for MOSAP-II-PAY
14:    **else**
15:        $\Pi_i \leftarrow r$
16:    **end if**
17: **end for**
18: **Output:** $\Pi^t = (\Pi_1, \ldots, \Pi_N)$
```

received a part of the requested charging units, the mechanisms only need to provide the remaining units of the request in order to complete the request and receive the payment. Finally, MOSAP-X-SCH returns the set X^t of users who are scheduled at time t.

The payment function MOSAP-X-PAY is given in Algorithm 7. This function calculates the *critical payment* of each user i if her EV is scheduled for charging at t. The critical payment of user i is the minimum value that she must report to receive the charging units at time t. MOSAP-X-PAY uses the set $\mathcal{H}^t$ of requests of users who are allocated or not allocated at t. This set does not include requests of users who are scheduled completely before t. MOSAP-X-PAY calculates f_i for all users in $\mathcal{H}^t$ (lines 1-4). Then, MOSAP-X-PAY determines the payment for all users that have been scheduled at time t (i.e., $x_i^t = 1$) and will obtain their full requested charge by t. In doing so, MOSAP-X-PAY calls the scheduling algorithm, MOSAP-X-SCH, without considering the participation of user i and with a capacity of $C^t + 1$ (i.e., the capacity before scheduling user i) (line 7). MOSAP-

X-SCH returns the set of users $\bar{X}$ who would receive their requested charging at time t without user i's participation. Then, MOSAP-X-PAY tries to find a user j who had not been scheduled at t when user i participated (i.e., $x_j^t = 0$), and would have been scheduled at t if user i did not participate (i.e., $\bar{x}_j^t = 1$) (line 8). If MOSAP-X-PAY finds such a user, it stores her index q (line 9), and it determines the payment of user i based on the priority metric of user q (line 13); otherwise user i pays a reserve price $r \geq 0$ (line 15). In other words, the payment of user i is calculated based on the requests of losing users (i.e., that of user q), who would win if user i would not participate. This is the minimum value that needs to be reported by user i to obtain her request. Since the provider wants to guarantee a minimum revenue from each unit sold, the mechanism includes a reserve price. If this minimum price is set the same for all units and at all time points, this would not affect the properties of our proposed mechanisms. Finally, the set Π^t is returned to the mechanism.

Under MOSAP-X, some of the users may not receive all their requested charging units. Even though these units are a few, MOSAP-X can adjust the allocation under well specified conditions. There are two possible ways to handle these partial allocations: burning and on-departure discharge. In burning, units are simply left allocated. For those allocated units, the provider does not receive any payment. In on-departure discharge, on departure of the user's EV, any allocated units are discharged from the battery. The model with on-departure discharge is more efficient in terms of resource utilization from the power provider's perspective, but it is not realistic to expect that we can discharge the partially allocated units from a car's battery on its departure. As a result, MOSAP-X uses burning in the case of partial allocation. The concept of burning has been used in the design of charging mechanisms in the literature (e.g., [113]), and it is proven to be effective in terms of strategy-proofness of the mechanisms.

In addition, preemption allows our mechanisms to be flexible on when the charging takes place. Power providers can utilize such a feature of our mechanisms to shift some charging from peak grid demand hours to reduce stress on the grid during peak times.

Example 1. We show the execution of the mechanism by considering a setting with one unit of capacity available at each time slot and five users, denoted by EV_i, $i = 1, \ldots, 5$, as

	$\hat{\beta}_i$	$\hat{a}_i$	$\hat{l}_i$	$\hat{d}_i$	$\hat{v}_i$
EV_1	$\hat{\beta}_1$	0	3	6	5
EV_2	$\hat{\beta}_2$	0	4	7	4
EV_3	$\hat{\beta}_3$	1	3	6	7
EV_4	$\hat{\beta}_4$	3	6	10	10
EV_5	$\hat{\beta}_5$	3	4	10	8

Table 3.1: User bids

	MOSAP-I f_i	MOSAP-II f_i	MOSAP-III f_i			
			$t = 0$	$t = 1$	$t = 2$	$t = 3$
EV_1	5	1.6	1.6	3.3	5.0	
EV_2	4	1.0	1.0	1.0	1.0	
EV_3	7	2.3		2.3	2.3	2.3
EV_4	10	1.6				1.6
EV_5	8	2.0				2.0
$\mathcal{S}$	$\{\hat{\beta}_4\}$	$\{\hat{\beta}_3, \hat{\beta}_5\}$	$\{\hat{\beta}_1, \hat{\beta}_3, \hat{\beta}_5\}$			
W	10	7+8 = 15	5+7+8=20			

Table 3.2: Execution of MOSAP-X

shown in Table 3.1. For example, user 1's bid $\hat{\beta}_1$ contains the following information: her request is submitted at time 0, with a deadline 6; she requests 3 units of charging with a bidding price 5. Table 3.2 show the execution of all three MOSAP-X-SCH mechanisms. In each column, the value of priority metrics, the set of winning users $\mathcal{S}$, and the obtained social welfare $\mathcal{W}$ are shown. For example, column f_i^I shows the priority metrics in MOSAP-I-SCH, the winning request is $\hat{\beta}_4$, and the obtained social welfare is 10. Figs 3.1-3.3 shows the resulting schedules of the users obtained by the three mechanisms. Using MOSAP-I-SCH, EV_1 is selected at time 0, and then interrupted at time 1, when EV_3 is selected. At time 3, EV_4 is selected because of its highest priority, thus, EV_3 is interrupted. None of other users has higher priority than EV_4 until her EV receives all the requested charging units. At time 9, none of the users are active to receive a charging unit. This scheduling process by MOSAP-I-SCH is shown in Fig. 3.1.

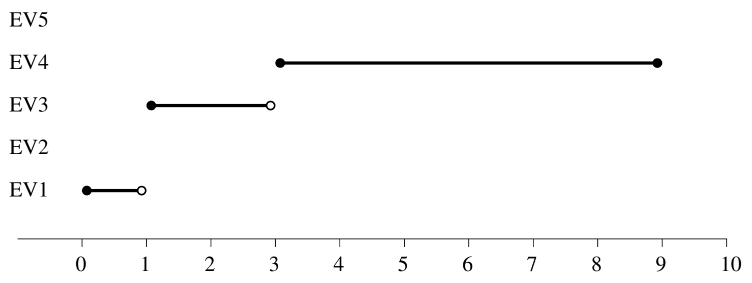

Figure 3.1: MOSAP-I-SCH

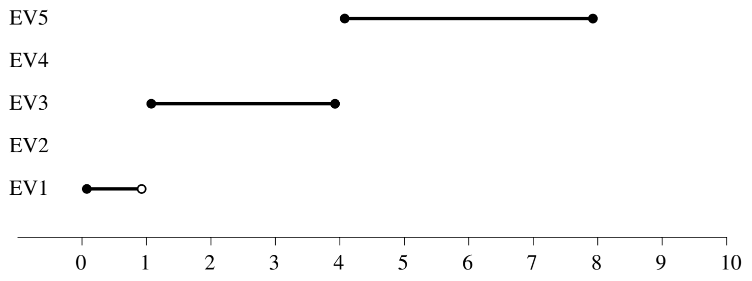

Figure 3.2: MOSAP-II-SCH

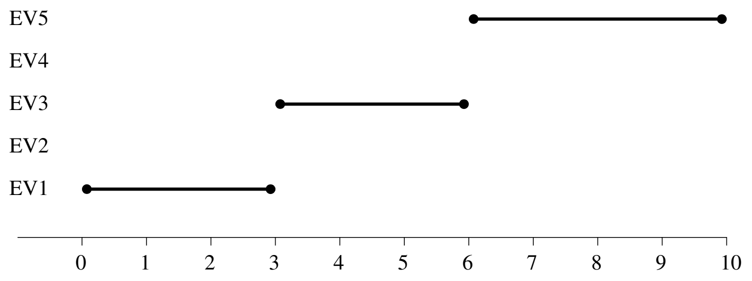

Figure 3.3: MOSAP-III-SCH

3.5 Properties of MOSAP

In this section, we investigate the properties of MOSAP-X. We first show that the mechanisms are *individually rational* (i.e., truthful users will never incur a loss). We then prove several lemmas in order to prove the strategy-proofness of MOSAP-X. At the end, we also present an example to analyze the effect of untruthful reporting on users and the mechanisms.

Theorem 6. *MOSAP-X mechanisms are individually rational.*

Proof. We consider user i as a winning user. We need to prove that if user i reports her true request then her utility is non-negative. This can be easily seen from the structure

of the MOSAP-X mechanisms. In line 13 of Algorithm 3, the payment for user i is set to $\Pi_i = f_q$ for MOSAP-I-PAY and MOSAP-III-PAY, and $\Pi_i = f_q l_i$ for MOSAP-II-PAY, where user q is the user who would have won if user i did not participate. Since user q appears after user i in the decreasing order of the priority metric in each of the selected mechanism, we have, $f_q \leq f_i$, thus, for each payment function, we have:
MOSAP-I-PAY: Because $v_q \leq v_i$ then $\Pi_i \leq v_i$;
MOSAP-II-PAY: $f_q l_i \leq f_i l_i$, thus, $\Pi_i \leq \frac{v_i}{l_i} l_i \leq v_i$;
MOSAP-III-PAY: In the last iteration of finding the priority metric to determine user i's payment, we have $\lambda_i^t = l_i - 1$, thus, $f_i = v_i$. Since $v_q \leq v_i$ then $\Pi_i \leq v_i$;
MOSAP-X-PAY always computes a payment $\Pi_i \leq v_i$. As a result, the utility of user i (i.e., $U_i(\beta_i) = v_i - \Pi_i \geq 0$) is non-negative, and she never incurs a loss. In addition, a truthful user who does not win is not incurring a loss since she obtains 0 utility. This proves the individual-rationality of MOSAP-X mechanisms. □

We now prove the following lemmas and use them to prove that MOSAP-X mechanisms are incentive-compatible. In order to prove that the mechanisms are incentive-compatible, we need to show that the scheduling algorithms are monotone, and the payment functions are based on the critical payment.

Lemma 1. *Let Γ_i be the space of possible requests user i may report to the MOSAP-X mechanisms. The scheduling algorithm MOSAP-X-SCH is monotone, for each $\hat{\beta}'_i, \hat{\beta}_i \in \Gamma_i$, $\hat{\beta}'_i \succeq \hat{\beta}_i$, if user i wins by $\boldsymbol{\mathcal{S}}(\hat{\beta}_i, \hat{\boldsymbol{\beta}}_{-i})$ then she wins by $\boldsymbol{\mathcal{S}}(\hat{\beta}'_i, \hat{\boldsymbol{\beta}}_{-i})$. In other words, if user i wins by bidding $\hat{\beta}_i$, then she will also win if she reports a more preferable bid $\hat{\beta}'_i$.*

Proof. Request $\hat{\beta}'_i$ is more preferred than $\hat{\beta}_i$ if user i requests less amount of charging units, submits an earlier request, a later deadline, and a higher value. It is only beneficial for the user to misreport $\hat{a}_i \geq a_i$ and $\hat{d}_i \leq d_i$. These cases of misreports do not represent more preferable bids, and thus, we will focus on misreports of v_i and l_i.

If user i reports $\hat{v}'_i \geq \hat{v}_i$, her priority metric increases in all the MOSAP-X mechanisms. As a result, bid $\hat{\beta}'_i$ will be selected as a winner by the MOSAP-X mechanisms if $\hat{\beta}_i$ is also selected as a winner.

Similarly, if a user is selected as a winner by reporting $\hat{l}_i$, she will also be selected by reporting $\hat{l}'_i \leq \hat{l}_i$. This is due to the fact that her priority metric either increases in case of MOSAP-II-SCH or remains the same in the cases of MOSAP-I-SCH and MOSAP-III-SCH. □

Lemma 2. *The payment function implemented by MOSAP-X-PAY is based on the critical payment.*

Proof. We need to show that Π_i^t determined by MOSAP-X-PAY is the minimum value that user i must report to get complete charging service. User i's payment is $\Pi_i = f_q$ for MOSAP-I-PAY and MOSAP-III-PAY, and $\Pi_i = f_q \hat{l}_i$ for MOSAP-II-PAY (line 13), where q is the index of user q appearing after user i based on the non-increasing order of the priority metrics (line 3), and she would have won if user i did not participate. We consider that user i submits a lower value $\hat{v}'_i < \Pi_i^t$. User i's new priority metrics are decreased. We show the following cases:
MOSAP-I-PAY: $f'_i = \hat{v}'_i < \Pi_i^t$.
MOSAP-II-PAY: $f'_i = \frac{\hat{v}'_i}{\hat{l}_i} < \frac{\Pi_i^t}{\hat{l}_i}$. Since $\Pi_i^t = f_q \hat{l}_i$, we have $f'_i < \frac{f_q \cdot \hat{l}_i}{\hat{l}_i}$.
MOSAP-III-PAY: $f'_i = \hat{v}'_i < \Pi_i^t$.
Thus, we have $f'_i < f_q$, that is, user i will appear after user q, who did not win. As a result, if user i reports a bid below the minimum value (i.e., Π_i^t), she loses; otherwise she wins. This unique value is the critical payment for user i. This, together with the fact that losing users pay zero, show that the payment function implemented by MOSAP-X-PAY is the critical payment. □

Theorem 7. *MOSAP-X mechanisms are incentive-compatible.*

Proof. Lemma 1 shows that the MOSAP-X-SCH is monotone. Lemma 2 shows that the MOSAP-X-PAY implements the critical payment. It follows from [101] that MOSAP-X are incentive-compatible. □

We show that our proposed mechanisms are robust against manipulation by users through the following example. To analyze the effect of untruthful reporting on the utility

	β_i	$\hat{a}_i$	$\hat{l}_i$	$\hat{d}_i$	$\hat{v}_i$
EV_1	β_1	0	3	6	5
EV_2	β_2	1	3	6	6
EV_3	β_3	2	2	4	4

Table 3.3: Users' true requests

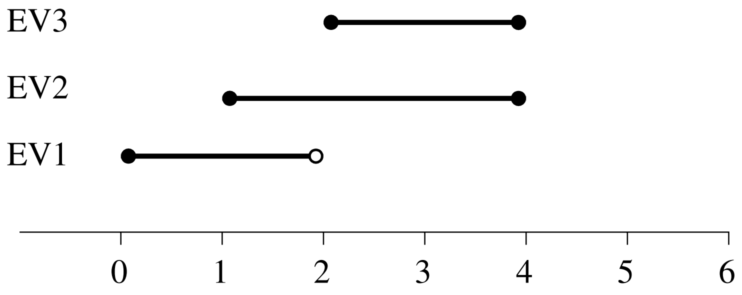

Figure 3.4: Example MOSAP-II-SCH

Table 3.4: Different scenarios for user EV_3's request declaration

Case	$\hat{\beta}_3$	Scenario	Status	Payment	Utility
I	$< 2, 2, 4, 4 >$	$\hat{v}_3 = v_3$	W	3.3	0.7
II	$< 2, 2, 4, 5 >$	$\hat{v}_3 > v_3$	W	3.3	0.7
III	$< 2, 2, 4, 3.5 >$	$\hat{v}_3 < v_3$	W	3.3	0.7
IV	$< 2, 2, 4, 3 >$	$\hat{v}_3 < v_3$	L	0	0
V	$< 2, 3, 4, 4 >$	$\hat{l}_3 > l_3$	L	0	0
VI	$< 2, 1, 4, 4 >$	$\hat{l}_3 < l_3$	W	3.3	0.7
VII	$< 3, 2, 4, 4 >$	$\hat{a}_3 > a_3$	L	0	0
VIII	$< 2, 2, 3, 4 >$	$\hat{d}_3 < d_3$	L	0	0

of the users participating in the MOSAP-II mechanism, we consider three users EV_1, EV_2 and EV_3, whose true requests are shown in Table 3.3. We consider the electricity capacity of $C = 2$ units. MOSAP-II-SCH schedules these users as shown in Fig 3.4, where all users declare their true requests. User EV_2 and EV_3 are selected as winners, and the payments of the winning users based on MOSAP-II-PAY are 5 and 3.3, respectively.

We assume that user EV_3 reports a different request, $\hat{\beta}_3$, from her true request $\beta_3 =< 2, 2, 4, 4 >$. As shown in Table 3.4, we analyze different scenarios, where user EV_3 submits different requests. In addition, we present the payment and utility of the user for all the cases.

In case I, user EV_3 submits her true request, that is, $\beta_3 = \hat{\beta}_3$. In this case, user EV_3 wins, and receives the requested charging units. The mechanism charges her \$3.3, and her utility is 4-3.3=0.7. In case II, user EV_3 submits a request with a higher bid $\hat{v}_3 = 5$. In this case, user EV_3 is still a winner and the mechanism determines the same payment for her as in case I, leading to a utility of 0.7. In case III, she submits a request with a lower bid $\hat{v}_3 = 3.5$, which is not less than the price determined by our mechanism (i.e., \$3.3). Thus, user EV_3 is still winning, and the mechanism charges her the same amount as in case I. However, if user EV_3 submits a request with a bid below the critical payment, she will not obtain her requested charging units, leading to zero utility. This is shown in case IV, where user EV_3 submits a bid $\hat{v}_3 = 3$. We now investigate scenarios in which user EV_3 requests a different amount of charging units than her true request. In case V, she requests more amount of charging $\hat{l}_3 = 3$ instead of 2 units in the case of her true request, case I. In this case, user EV_3 is not selected, leading to zero utility. In case VI, the user requests less charging units. In this case, user EV_3 is still a winner and the mechanism determines the same payment for her as in case I. This is due to the fact that the user declared a more preferable request than her actual request. The user does not gain more utility by such declarations. In case VII, the user submits her request with a later arrival, which makes the allocation unfeasible. In case VIII, user EV_3 submits her request with a sooner deadline, which makes the allocation unfeasible leading to zero utility for the user. We showed that if a user submits a request untruthfully, she can not increase her utility.

3.6 Experimental Results

We perform extensive experiments in order to investigate the properties of the proposed mechanisms, MOSAP-X. We compare the performance of MOSAP-X with that of VCG-SAP and FIXED, where VCG-SAP solves optimally the offline version of the problem, and FIXED is a fixed-price mechanism. In the FIXED mechanism, each unit of charging is allocated to a user chosen randomly. If a user receives her total requested units in the FIXED mechanism, she pays the reserve price. We rely on the VCG-SAP and FIXED results as benchmarks for our experiments. All algorithms are implemented in C++. SAP-IP is implemented using APIs provided by IBM ILOG CPLEX Optimization Studio Multiplatform Multilingual eAssembly. In this section, we describe the experimental setup and analyze the experimental results.

3.6.1 Experimental Setup

Following [113], we consider a general synthetic setting, in which we generate users and their requests from simple distributions. The main reason for this setup is to generate results that are easily reproducible. For each user i, we sample the EV arrival time a_i from the discrete uniform distribution on $\{0, 1, 2, \ldots, 23\}$ and the EV departure time from $\{a_i, a_i + 1, \ldots, 23\}$. We sample the number of required units l_i uniformly at random from $\{1, 2, \ldots, 5\}$. Finally, we generate v_i from an exponential distribution with rate 1. In addition, we consider 0.5 as the reserve price.

3.6.2 Analysis of Results

We analyze three sets of experiments: small-scale, large-scale, and sensitivity analysis on capacity. We compare the performance of MOSAP-X, VCG-SAP, and FIXED for different number of users and amount of capacity. We record the welfare, the revenue, the execution time, the total served users, and the total allocated units with payment for each mechanism. In the small-scale experiments, we consider that the available capacity is one unit, while in the large-scale experiments, we consider that the available capacity is 50 units. The reason that we analyzed our mechanisms in these two cases is due to the intractability of VCG-SAP. VCG-SAP cannot find the optimal solution in feasible time for all instances of the SAP problem. Therefore, we present the results of the small-scale experiments, where VCG-SAP is able to find the optimal solution in reasonable amount of time for all the instances. In addition, we analyze the effect of change in available charging capacity on both mechanisms by performing sensitivity analysis on the capacity.

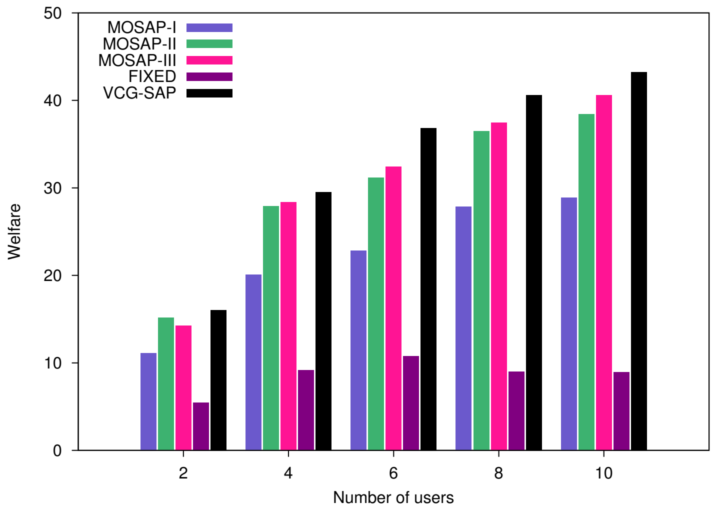

Figure 3.5: Small-scale experiments with 1 unit capacity: Welfare

Small-scale experiments

We analyze the performance of MOSAP-X, VCG-SAP, and FIXED, where the available capacity is 1 unit. In this case, the number of users that arrive every hour is between 2 and 10. Fig. 3.5 shows the welfare obtained by the mechanisms. These results show that MOSAP-II and MOSAP-III obtain a welfare very close to that obtained by the optimal VCG-SAP mechanism. Such results are very promising given the fact that MOSAP-X is an online mechanism which does not have any information about future demand. However, VCG-SAP is an offline mechanism and has all the information available a priori. However, the welfare obtained by MOSAP-I is not close to the optimal results because it does not consider the amount of requested charging units by users in its scheduling function. As expected, since FIXED randomly allocates the unit to users, its obtained welfare is far from the optimal results.

Fig. 3.6 shows the revenue achieved by the provider when using the mechanisms. Note that the VCG-SAP is optimal in terms of welfare and not the revenue. The results show that MOSAP-II obtains the highest revenue among all the mechanisms.

Fig. 3.7 shows the execution times of the mechanisms on a logarithmic scale. As we expected, the execution time of MOSAP-X and FIXED are very small. This is due to the fact that the time complexity of MOSAP-X and FIXED is polynomial in the size of input.

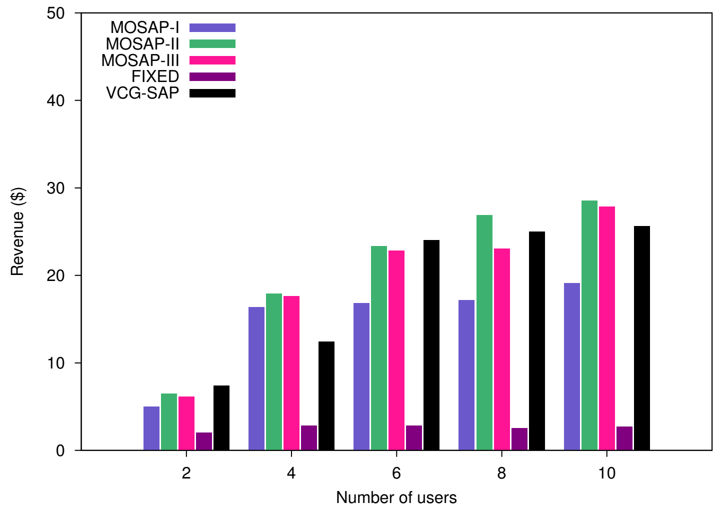

Figure 3.6: Small-scale experiments with 1 unit capacity: Revenue

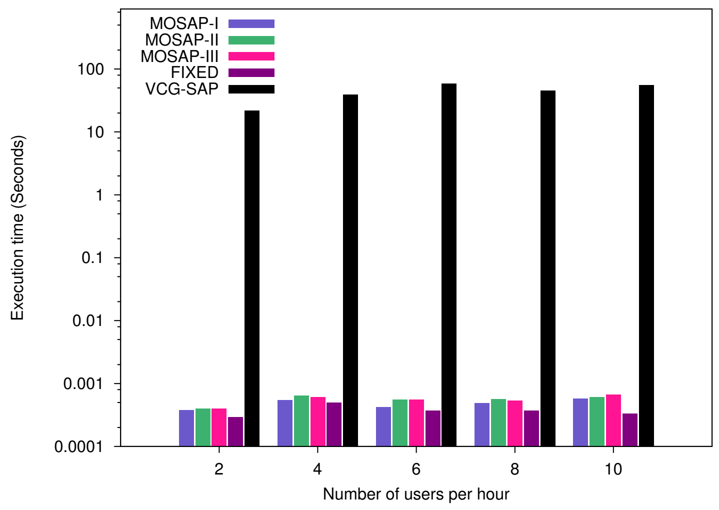

Figure 3.7: Small-scale experiments with 1 unit capacity: Execution time

The results show that MOSAP-X is suitable for providing charging services in realtime. Note that small execution time of online charging mechanisms is a must have property in such settings. However, the execution time of VCG-SAP, is more than five orders of magnitude greater than that of MOSAP-X.

Fig. 3.8 shows the average number of served users for the mechanisms. These users are the ones who have their requested charging units fully scheduled. MOSAP-II, MOSAP-III, and VCG-SAP serve more users than MOSAP-I and FIXED. This is due to the fact that

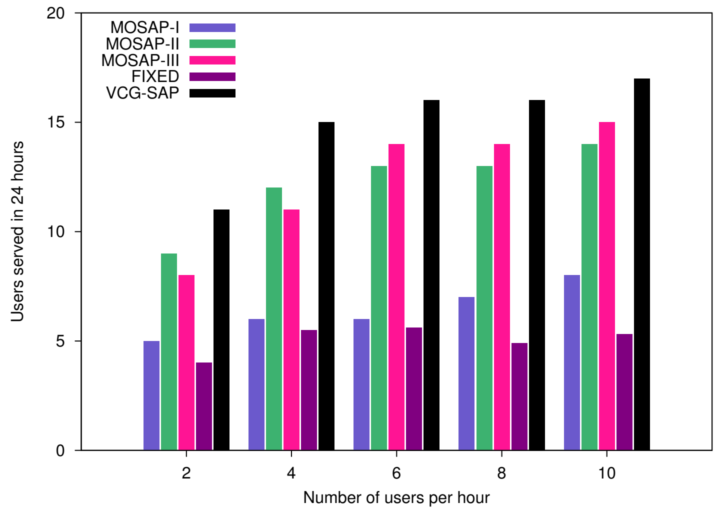

Figure 3.8: Small-scale experiments with 1 unit capacity: Total served users

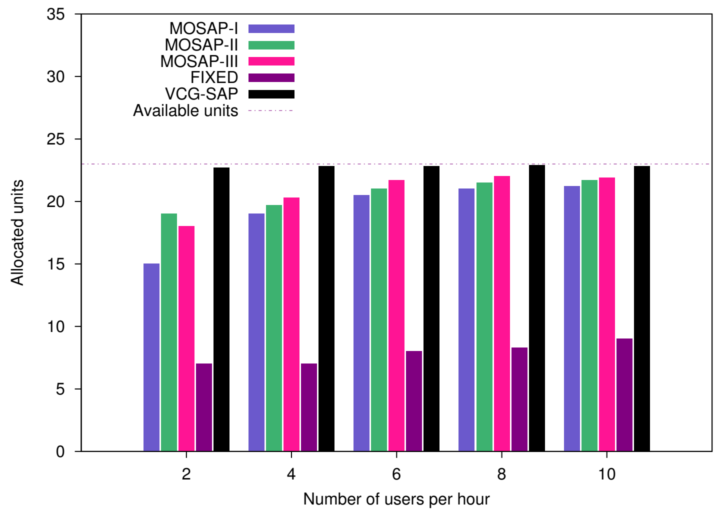

Figure 3.9: Small-scale experiments with 1 unit capacity: Total allocated units

the solution determined by MOSAP-II and MOSAP-III are closer to the optimal solution (as it is shown in Fig. 3.5). Note that the requested amount of charging by a user can be more than 1 unit.

Fig. 3.9 shows total allocated units with payment obtained by the mechanisms. The results show that VCG-SAP allocates almost all the available units during the 24 hours to users who receive their entire requests. MOSAP-X is also capable of allocating the entire requests of users close to that of optimal solution. The remaining units are allocated to some users who do not receive their entire requests due to preemption. However, the results obtained by FIXED are far from that of optimal despite the fact that all the units are allocated to users while these users are not necessarily receiving their entire requests.

Large-scale experiments

We analyze the performance of MOSAP-X, VCG-SAP, and FIXED, where the available capacity is 50 units. In this case, the number of users that arrive every hour is between 50 and 250. For the instance of the problem with more than 100 users in every hour, VCG-SAP was not able to find the optimal solution even after one hour which is the entire time interval. This is due to fact that the execution time of VCG-SAP becomes prohibitive for large instances of the problem. Note that in this online setting, the mechanisms are expected to respond in realtime. As a result, we did not capture the solutions obtained after one hour of execution of the mechanisms.

Fig. 3.10 shows the welfare obtained by the mechanisms. The results show that MOSAP-II and MOSAP-III obtain a welfare very close to the optimal (obtained by VCG-SAP) in cases with 50 and 100 users. For the remaining cases, MOSAP-II and MOSAP-III obtain the highest welfare among all the mechanisms. Similar to the welfare obtained by the mechanisms in the small-scale experiments presented in Fig 3.5, MOSAP-II and MOSAP-III obtain higher welfare than those obtained by MOSAP-I and FIXED mechanisms. As in the case of the small-scale experiments, the welfare obtained by MOSAP-I is not close to the optimal results because it does not consider the amount of requested charging units by users in its scheduling function. FIXED also obtains welfare far from the other mechanisms. Fig. 3.11 shows the revenue obtained by the provider when using the mechanisms. Note that the VCG-SAP is optimal in terms of welfare and not the revenue. The results show that MOSAP-II obtains the highest revenue among all the mechanisms in most cases. These results are in agreement with the results presented in Fig 3.6.

Fig. 3.12 shows the execution times of the mechanisms on a logarithmic scale. The execution time of MOSAP-X and FIXED are very small, in the order of milliseconds. However, the execution time of VCG-SAP, is more than five orders of magnitude greater

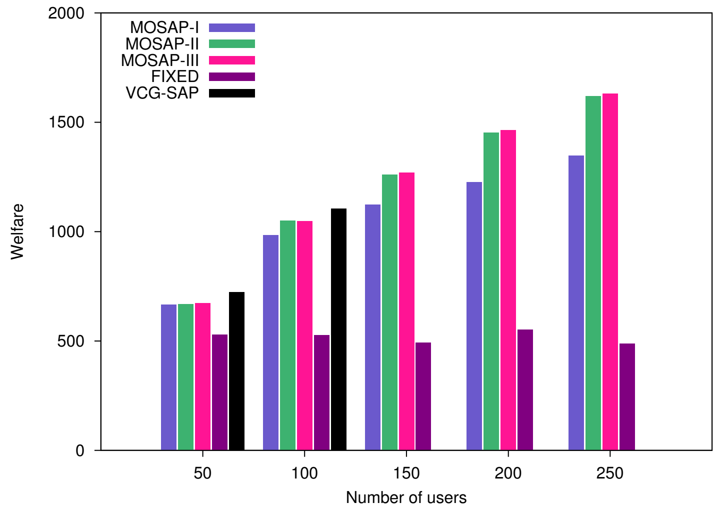

Figure 3.10: Large-scale experiments: Welfare

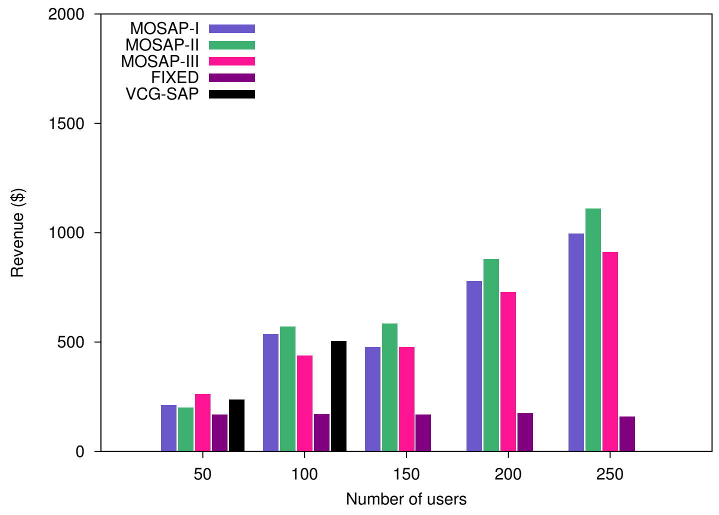

Figure 3.11: Large-scale experiments: Revenue

than that of MOSAP-X in the first two cases. A comparison of the execution time of VCG-SAP between small-scale and large-scale experiments shows that the execution time of VCG-SAP grows exponentially when the available charging capacity and the number of users increase. The fact that the execution time of MOSAP-X even for large-scale experiments is in terms of milliseconds make it suitable to be incorporated in online charging settings. Fig. 3.13 shows the average number of served users whose entire requests are scheduled by the mechanisms. MOSAP-II, MOSAP-III, and VCG-SAP serve more users

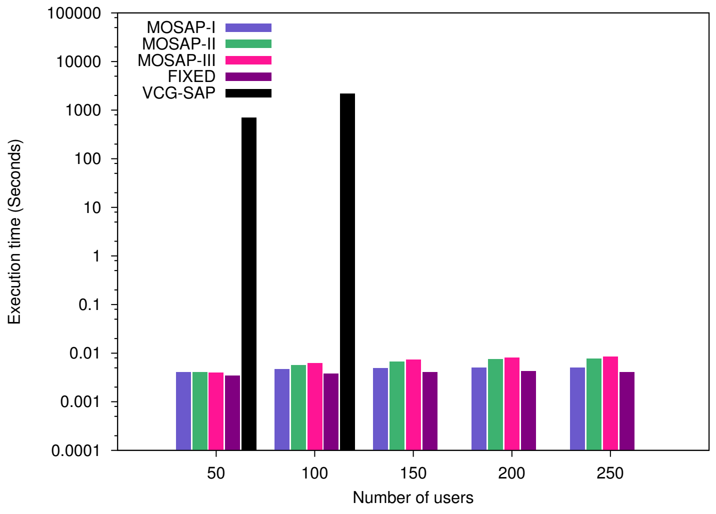

Figure 3.12: Large-scale experiments: Execution time

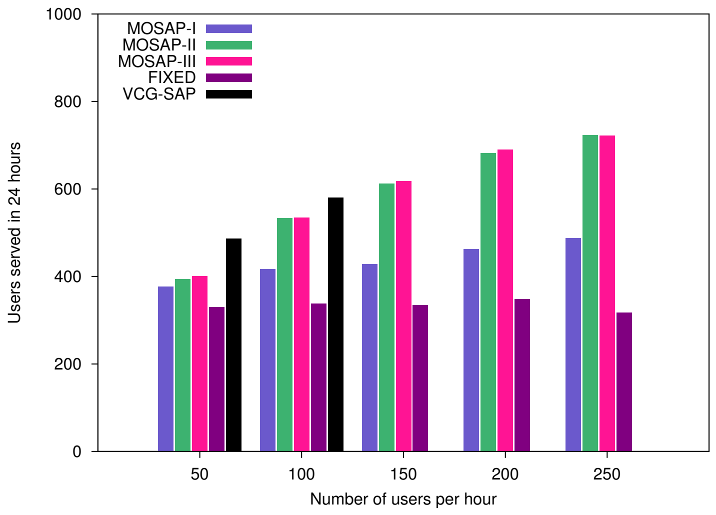

Figure 3.13: Large-scale experiments: Total served users

than MOSAP-I and FIXED. MOSAP-I selects the users only based on their values with no consideration for the requested amount of charging units. This prevents MOSAP-I to serve a higher number of users than MOSAP-II and MOSAP-III, given the limited amount of charging capacity.

Fig. 3.14 shows total allocated units with payment obtained by the mechanisms. These results show that VCG-SAP allocates almost all the available units during the 24 hours to users who receive their entire requests. MOSAP-X is also capable of allocating the entire

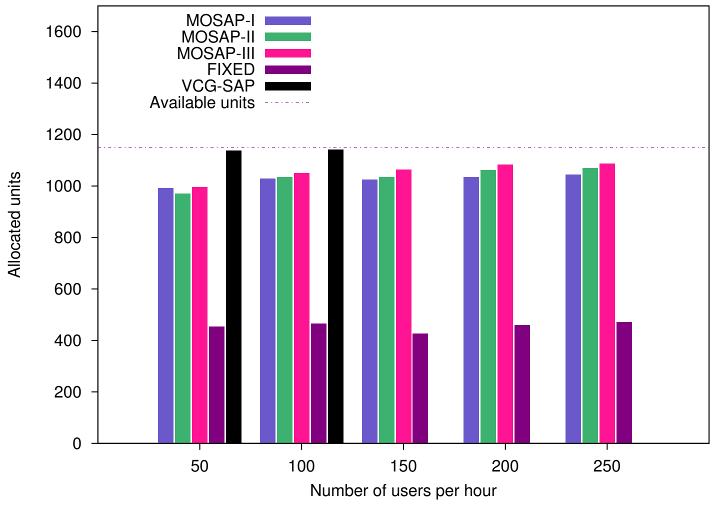

Figure 3.14: Large-scale experiments: Total allocated units with payment

requests of users close to that of optimal solution. The remaining units are allocated to some users who do not receive their entire requests due to preemption. However, the results obtained by FIXED are far from the optimal despite the fact that all the units are allocated to users while these users are not necessarily receiving their entire requests.

From the results of these experiments we can conclude that MOSAP-II obtains on average higher revenue than the other mechanisms, while at the same time finds solutions close to the optimal solutions obtained by VCG-SAP. MOSAP-X finds the charging schedule and payment of users much faster than VCG-SAP. From the results of these experiments we can conclude that MOSAP-X is very suitable for utility providers, since it allows them to make decisions in real-time.

Sensitivity analysis on capacity

To show the effects of change in capacity on the performance of MOSAP-X, we perform sensitivity analysis with respect to capacity. For this set of experiments, the number of users that arrive every hour is 100, while the capacity per hour is varied between 20 and 80 units. In this setting, VCG-SAP could not find the optimal solution in one hour when the capacity is 80. As a result, there is no bar for VCG-SAP in Fig. 3.15-Fig. 3.19 for the case of 80 units.

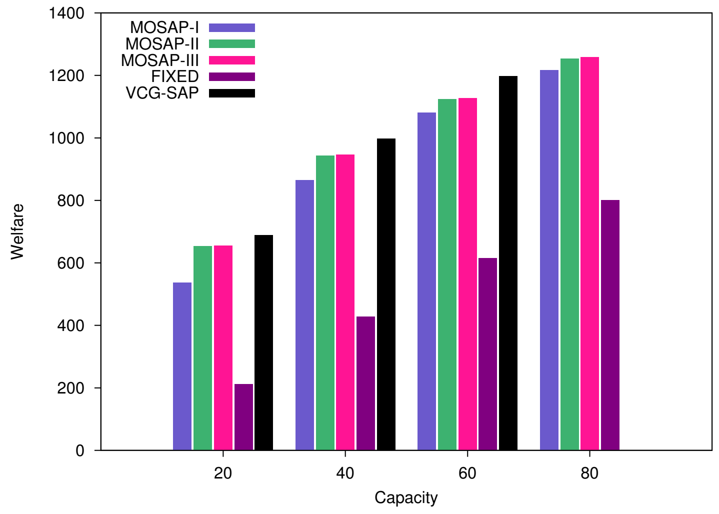

Figure 3.15: Sensitivity analysis of available capacity: Welfare

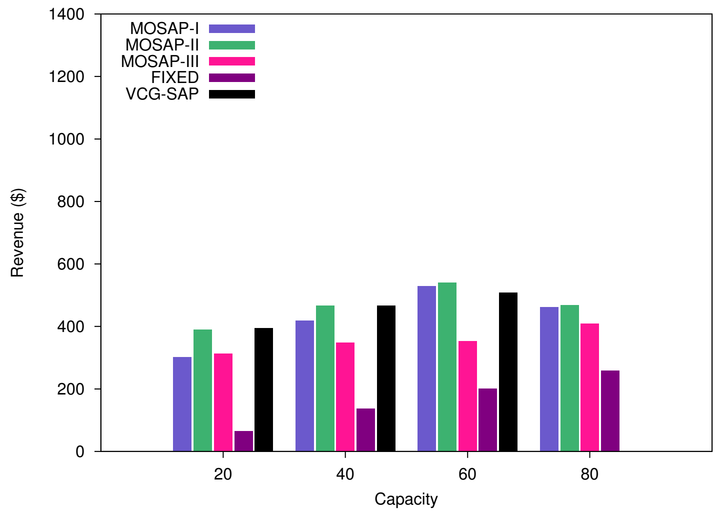

Figure 3.16: Sensitivity analysis of available capacity: Revenue

Fig. 3.15 shows the welfare obtained by the mechanisms. The results show that MOSAP-II and MOSAP-III obtain a welfare very close to optimal (obtained by VCG-SAP). By increasing the capacity, the obtained welfare by all the mechanisms increases since more users can be served.

Fig. 3.16 shows the revenue obtained by the provider, where MOSAP-II obtains the highest revenue among all the mechanisms. By increasing the capacity from 20 to 60, the revenue obtained by the provider increases for all the mechanisms. However, when the

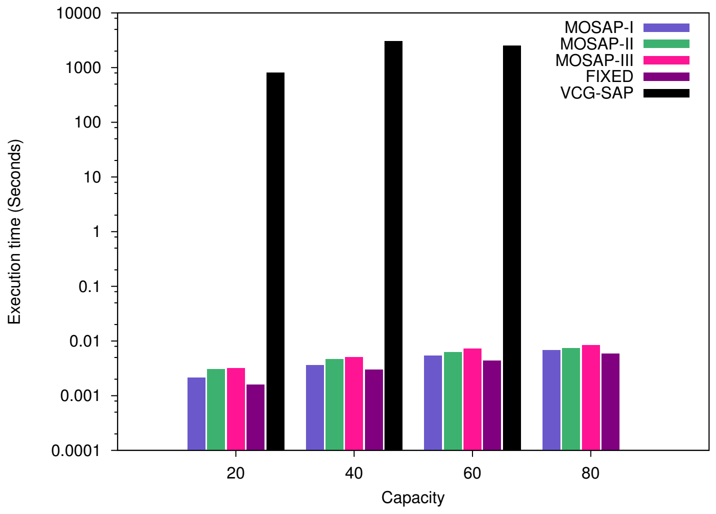

Figure 3.17: Sensitivity analysis of available capacity: Execution time

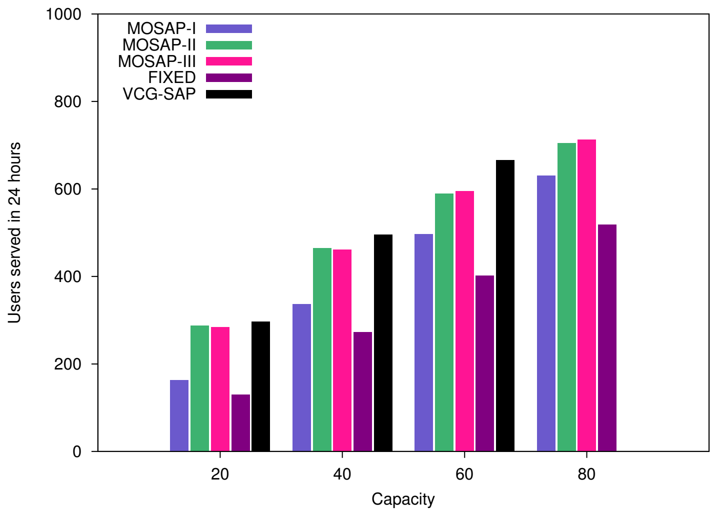

Figure 3.18: Sensitivity analysis of available capacity: Total served users

capacity is 80 units, we do not observe such increase in the revenue. This is due to the fact that when the supply is high, and the mechanisms may be able to fulfill more requests, the price of charging units can decrease leading to a lower revenue.

Fig. 3.17 shows the execution times of the mechanisms on a logarithmic scale. The execution time of MOSAP-X and FIXED are very small. The execution time of VCG-SAP does not necessarily increase with the increase in capacity since finding optimal solutions for the problem instances with lower capacity may need more time. Fig. 3.18 shows the

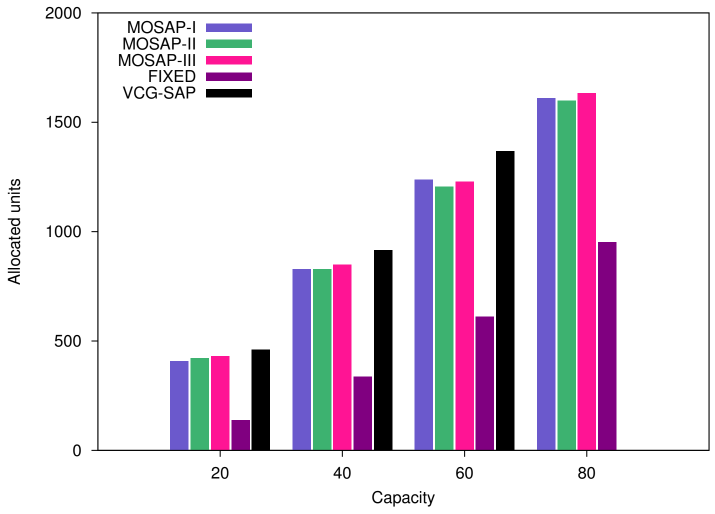

Figure 3.19: Sensitivity analysis of available capacity: Total allocated units with payment

average number of served users whose entire requests are scheduled by the mechanisms. The results show that the number of served users increases by all the mechanisms with the increase in capacity. Fig. 3.19 shows total allocated units with payment obtained by the mechanisms. These results show that MOSAP-X is capable of allocating the entire requests of users close to that of optimal solution.

In real world settings, both the capacity of the utility provider and the arrival rate of charging requests can vary over time. We design our experiments to analyze both of these scenarios. In the small-scale and large-scale experiments, the number of requests changes while we choose a fixed amount of capacity. We also perform a sensitivity analysis on capacity while the request arrival rate is fixed. From all experiments, we conclude that MOSAP-X is capable of providing online scheduling and pricing services in real world settings. These results show that, MOSAP-X also provides these services obtaining high revenue, close to optimal welfare, small execution time, while at the same time, users do not need to strategize to interact with the mechanism.

3.7 Conclusion

The dynamics of charging requests and the fact that utility providers need to consider load balancing necessitates designing preemption-aware online mechanisms for EV charging. In this study, we proposed a framework for EV charging considering the incentives of both utility providers and EV drivers. Our proposed framework brings about a win-win situation in which EV drivers can receive their charging requests at lower prices, and utility providers can sell their unused capacity while considering their load balancing objectives. We introduced the problem of online scheduling and pricing for EV charging, and designed a family of online mechanisms, MOSAP-X. We proved that our proposed mechanisms are incentive-compatible, where truthful reporting is a dominant strategy for users. We performed extensive experiments that showed that the proposed mechanisms are not only capable of finding close to optimal solutions, but are also very fast and obtain high revenue. The promising results make MOSAP-X suitable for scheduling and pricing EV charging in real-time.

CHAPTER 4: HIERARCHICAL TIME-DEPENDENT SHORTEST PATH ALGORITHMS FOR VEHICLE ROUTING UNDER ITS

4.1 Introduction

The quickly expanding Intelligent Transportation Systems (ITS) coverage around the world can be a key enabler for efficient vehicle route planning and for reducing the effects of traffic congestion on travel times. ITS provides valuable information for a time-dependent road network, such as time-varying travel times for traversing road segments at high resolution (e.g., one minute). Routing algorithms must exploit these traffic information feeds efficiently, both to plan the route in advance and to update it en route. In general, an efficient routing algorithm should strike a balance among preprocessing time, query time, optimality gap, and storage/processor memory requirements. In addition, the scalability of the routing algorithm for handling large-scale road networks while maintaining reasonable response times is an important property. Depending on the form of implementation of the routing application, however, some of the aforementioned features may be prioritized over others. In this study, we focus on large-scale deterministic time-dependent transportation networks. The need for fast responses to ITS information puts the speed-up techniques for shortest path problems (SPP) on time-dependent networks at the heart of computational needs for routing. In addition, a vast majority of vehicle routing navigation systems, whether built-in or portable, lack the ability to rely on online servers and must compute the route in a stand-alone mode with limited hardware processing/memory capacity. This last aspect is the primary focus of this study to design computationally efficient yet effective hierarchical search strategies and algorithms to solve the time-dependent shortest path problem (TDSP).

Definition 11 (Time-Dependent Shortest Path (TDSP)). *Given a time-dependent network, an origin O, a destination D, and a start time, the* time-dependent shortest path *is a path with the minimum travel time among all paths from O to D starting at the specified starting time.*

The TDSP problem is an adaptation of SPP to time-dependent networks. [20] first studied the TDSP problem using dynamic programming. [26] studied the generalization of Dijkstra's algorithm for determining TDSP with the same time complexity as the SPP problem. [1] and [63] proved that the TDSP problem is polynomially solvable. See [34] for a recent study on the complexity of the TDSP problem.

Dynamic programming methods are prevalent in the literature for the TDSP problem. Such methods suffer from the curse of dimensionality in dealing with the scale and complexity of transportation networks. They require overly long query times for computing the route and for offering rerouting options once the vehicle is en route. On the other hand, naïve algorithms that arbitrarily limit the degree of ITS "look ahead" to a small neighborhood ahead of the vehicle to reduce the state space can lead to a higher optimality gap.

An approach to speeding up the computation of shortest paths is pre-computing the optimal paths, short-cuts, or lower bounds for all OD pairs or a subset at different time windows [10, 125]. Methods based on ALT (A*, Landmarks, Triangle inequality) employ landmarks to find lower bounds in order to direct the search in a reduced search space [46, 47, 48]. Bidirectional ALT further reduces the search space by adding a backward search from the destination to reduce the search space that has to be explored by the forward search [87, 48]. In ALT-based methods, there is a tradeoff between choosing well-positioned landmarks and preprocessing time. These methods, however, require large memory space, rendering them ineffective for large road networks as well as for vehicles not relying on online routing services.

There are extensive studies on designing routing algorithms for stochastic networks, which each road segment has stochastic traversal times. There are two versions of the shortest path problem on stochastic networks, the expected shortest path problem [37], where all information on the arc weights is available before starting the trip; and the shortest path with recourse problem (SPR) [108, 133], where only local traffic information is available. SPR is more realistic in routing applications since in reality all information on traffic network dynamics is not available. While it is desirable to consider the stochastic nature of the traffic networks, solving stochastic routing problems is generally complex and prohibitive for real-time routing on large-scale road networks. Hence, we focus on large-scale deterministic and time-dependent transportation networks.

In this study, we propose an algorithm capable of solving TDSP in milliseconds on large-scale dynamic road networks without the need for storing memory-intensive precomputed paths, short-cuts, or bounds. In particular, we propose new search strategies that exploit the hierarchical structure of efficient road network representations.

Hierarchical approaches have been used in routing algorithms for large road networks, and have proven to be effective on both static networks [31, 59, 61, 35, 4, 111, 55, 5, 125] and dynamic networks [16, 121, 12, 40, 22]. A hierarchical search can dramatically reduce the search space. This is due to the fact that the search will take place predominantly at higher levels of network representations that tend to be sparse, with far fewer nodes and arcs. These methods mostly employ hierarchical representations based on the fixed topology and functional classification of road networks. Functional classification categorizes streets and highways into classes based on the character of service they are intended to provide. The classification is rooted in the road network design and helps determine the speed category and travel time of passing through the road under free-flow conditions. One issue inherited with a majority of hierarchical routing algorithms in the literature is enforcing the vehicle to travel over higher-level arcs (e.g., highways) without considering the traffic state of those arcs. Although the speed limit is higher at higher levels, and the optimal route might pass through higher levels under free-flow conditions, this route may not necessarily be optimal under different traffic conditions. Therefore, incorporating just the fixed topology of road networks and its functional classes may not be adequate for efficient hierarchical routing.

Instead of a functional class representation, we employ an emerging concept in analyzing complex networks called "community structure detection" [18, 97] to form hierarchical community-based representations of road networks efficiently [96, 11]. We present a model of the hierarchical representation to aid the computational performance of our proposed algorithm for TDSP. While it has been shown that the community detection methods are effective for path-finding in static networks [125], there are no studies for time-dependent networks. Our proposed algorithm for solving TDSP employs new hierarchical search strategies to reduce the state space without compromising optimality gap.

4.1.1 Our Contribution

We propose a hierarchical time-dependent shortest path algorithm (HTNGD) to solve the deterministic TDSP problem on large-scale networks. HTNGD uses community-based hierarchical representations of road networks, and it recursively reduces the search space in each level of the hierarchy by using our proposed search strategy algorithm, TNGD. We perform extensive experiments in order to investigate the performance of HTNGD. We use time-dependent A* (TA*) as a benchmark when we investigate the performance of HTNGD, and we compare HTNGD with the most successful speedup techniques in the literature. The results show that the overhead memory requirement and the pre-processing time of HTNGD are the lowest, and its query time is in terms milliseconds. These properties make HTNGD suitable for deployment in vehicle routing navigation systems that do not rely on online servers.

4.1.2 Organization

The rest of the study is organized as follows. We explain hierarchical community-based representation of road networks in section 4.2. Section 4.3 describes the proposed algorithms for solving TDSP. Section 4.4 presents experimental results from applying the proposed algorithm on Detroit, New York, and San Francisco road networks. Finally, section 4.5 offers some concluding remarks and directions for future research.

4.2 Hierarchical Representation of Road Networks

Complex networks have attracted a great deal of attention across many fields of science [51, 100, 99]. A recently proposed concept in analyzing complex networks is their "community structure" [98, 18]. Many networks can be decomposed into communities such that the densely connected subsets of nodes form communities with only sparser connections between them. A wide variety of methods have been lately developed for detecting communities in networks (see [33] for a recent review).

Road networks are commonly represented by directed graphs where streets form the arcs, and intersections are considered as nodes. To capture the dynamics of road networks, arc traverse times can be considered as arc "weight." Community detection methods can be employed to decompose the weighted road network to effectively represent the network structure and its connectivity [95]. Hierarchical search strategies can exploit this community structure for solving the TDSP problem.

There are two approaches to build hierarchical representations of networks in the literature [33]: agglomerative and divisive. In agglomerative, a bottom-up approach, the detected communities in a network become an input to another iteration of community detection method [105]. In divisive, a top-down approach, all nodes are considered as one community, then it splits into communities in lower levels of the hierarchy [110]. In both approaches, each hierarchy forms a directed graph itself with fewer arcs and nodes as we go up the levels. These higher levels are abstractions of their lower-level graphs.

To model each level of the hierarchy, we consider the graph in level h as $G^h(V^h, A^h, W^h)$ where V^h is a set of nodes, A^h is a set of arcs, and W^h is a set of arc weights. Suppose that G^h is partitioned into k^h communities $C_i^h(V_i^h, A_i^h, W_i^h)$, where $i = 1, \ldots, k^h$ with the following properties:

$$\begin{cases} \bigcup_{i=1}^{k^h} V_i^h = V^h, \\ \bigcup_{i=1}^{k^h} A_i^h \subseteq A^h \end{cases} \tag{4.1}$$

where $\forall p, q, V_p^h \cap V_q^h = \emptyset$, $A_p^h \cap A_q^h = \emptyset, 1 \leq p, q \leq k^h$, and $p \neq q$. In the rest of the study,

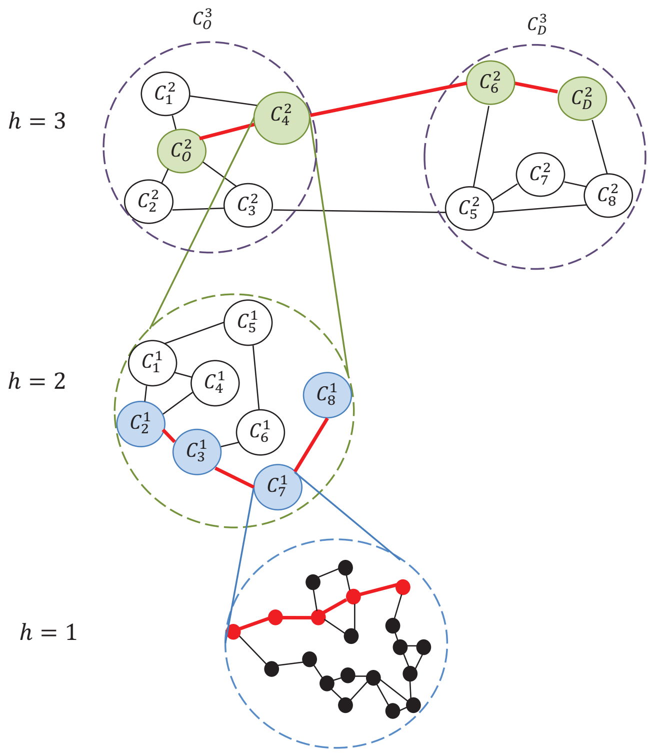

Figure 4.1: Illustrative example for the hierarchical representation of a network

we refer to community $C_i^h(V_i^h, A_i^h, W_i^h)$ as C_i^h.

In each community C_i^h, a subset of A^h, A_i^h, connects its nodes, V_i^h such that A_i^h represents intra-community arcs. In addition to these arcs, $A^h \,/ \bigcup_{i=1}^{k^h} A_i^h$ is a subset of arcs representing the intercommunity arcs, which connect pairs of communities in level h. For each arc in $A^h \,/ \bigcup_{i=1}^{k^h} A_i^h$ that connects two communities C_p^h and C_q^h, we define $w^h_{C_p^h C_q^h}$ as the travel time between centers of those communities. We set a virtual vertex as the center of a community. In the case of road networks, the coordinates of the center is the average of coordinates of all vertices within the projection of that community to the lowest level. Therefore, we set travel time $w^h_{C_p^h C_q^h}$ as the distance between the virtual vertices divided by the maximum speed limit. Note that the projection of each community C_i^h to the lowest level covers a subset of nodes in G^1.

Each community in level $h-1$ is represented by a node in level h. That means each community C_p^{h-1}, $1 \leq p \leq k^{h-1}$, is represented by a node $v \in V^h$. If v is a vertex $(v \in V_q^h)$ that belongs to C_q^h, $1 \leq q \leq k^h$, then C_q^h is a super-community of C_p^{h-1} and C_p^{h-1} is a sub-community of C_q^h. In each level, a node represents a sub-community. In general, $\bigcup_{i=1}^{k^{h-1}} C_i^{h-1} = V^h$. Thus, there is a one-to-one correspondence between V^{h+1} and C^h, where $C^h = \bigcup_{i=1}^{k^h} C_i^h$.

In all levels of the hierarchy, V^h is the set of communities of level $h-1$, where $h \neq 1$. If $h = 1$, G^1 represents the actual road network, where V^1, A^1 and W^1 represent sets of road intersections as communities, road segments, and road segment travel times, respectively. In our proposed time-dependent model of the road network, we denote w_{ij}^t as the travel time of the arc $(i, j) \in A^1$ connecting $i \in V^1$ to $j \in V^1$, where t is the arrival time at node i.

Fig. 4.1 shows a highly stylized example to illustrate the hierarchical representation of an undirected and an unweighted network with three levels of hierarchy. The graph in level 3 (i.e., G^3) consists of 10 nodes that are partitioned into two communities C_O^3 and C_D^3. Each node in this level is a community in level 2. For example, C_4^2 is represented as a node in level $h = 3$ which along with four other nodes forms community C_O^3 in level 3. Therefore, community C_O^3 is its super-community. In addition, (C_O^2, C_4^2) is an intra-community arc within community C_O^3, and (C_4^2, C_6^2) is intercommunity arc that connects two communities C_O^3 and C_D^3. Community C_4^2 in level 2 consists of 8 nodes (sub-communities), $C_1^1, \ldots, C_8^1$. For example, C_1^1 is a sub-community of C_4^2. We show the projection of community C_7^1 in level 1, which is a part of the actual graph G^1.

A modularity measure was first introduced by [98] to measure the strength of partition of a network into communities. This measure gives a value, ψ, between -1 and 1 for a partition based on the density of arcs inside communities in comparison with the density of arcs between communities. A higher value of ψ indicates a better partitioning of the network. ψ is a property of a network and a specific partition of the network into communities. For simplicity, we assume nodes i and j belong to communities C_i and C_j, respectively. In the case of weighted directed networks, the modularity measure for all arc (i, j) and a given partition is defined as follows:

$$\psi = \frac{1}{m} \sum_{(i,j)} \left[b_{ij} - \frac{d_i^{in} d_j^{out}}{m} \right] \delta(C_i, C_j) \tag{4.2}$$

$$\delta(C_i, C_j) = \begin{cases} 1 & \text{if } C_i = C_j \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

$$m = \sum_{(i,j)} b_{ij} \tag{4.4}$$

where b_{ij} represents the closeness weight of the arc between i and j, and d_i^{in} (d_j^{out}) is the sum of the incoming (outgoing) arc closeness weights attached to vertex i (j). It is worth mentioning that b_{ij} indicates closeness or similarity between nodes i and j that can give useful information about communities. Not all weights on network arcs are necessarily appropriate for determining community structure. In traffic networks, the inverse of travel time between nodes i and j can be used as the value for b_{ij} in order to find densely connected subsets of nodes as communities. For example, if the travel time between two nodes is long, it does not mean that these nodes are similar so they may be assigned to different communities.

We employ the Louvain method [11], which is an agglomerative approach for constructing hierarchical representation of the network. This method not only extracts a hierarchical community structure, but exhibits excellent computational performance even for large-scale directed networks. The Louvain method is a heuristic method based on the gain in modularity, $\Delta\psi_i$, by adding (removing) a vertex i into (from) a community C in each iteration of their proposed method. The gain in modularity, $\Delta\psi_i$, for directed and weighted networks is defined as follows:

$$\Delta\psi_i = \left[\frac{\sum_{j,\ k\in C} b_{jk} + \sum_{j\in C} b_{ij} + \sum_{j\in C} b_{ji}}{m} - \left(\frac{\sum_{j\in C,\ k\notin C} b_{jk} + d_i^{out}}{m}\right)\left(\frac{\sum_{j\in C,\ k\notin C} b_{kj} + d_i^{in}}{m}\right)\right]$$

$$- \left[\frac{\sum_{j,\ k\in C} b_{jk}}{m} - \frac{(\sum_{j\in C,\ k\notin C} b_{jk})(\sum_{j\in C,\ k\notin C} b_{kj})}{m^2} - \frac{d_i^{in} d_i^{out}}{m^2}\right] \tag{4.5}$$

where $\sum_{j,\ k\in C} b_{jk}$ is the sum of the weights of intra-community arcs of C, $\sum_{j\in C,\ k\notin C} b_{jk}$ is the sum of the weights of the arcs incident to vertices in C, and $\sum_{j\in C} b_{ij}$ is the sum of the weights of the arcs from i to vertices in C. Each vertex i is added to one of its neighboring communities that has the highest modularity gain.

Our proposed algorithm is not limited to any specific community structure detection methods; other community structure detection or graph partitioning methods can be applied. In the next section, we propose our hierarchical search method using the proposed hierarchical graph model.

4.3 Hierarchical Time-Dependent Shortest Paths

We propose a new hierarchical search algorithm for solving the TDSP problem on dynamic road networks with discrete and deterministic time-varying travel time. The algorithm exploits the hierarchical representation of the road network, as outlined in section 4.2.

We first introduce a Time-dependent Neighborhood Goal Directed (TNGD) search algorithm. The task of TNGD is to determine a spectrum of promising communities for exploration in each level of the hierarchy. We then propose a Hierarchical Time-dependent Neighborhood Goal Directed (HTNGD) algorithm that recursively employs TNGD to solve the TDSP problem. HTNGD efficiently searches over the entire hierarchical representation of the road network.

Table 4.1: Notation

t_O	Trip start time
C_O^h	Origin community in level h
C_D^h	Destination community in level h
α	Spectrum control parameter
f_v	Estimated minimum total travel time among all paths passing through community v from C_O^h to C_D^h
g_v	Minimum arrival time from C_O^h to community v starting at time t_O
$e(v, C_D^h, g_v)$	Lower bound estimate on travel time to go from v to C_D^h assuming the arrival time to v is g_v
S	Set of visited communities
N	Set of nominated communities for the selection of the next community
CS_{OD}^h	Core set in level h
Q^h	Spectrum of communities in level h

4.3.1 Time-dependent Neighborhood Goal Directed (TNGD) Search Algorithm

We consider a graph $G^h(V^h,\ A^h, W^h)$ as described in section 4.2 to find a spectrum of communities between C_O^h and C_D^h in level h, where C_O^h and C_D^h are the communities containing O and D, respectively. We define a spectrum Q^h as follows.

Definition 12 (Spectrum)**.** *A spectrum Q^h is a set of communities in level h such that the projection of that spectrum to the lowest level of the hierarchy structure contains at least one path from O to D.*

In this subsection, we describe how TNGD finds Q^h. The likelihood of obtaining the shortest path in the spectrum can be increased by increasing the size of the spectrum.

TNGD algorithm is designed in a way that it returns a spectrum of communities connected through intercommunity arcs. It finds a set of connected communities, the core set CS_{OD}^h, connecting C_O^h and C_D^h with the shortest path through the community centers with the condition that there is at least one intercommunity arc for every consecutive pair of communities on the path. Note that this shortest path is at a particular level h, and the communities along this path identify the candidate communities for exploration at the lower level. Communities in the core set CS^h_{OD} build a spectrum Q^h.

To increase the likelihood of finding the shortest path on the actual road network represented by G^1, TNGD can extend the initial spectrum Q^h by adding neighbor communities of the core set CS^h_{OD}. However, this comes at a cost of increasing run time. Hence, TNGD employs a parameter α to strike a good balance between efficiency (search cost) and effectiveness (path optimality). If $\alpha = 1$, TNGD includes all additional communities with a direct intercommunity arc to the core set, leading to a spectrum of communities $Q^h_{\alpha=1}$. If $\alpha = 2$, TNGD extends the spectrum $Q^h_{\alpha=1}$ by including once again all additional communities with a direct intercommunity arc to the current spectrum. This recursive procedure can be applied for any particular integer $\alpha \geq 1$. If $\alpha = 0$, TNGD returns just the core set. At the lowest level of the hierarchical representation, there is no need to build a spectrum; hence, α is set to zero.

We define a set of notations assuming a time-dependent network in Table 4.1. The proposed TNGD algorithm is given in Algorithm 8. The description of the TNGD algorithm is as follows:

TNGD starts with $G^h, C^h_O,\ C^h_D,\ h$, and α as input parameters. The objective of TNGD is to find a spectrum of communities in the level h using the parameter α. The algorithm uses S and N to store a set of visited communities and a set of communities to visit in the next iteration, respectively. TNGD initializes $S = \emptyset$, $N = C^h_O$, $f_{C^h_O} = \infty$, and the core set $CS^h_{OD} = \{C^h_O,\ C^h_D\}$ (line 1). It also initializes $g_{C^h_O}$ to departure time t_O, and g_u to infinity for all communities u in level h except for C^h_O (line 2). TNGD updates g_u to minimum arrival time from C^h_O to community u (lines 3-22). Note that if $t_O = 0$, g_u is minimum travel time. TNGD selects a community v from N with minimum total travel time (line 4). Estimated minimum total travel time f_v is the sum of the minimum travel time from C^h_O to community v and the heuristic estimate of lower bound on travel time to go from an intermediate community v to the destination community C^h_D, assuming the arrival time to v is g_v (i.e., $f_v \geq g_v + e(v, C^h_D, g_v)$). Then, it removes v from the nominated set N

Algorithm 8 Time-dependent Neighborhood Goal Directed (TNGD) Algorithm ($\mathbf{TNGD(G^h, C_O^h, C_D^h, h, \alpha)}$)

```
1: $v \leftarrow C_O^h,\ S = \emptyset,\ N = \{v\},\ f_v = \infty,\ CS_{OD}^h = \{C_O^h,\ C_D^h\}$
2: $g_v = t_O,\ g_u = \ \infty,\ \forall u \in V^h,\ u \neq v$
3: while $N \neq \emptyset$ do
4:   $v \leftarrow \arg\min_{n \in N} f_n$
5:   if $v \neq C_D^h$ then
6:     $N \leftarrow N \backslash \{v\}$
7:     $S \leftarrow S \cup \{v\}$
8:     for all $u$ where $(v, u) \in A^h$ do
9:       if $u \in S$ then
10:        Continue;
11:      else
12:        if $u \notin\ N$ or $g_v + w_{vu}^{g_v} < g_u$ then
13:          $N \leftarrow N \cup \{u\}$
14:          $g_u \leftarrow g_v + w_{vu}^{g_v}$
15:          $f_u \leftarrow g_u + e(u, C_D^h, g_u)$
16:        end if
17:      end if
18:    end for
19:  else
20:    Break;
21:  end if
22: end while
23: Construct $CS_{OD}^h$
24: {Build a spectrum}
25: $Q^h \leftarrow CS_{OD}^h$
26: $y \leftarrow CS_{OD}^h$
27: while $\alpha > 0$ do
28:   for all $v \in y$ do
29:     for all $u$ where $(v, u) \in A^h$ do
30:       if $u\ \notin Q^h$ then
31:         $Q^h \leftarrow Q^h \cup u$
32:       end if
33:     end for
34:   end for
35:   $y \leftarrow Q^h \backslash y$
36:   $\alpha \leftarrow \alpha - 1$
37: end while
38: Output: $CS_{OD}^h,\ Q^h$
```

and adds it to the visited set S (lines 6-7). TNGD updates N, g_u, and f_u for each neighbor community u of community v (i.e., with a direct intercommunity arc) where either u is not in the nominated set or there is a shorter path using v to reach to u (lines 12-16). If the travel time from the origin community to reach the neighbor community u passing through v, $g_v + w_{vu}^{g_v}$, is smaller than the current travel time of the neighbor g_u, TNGD updates the travel time to the smaller time (lines 12-14). Note that $w_{vu}^{g_v}$ is the time-dependent travel time of the arc (v, u), where the arrival time to v is g_v. Then, TNGD updates the nominated set N (line 13) by adding community u to N. TNGD computes $e(u, C_D^h, g_u)$, which is a lower-bound estimate on travel time to go from u to C_D^h, assuming the arrival time to u is g_u, and then updates f_u (lines 14-15). TNGD can use any lower-bound function to calculate the value of $e(u, C_D^h, g_u)$, for example, travel time from u to C_D^h under free-flow condition can be used as a lower bound (e.g., a vehicle cannot travel from u to C_D^h faster than when it is under free flow condition). While it is desirable to use a tight lower bound such as minimum travel time, calculating such a tight lower bound increases the execution time of the algorithm. If such bounds are calculated offline, the algorithm requires large memory space to save such lower bounds, which is not in alignment with our goal to decrease the need to store preprocessed shortest paths, shortcuts, lower bounds, etc.

TNGD stores communities forming the minimum total travel time path from C_O^h to C_D^h as a core set CS_{OD}^h (line 23). The core set CS_{OD}^h only contains communities in the level h. TNGD initializes the spectrum Q^h by the obtained core set (line 25). To avoid removing some promising communities, the algorithm extends the search space by adding neighbor communities to the selected communities in the core set (lines 27-37), yielding spectrum Q^h of the core set CS_{OD}^h. In doing so, TNGD uses a temporary set y initialized with the core set (line 26). For each community in y, TNGD adds to the spectrum its neighbor communities which do not belong to the spectrum (lines 28-34). Then, TNGD updates y to the set of newly added communities to the spectrum (line 35) and decrements α (line 36). Using y decreases the amount of computation to build the spectrum since TNGD does not need to consider communities that are already belong to the spectrum. The output parameters of TNGD are the core set and the spectrum.

TNGD always finds the shortest path in each level as long as the estimated travel time obtained by the heuristic function is a lower bound of the actual travel time. The goal of proposing TNGD is to reduce the search space in each level of hierarchy by eliminating communities that would not be traversed by the optimal path. In the case of the lowest level where $h = 1$, α is always set to 0. As a result, TNGD in the lowest level becomes a time-dependent goal directed algorithm exploring only a subset of nodes selected by the projection of higher spectrums instead of the whole actual network.

4.3.2 Hierarchical Time-dependent Neighborhood Goal Directed (HTNGD) Algorithm

We now propose the Hierarchical Time-dependent Neighborhood Goal Directed (HTNGD) algorithm that incorporates a new hierarchical search strategy. HTNGD recursively employs TNGD, starting with the highest level of the hierarchy in which O and D fall into two distinct communities. The spectrum of communities resulting from TNGD is recursively projected to the level below, identifying the collection of communities to be searched at the level below. The process terminates at the lowest level, with TNGD identifying the shortest path.

The proposed HTNGD algorithm is given in Algorithm 9. The full details of HTNGD are outlined below. The algorithm receives an OD pair and α as input parameters. It finds the communities C_O^h and C_D^h in the highest level of hierarchy (lines 2-3). If O and D are located within the same community at this level, the algorithm proceeds to the next lower level for the route search (lines 4-6). This procedure continues until O and D fall into different communities. Then, the algorithm executes the TNGD on G^h to find the spectrum Q^h (lines 8- 12).

To eliminate communities that do not belong to the current spectrum from the search space, we set the weights of the intercommunity arcs going out of the spectrum Q^h to infinity (lines 13-23). To do so, for each community in the current spectrum, the algorithm first finds communities that fall into the projection of that community at the lower-level

Algorithm 9 Hierarchical Time-dependent Neighborhood Goal Directed Algorithm ($\mathbf{HTNGD(O, D, \alpha)}$)

```
1: for all levels $h$ from top to bottom do
2:    Find community $C_O^h$ containing $O$ in $G^h$
3:    Find community $C_D^h$ containing $D$ in $G^h$
4:    if $C_O^h = C_D^h$ then
5:       {in the same community}
6:       Continue; {go to the lower level}
7:    else
8:       {in different communities}
9:       if $h = 1$ then
10:          $(CS_{OD}^1,\ Q^1) = TNGD(G^1, C_O^1,\ C_D^1,\ 1,\ 0)$
11:       else
12:          $(CS_{OD}^h,\ Q^h) = TNGD(G^h, C_O^h,\ C_D^h,\ h,\ \alpha)$
13:          {changes in the lower level graph}
14:          for all communities $C_i^h \in Q^h$ do
15:             for all sub-communities $C_j^{h-1}$ of $C_i^h$ do
16:                for all $C_p^{h-1}$ where $C_j^{h-1}C_p^{h-1} \in A^{h-1}$ do
17:                   $C_q^h \leftarrow$ super-community $C_p^{h-1}$ in $G^h$
18:                   if $C_q^h \notin Q^h$ then
19:                      Update $w_{C_j^{h-1}C_p^{h-1}}$ to $\infty$ in $G^{h-1}$
20:                   end if
21:                end for
22:             end for
23:          end for
24:       end if
25:    end if
26: end for
27: Output: Shortest path $CS_{OD}^1$
```

denoted sub-communities. Then, for the selected sub-communities, it finds their neighbor communities with direct intercommunity arc (line 16). If the communities of these neighbors at the level above (denoted super-communities) are not in the spectrum, the algorithm sets the weight of their intercommunity arcs to infinity (line 19). These changes are tracked in G^{h-1}.

The algorithm then proceeds to the lower level and repeats the process until it reaches the lowest level of the hierarchical graph that is the actual road network. However, instead of finding the optimal path in the whole road network, it only searches nodes that are part

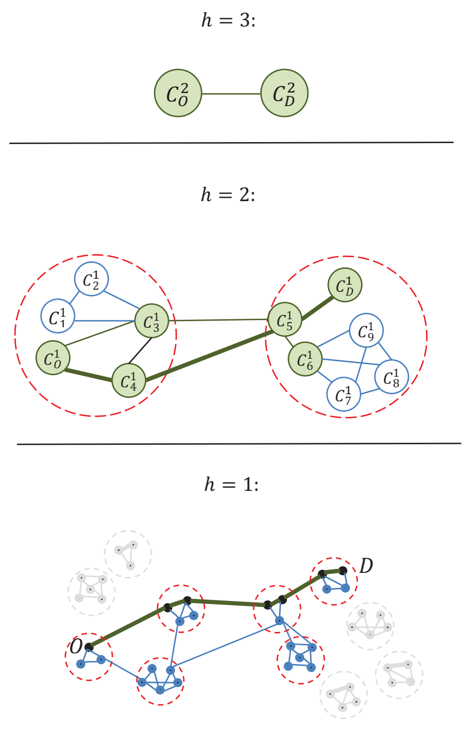

Figure 4.2: Illustrative example for HTNGD

of the projection of spectrum from level $h = 2$. At this lowest level, HTNGD sets α to zero and employs TNGD to find the optimal path from O to D within the reduced search space.

We consider a highly stylized example to illustrate how HTNGD works with $\alpha = 1$. Fig. 4.2 shows three levels of hierarchy, where the top level only has two nodes. We consider O and D to fall into C_O^2 and C_D^2, respectively. All sub-communities of these two communities are shown in the second level, $h = 2$. In this level, O and D fall into C_O^1 and C_D^1, respectively. HTNGD calls TNGD to find the core set, $CS_{OD}^1 = \{C_O^1, C_4^1, C_5^1, C_D^1\}$. Since α is set to one, the spectrum Q^1 contains the immediate neighbor communities of CS_{OD}^1. Therefore, $Q^1 = \{C_O^1, C_3^1, C_4^1, C_5^1, C_6^1, C_D^1\}$. HTNGD eliminates communities not included in Q^1 from further search space, $C_1^1, C_2^1, C_7^1, C_8^1$, and C_9^1. Then, HTNGD projects Q^1 onto the lowest level of the hierarchy, G^1. Finally, HTNGD finds the shortest path between O and D using the reduced search space at this lowest level, $h = 1$. The optimal shortest path is shown by a bold line in Fig. 4.2.

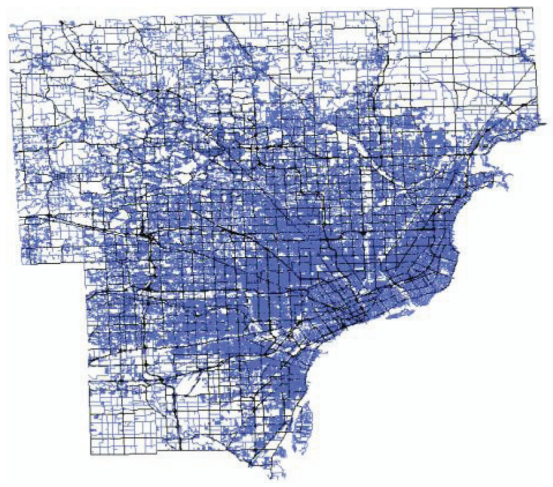

Figure 4.3: Metro Detroit road network: 465,938 arcs and 168,806 nodes

4.4 Experimental Results

We study the performance of our proposed algorithm on the road networks of metropolitan Detroit, New York, and San Francisco. We use two sources for extracting their directed graphs. The first source is NAVTEQ [88] for Metro Detroit. It consists of coordinates of intersections, road segment distances, and speed limits. We extract the graph with its features using ArcGIS Desktop 10. Fig. 4.3 shows the full road network of Metro Detroit. The second source is the center for Discrete Mathematics and Theoretical Computer Science (DIMACS) at Rutgers University [25]. It consists of coordinates of intersections, distance graph, and travel time graph for New York and San Francisco. Table 4.2 shows the number of nodes and arcs of these three road networks. All algorithms are implemented in C++. Experiments are conducted on an Intel 2.53 GHz with 3GB RAM Linux platform.

4.4.1 Generating Time-Dependent Networks

Given the unavailability of time-dependent arc travel times for all arcs of the road networks under study (e.g., ITS coverage is mostly limited to highways), we adopt the following

Table 4.2: Properties of selected road networks

	No. of nodes	No. of arcs
Detroit	168,806	465,938
New York	264,346	733,846
San Francisco	321,270	800,172

procedure for generating such data. Many transportation studies (e.g., [87, 22]) have also employed similar artificially generated time-dependent travel time datasets.

Given that the travel time index (TTI) varies by time of day, we rely on the latest TTI as reported by the Texas Transportation Institute for the cities under study to calibrate the traffic speeds for individual arcs at one-minute resolutions for a typical weekday [120]. TTI corresponds to the ratio of travel time in a particular period to the travel time at free-flow condition. For example, a value of 1.3 for a certain time of day indicates that a 20-minute trip under free-flow condition takes an average of $20 \times 1.3 = 26$ minutes in that period. Note that a TTI of 1 corresponds to the free-flow of traffic without any congestion. Therefore, we equate this to posted speed limits for individual arcs. During rush hours, TTI significantly exceeds 1 and corresponds to reduced traffic speeds. For instance, for the Metro Detroit region the TTI is 1.2 and 1.28 during morning and afternoon rush hours, respectively. Fig. 4.4 shows the TTI for the Metro Detroit region in more detail. We adjusted the traffic speeds for every arc of the network, as a function of time of day, to match the average TTI profile at a one-minute resolution. To generate a representative time dependent travel network, we employed the following approach. We selected coordinates for ten stationary congestion spots covering the Detroit Metro network. Based on the distance proximity between the nearest congestion spot and the mid-point of each arc, the travel time index profile for the arcs (at a one-minute resolution) is generated as follows:

$$w_{ij}^t(1 + (TTI^t - 1)\frac{1}{0.25\lambda_{ij} + 1}) \tag{4.6}$$

where λ_{ij} is the distance proximity between the nearest congestion spot and the mid-point

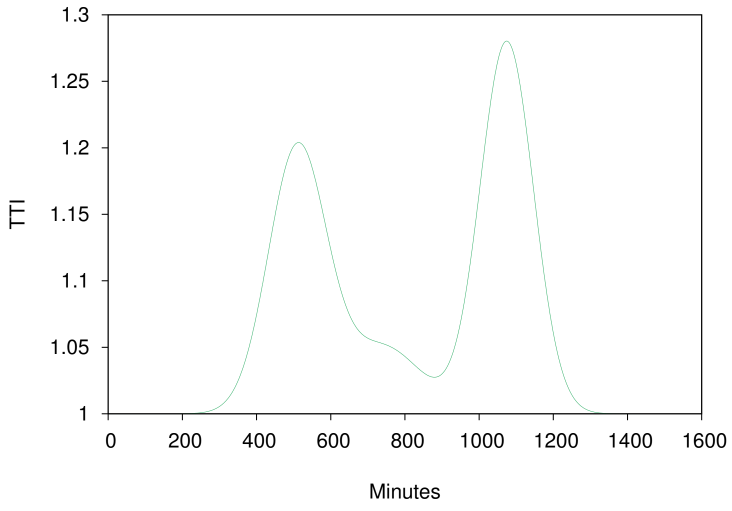

Figure 4.4: Travel time index (TTI)

of the arc from i to j. As designed, different arcs of the network exhibit different travel time index profiles based on their proximity to the congestion spots (nearby arcs will experience the full impact of recurrent congestion and distant arcs will mostly maintain free flow travel conditions). Note that the intention here is not to mimic real-world traffic dynamics but to generate a time dependent network to objectively evaluate the proposed algorithm.

4.4.2 Experimental Setup

We construct the hierarchical representation using travel times under free-flow. Table 4.3 reports the number of communities identified in each level of the hierarchy using the hierarchical community detection algorithm. Louvain's community detection algorithm establishes the same number of hierarchy levels in New York and San Francisco while extracting one more level for Detroit. This is because the Metro Detroit network is sparser than the other two networks. In the first level ($h = 1$), each community contains just a single node from the network. As the level increases in the hierarchy, more nodes are merged to construct each community. Therefore, there are fewer communities at the higher levels. This algorithm finds the hierarchical communities for each studied road network in less than a second.

For all levels of the hierarchy, we build the hierarchical representation $G^h(V^h,\ A^h, W^h)$

Table 4.3: Number of communities in each level

h	Detroit	New York	San Francisco
1	168,806	264,347	321,271
2	67,136	79,261	93,100
3	21,508	18,968	23,104
4	5,833	4,007	5,085
5	1,453	952	1244
6	457	438	672
7	368	-	-

as explained in section 4.2. At the lowest level, $w^h_{C^h_i C^h_j}(t) = w^t_{ij}$, where w^t_{ij} is the time-dependent travel time of going from node i to node j and t is the arrival time at node i. However, in our experiments, for the higher levels, $w^h_{C^h_i C^h_j}(t)$ is the estimated lower bound of travel time from C^h_i to C^h_j based on the straight-line distance between centers of those communities and the speed limit. These estimates can be replaced with more precise information when available, and they may lead to further improvements in computational efficiency. The fixed topology of road networks gives routing algorithms for vehicular networks the benefit of using coordinates; other networks may not have such a privilege. We employ a haversine distance to estimate the distance between any given pair of nodes or communities. Haversine distance d of two vertices i and j is computed using the following formula:

$$\begin{aligned} a =& \sin^2\left(\frac{lat_i - lat_j}{2}\right) + \\ & \cos(lat_i)\cos(lat_j)\,\sin^2\left(\frac{long_i - long_j}{2}\right) \end{aligned} \tag{4.7}$$

$$c = 2\ \text{atan2}(\sqrt{a},\ \sqrt{1-a}) \tag{4.8}$$

$$d = R\ c \tag{4.9}$$

where R is earth's radius (3,961 miles).

We set up an extensive experimental evaluation of our proposed routing algorithm. To analyze effects of OD pairs distance on the proposed algorithm, our tests are executed on five different classes of OD pairs distance: less than 5 miles, 5 to 10 miles, 10 to 20 miles, 20 to 30 miles, and 30 to 40 miles.

We first evaluate the HTNGD using 1,000 randomly selected OD pairs in each class from the road networks of Detroit, New York, and San Francisco, resulting in a total of 15,000 OD pairs (i.e., 1000 $\times$ 5 classes $\times$ 3 cities). We randomly select trip start times throughout the day from 1,440 (i.e., 24 hours $\times$ 60 minutes/hour) time windows. We also perform sensitivity analysis for the spectrum control parameter α over five different values of α for the Detroit dataset. The selected values for α are as follows: 1, 2, 3, $L-h$, and $2(L-h)$, where L is the number of levels and h is the level of hierarchy in the algorithm. To analyze the performance of HTNGD under different traffic conditions, we only consider the Detroit dataset. We choose two distinct traffic conditions: free-flow (early morning) and high traffic (afternoon rush-hour).

4.4.3 Evaluation of HTNGD

As noted earlier, vehicle routing navigation systems, whether built-in or portable, lack the ability to rely on online servers and have to compute the route, given an origin/destination pair and departure time, in a stand-alone mode with limited hardware processing/memory capacity. This mostly renders methods that store preprocessed shortest paths, shortcuts, and lower bounds impractical due to their massive memory requirements. The proposed HTNGD algorithms are explicitly designed to overcome these limitations.

In this subsection, we evaluate the performance of the proposed HTNGD algorithms in time-dependent road networks generated for Detroit, New York, and San Francisco. We compare the results of HTNGD to an adaptation of A* algorithm for time-dependent networks. The reader is referred to [13] for such adaptations. Time-dependent A* algorithms do not require storage of preprocessed shortest paths, shortcuts, or lower bounds, and hence, qualify for fair comparison with the proposed HTNGD algorithms.

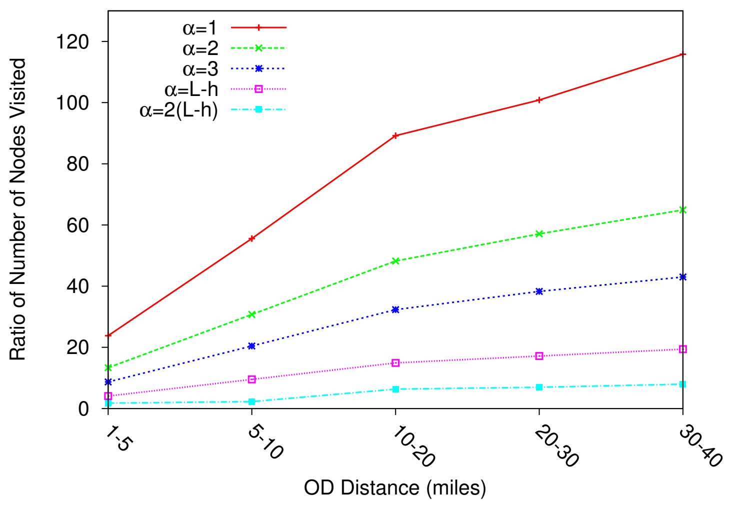

Figure 4.5: Average number of nodes visited by TA* compared to HTNGD during search

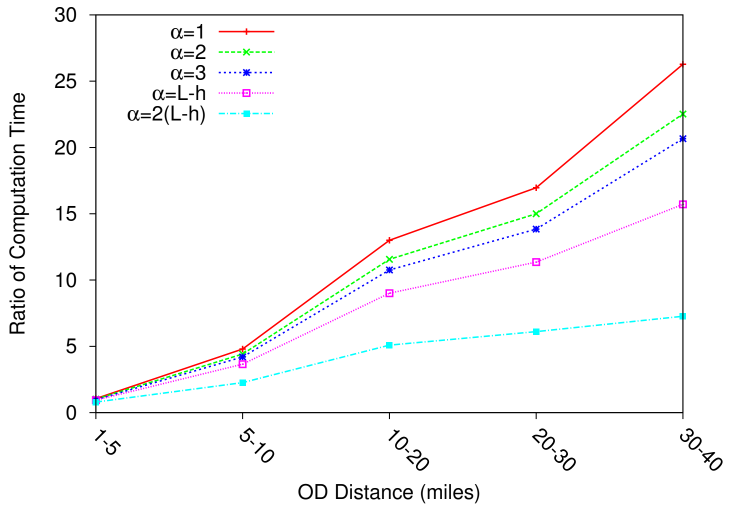

Figure 4.6: Speedup factor for HTNGD vs TA*

$TNGD(G^1, O, D, h = 1, \alpha = 0)$ works as an adaptation of A* on the time-dependent network G^1. This means that TNGD with $\alpha = 0$ on the whole network is a time-dependent A* algorithm (TA*). However, HTNGD on the lowest level calls TNGD with $\alpha = 0$ on the reduced search space. Therefore, for fair analysis of the performance of HTNGD, we compare HTNGD with TA*. Note that TA* always finds the optimal shortest path as long as the estimated travel time obtained by the heuristic function is a lower bound of the actual travel time. This is always the case in our proposed TA*.

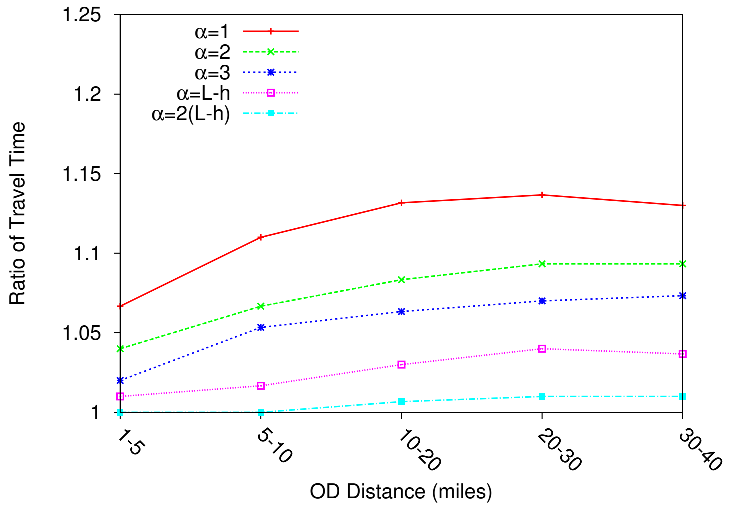

Figure 4.7: Average travel time of the path provided by HTNGD compared to TA*

The ratio of the number of visited nodes in HTNGD compared to TA* on the described test sets are presented in Fig. 4.5. With an increase in the distance between OD pairs, both HTNGD and TA* explore more nodes to find the path. However, as shown in the figure, HTNGD visits many fewer nodes than TA*. This is primarily attributable to the hierarchical search and projection strategy of HTNGD. For example, for the OD distance class of 10-20 miles and $\alpha = L - h$, TA* explores 14.92 times more nodes than HTNGD. This ratio goes upto 89.21 in the case of $\alpha = 1$ for the same OD class.

Fig. 4.6 compares the computational time differences of HTNGD over TA*. The results show significant computational efficiency of HTNGD over TA*. For the case of $\alpha = L - h$, HTNGD is 9.0 times faster than TA* for the OD distance class of 10-20 miles, and is over 15.70 times faster for longer distances. In the case of $\alpha = 1$, HTNGD is 26.27 times faster than TA* for longer distances. The results of Fig. 4.5 and Fig. 4.6 show that the decrease in the number of visited nodes leads to a faster execution time of the HTNGD. This is due to the fact that the decrease in the number of visited nodes reduces the search space leading to a faster execution time.

In addition, we study the optimality of the path identified by HTNGD. We compare the total travel time of the paths obtained by HTNGD and TA* in Fig. 4.7. The results vary based on different values of α. As noted earlier, proper selection of α is critical in

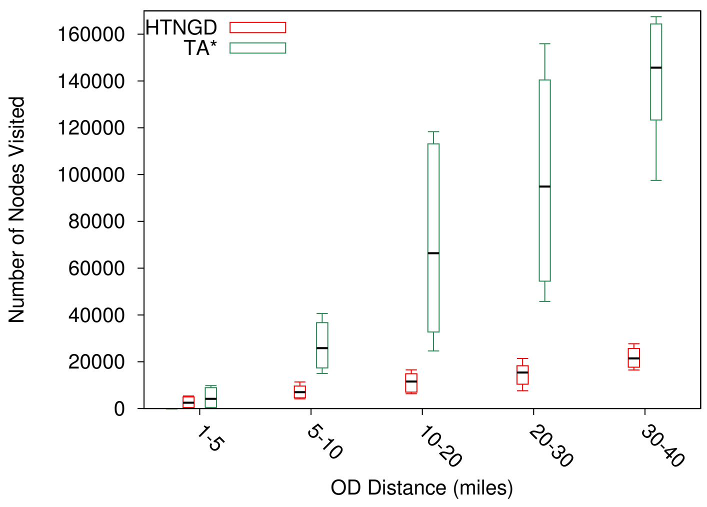

Figure 4.8: Number of visited nodes of HTNGD ($\alpha = 2(L - h)$) and TA*

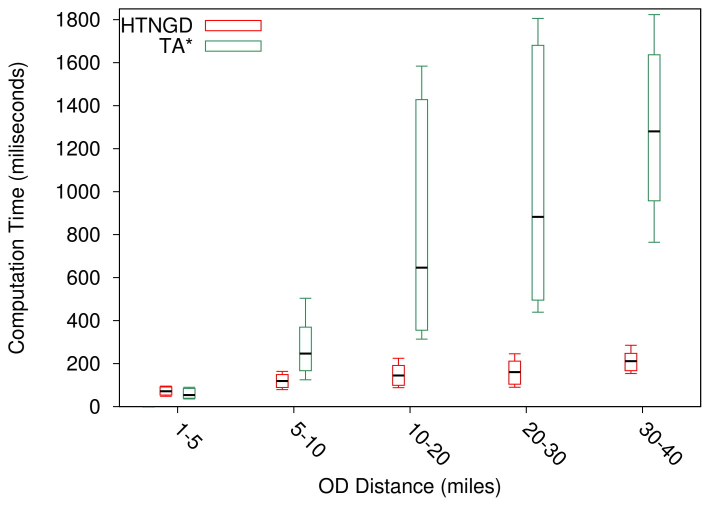

Figure 4.9: Computation time of HTNGD ($\alpha = 2(L - h)$) and TA*

the tradeoff between computation time and optimality gap. The optimality gap is the difference between the travel time of the optimal path and the path obtained by HTNGD. Fig. 4.7 shows the ratio of travel time of HTNGD to TA*. HTNGD with $\alpha = (L - h)$ results in a trip travel time that is 3.0% more than that of TA* for the OD distance class of 10-20 miles. If needed, one can further decrease the optimality gap by increasing the value of parameter α, but at the cost of increasing the execution time.

Figs. 4.8-4.10 describe the results with $\alpha = 2(L - h)$ for the Detroit dataset in more

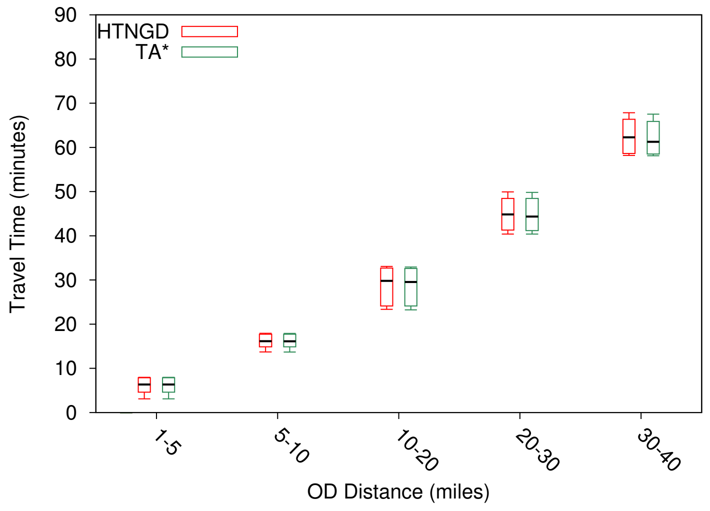

Figure 4.10: Travel time of HTNGD ($\alpha = 2(L - h)$) and TA*

detail. These figures present the distribution of the results with minimum, 10 percentile, average, 90 percentile, and maximum values. Fig. 4.8 presents the distributions of the number of nodes visited during the search. TA* on average visits 6.04 times more nodes than HTNGD. Fig. 4.9 shows the distributions of the computation time of HTNGD compared to TA*. The computation time of HTNGD over all selected OD pairs is on average 4.85 times faster than that of TA*. Clearly, our proposed algorithm performs even better than TA* for longer OD distances (i.e., 6.05 times faster). The distributions of the obtained results for the total travel time are shown in Fig. 4.10. HTNGD results in a total travel time that is on average 1.1% longer than those of TA* for all classes of OD pairs. For all the selected OD pairs in the classes of 1-5 and 5-10 miles, HTNGD with $\alpha = 2(L-h)$ finds the optimal paths.

To investigate the impact of traffic conditions on the performance of our proposed routing algorithm, we now compare the performance of HTNGD with the 5,000 selected OD pairs from the Detroit road network under two distinct traffic conditions: trip start times of midnight (closer to free-flow) versus 6:30PM (experiencing significant recurrent congestion). We study the effects of traffic conditions on our proposed algorithm with $\alpha = 2(L - h)$.

Fig. 4.11 presents the average travel time of HTNGD and TA*. As expected, the results

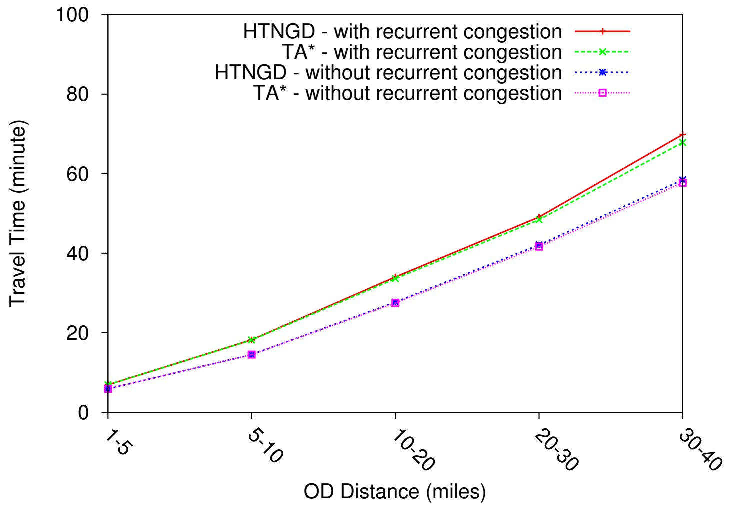

Figure 4.11: Average travel time in early morning vs afternoon rush-hour

show that the average travel time increases for both methods during rush-hour.

Fig. 4.12 shows the average computation time of HTNGD and TA* under free-flow and evening rush-hour. The results show that HTNGD in both test cases performs almost the same. In addition, TA* in both test cases has almost the same computation time. This is due to the fact that the complexity of both algorithms is independent of arc weights, here interpreted as the road segment travel time (with or without congestion). As a result, there are no significant changes in the performance of each algorithm in terms of computation time regarding the traffic congestion.

We compare HTNGD with the most successful speedup techniques in Table 4.4. The reader is referred to [5, 40] for information on other relevant algorithms. We analyze data from several study, and compare pre-processing time, additional storage requirement based on byte per node, and query time. We also present the hardware used in each of the selected studies in the footnote of the table. TNR has the lowest query time, however, it requires 2,760 seconds for pre-processing, and 193 Bytes/node for additional storage space. HTNGD requires the least amount of pre-processing time and storage except than Dijkstra. In addition, query time of HTNGD is reasonable, and it is in terms of milliseconds. In this table, we present the average query time of HTNGD with $\alpha = 2(L - h)$. The query time of HTNGD can be reduced by choosing lower values for α. Note that the query time of all

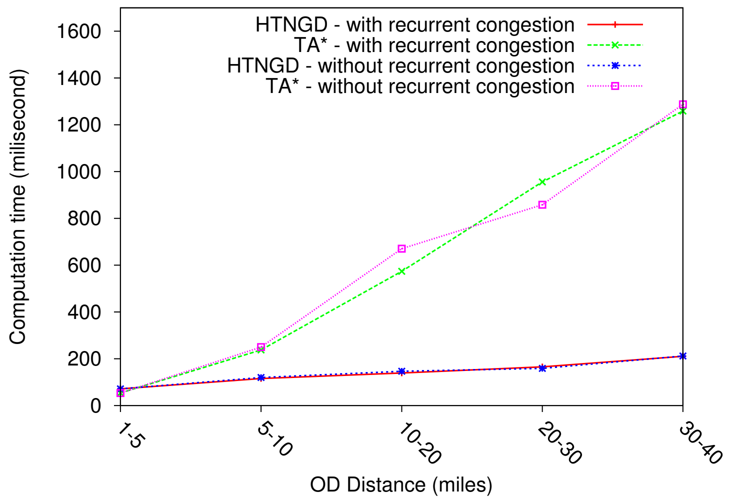

Figure 4.12: Average computation time in early morning vs. afternoon rush-hour

Table 4.4: Comparison of various methods.

Method	Data from	Pre-processing Time (s)	Storage requirement (Byte/node)	Query Time (ms)
Dijkstra[1]	[5]	0	0	5,591.6
TNR[4]	[40]	2,760	193	0.0033
AF[3]	[55]	129,360	25	1.1
SHARC[1]	[4]	4,860	14.5	0.29
HH	[121]	780	48	0.61
CALT[1]	[5]	660	15.4	1.34
ALT[2]	[48]	780	70	120.1
TDCALT[1]	[22]	1,680	256	188.2
HTNGD[5]	this study	0.98	10	141.3

[1]2.6 GHz AMD Opteron, SuSE Linux 10.2, 16GB RAM

[2]2.4 GHz AMD Opteron, Windows Server 2003, 16GB RAM

[3]2.2 GHz AMD Opteron, SuSE Linux 9.1, 4GB RAM

[4]2.0 GHz AMD Opteron, SuSE Linux 10.3, 8GB RAM

[5]2.53 GHz Intel, Fedora Linux 12, 3GB RAM

the algorithms is less than one second.

From all these results, we conclude that HTNGD not only provides accurate route guidance, but also offers significant computational efficiency over other methods without large memory requirements.

4.5 Conclusion

The expanding coverage of Intelligent Transportation Systems is necessitating the development of real-time algorithms for vehicle routing on time-dependent networks. This study provides a new approach for solving the time-dependent shortest path (TDSP) problem on large-scale dynamic networks with deterministic time-varying travel time. In particular, we proposed a hierarchical time-dependent shortest path algorithm to solve the TDSP problem that can utilize community-based hierarchical representations of road networks. The proposed algorithm (HTNGD) generates routes in real-time in terms of milliseconds on large-scale networks without having to store a large number of pre-calculated shortest paths and lower bounds. A key property of the proposed algorithm is its low memory requirements. The significant reduction in memory requirements of HTNGD compared to that of other current methods makes HTNGD suitable to be incorporated in vehicle routing navigation systems. Extensive experimental evaluations of the proposed approach on Detroit, New York, and San Francisco road networks demonstrate the computational efficiency and accuracy of the proposed method. We plan to extend this research to energy-efficient routing of plug-in hybrid and pure electric vehicles.

CHAPTER 5: CONCLUSIONS AND FUTURE RESEARCH

In this chapter, we present a summary and future directions of research that may stem from our work.

5.1 Summary

5.1.1 Optimal Routing for Plug-in Hybrid Electric Vehicles

We introduced the Energy-Efficient Routing problem (EERP) for Plug-in Hybrid Electric Vehicles. We presented the hardness proof of the EERP. We then proposed two exact pseudopolynomial algorithms and an FPTAS algorithm to solve the EERP. From an algorithmic perspective, the proposed two-phase approaches improve the state of the art in optimally solving shortest path problems on general constrained multi-graph networks. In the context of vehicle routing, this is the first study to take into account energy efficiency difference of different operating modes of PHEVs during route planning, which is a high level powertrain energy management procedure. Experimental evaluations of the proposed algorithms on Southeast Michigan road network demonstrate significant fuel economy improvement potential In addition, the results show the computational efficiency and accuracy of the proposed algorithms.

5.1.2 Online Scheduling and Pricing for Electric Vehicle Charging

The dynamics of charging requests and the fact that utility providers need to consider load balancing necessitates designing preemption-aware online mechanisms for EV charging. In this study, we proposed a framework for EV charging considering the incentives of both utility providers and EV drivers. Our proposed framework brings about a win-win situation in which EV drivers can receive their charging requests at lower prices, and utility providers can sell their unused capacity while considering their load balancing objectives. We introduced the problem of online scheduling and pricing for EV charging, and designed a family of online mechanisms, MOSAP-X. We proved that our proposed mechanisms are incentive-compatible, where truthful reporting is a dominant strategy for users. We performed extensive experiments that showed that the proposed mechanisms are not only capable of finding close to optimal solutions, but are also very fast and obtain high revenue. The promising results make MOSAP-X suitable for scheduling and pricing EV charging in real-time.

5.1.3 Hierarchical Time-Dependent Shortest Path for Routing on Dynamic Road Networks under ITS

The expanding coverage of Intelligent Transportation Systems is necessitating the development of real-time algorithms for vehicle routing on time-dependent networks. This study provides a new approach for solving the time-dependent shortest path (TDSP) problem on large-scale dynamic networks with deterministic time-varying travel time. In particular, we proposed a hierarchical time-dependent shortest path algorithm to solve the TDSP problem that can utilize community-based hierarchical representations of road networks. The proposed algorithm (HTNGD) generates routes in real-time in terms of milliseconds on large-scale networks without having to store a large number of pre-calculated shortest paths and lower bounds. A key property of the proposed algorithm is its low memory requirements. The significant reduction in memory requirements of HTNGD compared to that of other current methods makes HTNGD suitable to be incorporated in vehicle routing navigation systems. Extensive experimental evaluations of the proposed approach on Detroit, New York, and San Francisco road networks demonstrate the computational efficiency and accuracy of the proposed method. We plan to extend this research to energy-efficient routing of plug-in hybrid and pure electric vehicles.

5.2 Future Research Directions

We believe that this dissertation will encourage new research work in the area of vehicle electrification. The following are immediate research opportunities and promising directions that can be pursued following our work.

Reducing EV driver's range anxiety by developing range-aware routing and recharging policies brings about new network optimization problems. Decisions include where to stop and how much to recharge at each charging station with deterministic/stochastic travel costs and homogeneous/nonhomogeneous charging stations with various charging rates. A new class of network problems arieses when we consider the associated probability of charging stations availablity.

It would be interesting to extend EERP by considering the combination of a lower level energy management system (powertrain energy management) and our proposed higher level routing algorithms on an actual PHEV in production. Our estimate is that such a combined approach would further improve the fuel consumption savings. Another interesting extension to our research can be incorporating stochastic features into the EERP. However, this extension will make the problem strongly NP-hard, thus, incorporating the stochasticity in the solution methods would be a challenge on large-scale road networks. In addition, considering other objectives such as minimizing travel time along with minimizing energy consumption makes the problem multi-objective, and brings about more challenges. It may be of interest to investigate generalization of the EERP to multi-objective problems (e.g., considering both travel time and fuel economy).

Improvement of EV charging systems performance in varius online and dynamic settings is being highly demanded by a growing number of real world systems. Designing such mechanisms is challenging in both methodology development and implementation, especially in the presence of multiple utility providers. There is an ever-growing need for designing new online mechanisms and unifying frameworks and models for the emerging EV charging markets.

Growing adoption of electrified vehicles along with introduction of new EV models capable of traveling longer distances is demanding effective infrastructure planning and deployment. A number of important research directions arise from this: charging station network design for enhanced system-wide performance. The societal benefits of large-scale adoption of EVs cannot be realized without adequate deployment of publicly accessible charging stations due to mutual dependence of EV sales and public infrastructure deployment. Such infrastructure deployment also presents a number of unique opportunities for promoting livability while helping to reduce the negative side-effects of transportation (e.g., congestion and emissions). Ever-growing need to recharge EVs away from base locations (e.g., residential locations) necessitates designing effective networks of charging stations. Such decision problems need to be tackled by commercial businesses, public authorities, and electric utility providers considering drivers preferences. We should investigate the impacts of charging station network design (e.g., number, type and location of charging stations) on driver's range anxiety, walking distance from the station to their temporary destination (e.g. office and restaurant), cost of charging, etc.

The availability of charging stations and network coverage in urban areas frequented by the EV users is a decision problem that needs to be tackled by commercial businesses and public authorities. In designing the infrastructure for the charging station network, such considerations as drivers' preferences (e.g., drivers may prefer to charge their vehicle at a slower pace for benefiting from overall price reductions) and impact on the local electricity demand must be taken into account. The impact on the local electricity demand is especially relevant in the case of intensive use of rapid charging, which requires a large amount of power to be delivered over a short period. The existing electric infrastructure may not be adequately designed to satisfy the surge in power demand for the electric service stations in these areas.

List of Publications

Journal Articles

J1. Optimal Routing for Plug-in Hybrid Electric Vehicles * [93]
Mahyar Nejad, Lena Mashayekhy, Daniel Grosu, and Ratna Babu Chinnam
Transportation Science, 2015. (under second round review)
*** Best Student Paper Award, INFORMS Energy, Natural Resources, and the Environment Section (ENRE), 2014.**
*** Runner-Up Award for the Best Student Paper, Production and Operations Management Society (POMS), College of Sustainable Operations, 2014.**

J2. Online Scheduling and Pricing for Electric Vehicle Charging [89]
Mahyar Nejad, Lena Mashayekhy, Ratna Babu Chinnam, and Daniel Grosu
IIE Transactions, 2014. (under second round review)

J3. Hierarchical Time-Dependent Shortest Path for Routing on Dynamic Road Networks under ITS [90]
Mahyar Nejad, Lena Mashayekhy, Ratna Babu Chinnam, and Anthony Phillips
IIE Transactions, Vol. 47, 2015. (accepted)

J4. Truthful Greedy Mechanisms for Dynamic Virtual Machine Provisioning and Allocation in Clouds * [92]
Mahyar Nejad, Lena Mashayekhy, and Daniel Grosu
IEEE Transactions on Parallel and Distributed Systems, Vol. 26, No. 2, pp. 594-603, 2015.
*** Best Paper Runner-Up Award - 2014 INFORMS Service Science**

J5. Energy-Aware Scheduling of MapReduce Jobs for Big Data Applications [83]
Lena Mashayekhy, Mahyar Nejad, Daniel Grosu, Quan Zhang, and Weisong Shi
IEEE Transactions on Parallel and Distributed Systems, Vol. 26, 2015 (accepted)

J6. An Online Mechanism for Resource Allocation and Pricing in Clouds [81]
Lena Mashayekhy, Mahyar Nejad, Daniel Grosu, and Athanasios Vasilakos
IEEE Transactions on Computers, Vol. 64, 2015 (accepted)

J7. A PTAS Mechanism for Provisioning and Allocation of Heterogeneous Cloud Resources [79]
Lena Mashayekhy, Mahyar Nejad, and Daniel Grosu
IEEE Transactions on Parallel and Distributed Systems, Vol. 26, 2015 (accepted)

J8. Cloud Federations in the Sky: Formation Game and Mechanism [77]
Lena Mashayekhy, Mahyar Nejad, and Daniel Grosu
IEEE Transactions on Cloud Computing, Vol. 3, No. 1, pp. 14-27, 2015.

J9. Physical Machine Resource Management in Clouds: A Mechanism Design Approach [78]
Lena Mashayekhy, Mahyar Nejad, and Daniel Grosu
IEEE Transactions on Cloud Computing, Special Issue on Economics and Market Mechanisms for Cloud Computing, 2014 (accepted).

J10. Designing customer-oriented catalogs in e-CRM using an effective self-adaptive genetic algorithm [72]
Iraj Mahdavi, Mahyar Nejad, and Fereydoun Adbesh
Expert Systems with Applications, Volume 38, No. 1, January 2011.

Refereed Conference Papers

C1. A Two-Sided Market Mechanism for Trading Big Data Computing Commodities [76]
Lena Mashayekhy, Mahyar Nejad, and Daniel Grosu
Proc. of the IEEE International Conference on Big Data (IEEE BigData 2014), Washington DC, USA, October 2014.

C2. A Framework for Data Protection in Cloud Federations [75]
Lena Mashayekhy, Mahyar Nejad, and Daniel Grosu
Proc. of the 43rd International Conference on Parallel Processing (ICPP'14), pp. 283-290, Minneapolis, USA, September 2014.

C3. Energy-aware Scheduling of MapReduce Jobs [80]
Lena Mashayekhy, Mahyar Nejad, Daniel Grosu, Dajun Lu, and Weisong Shi
Proc. of the 3rd IEEE International Congress on Big Data (BigData'14)-Research Track, pp. 32-39, Anchorage, USA, June 2014. (Acceptance rate: 19%)

C4. Incentive-Compatible Online Mechanisms for Resource Provisioning and Allocation in Clouds [82]
Lena Mashayekhy, Mahyar Nejad, Daniel Grosu, and Athanasios Vasilakos
Proc. of the 7th IEEE International Conference on Cloud Computing (CLOUD'14)-Research Track, pp. 312-319, Anchorage, USA, June 2014. (Acceptance rate: 19%)

C5. A Truthful Approximation Mechanism for Autonomic Virtual Machine Provisioning and Allocation in Clouds [74]
Lena Mashayekhy, Mahyar Nejad, and Daniel Grosu
Proc. of the ACM Cloud and Autonomic Computing Conference (CAC'13), pp. 1-10, Miami, USA, August 2013.

C6. A Family of Greedy Mechanisms for Dynamic Virtual Machine Provisioning and Allocation in Clouds[91]
Mahyar Nejad, Lena Mashayekhy, and Daniel Grosu
Proc. of the 6th IEEE International Conference on Cloud Computing (CLOUD'13)-Research Track, pp. 188-195, Santa Clara, USA, July 2013. (Acceptance rate: 18%)

C7. Effects of Traffic Network Dynamics on Hierarchical Community-based Representations of Large Road Networks [95]
Mahyar Nejad, Lena Mashayekhy, and Ratna Babu Chinnam
Proc. of the 15th IEEE International Intelligent Transportation Systems Conference (ITSC'12), pp. 1900-1905, Anchorage, USA, September 2012.

C8. State Space Reduction in Modeling Traffic Network Dynamics for Dynamic Routing under ITS [94]
Mahyar Nejad, Lena Mashayekhy, Ali Taghavi, and Ratna Babu Chinnam
Proc. of the 14th IEEE International Intelligent Transportation Systems Conference (ITSC'11), pp. 277-282, Washington DC, USA, October 2011.

REFERENCES

[1] B.-H. Ahn and J.-Y. Shin. Vehicle-routeing with time windows and time-varying congestion. *Journal of the Operational Research Society*, pages 393–400, 1991.

[2] R. Ahuja, T. Magnanti, and J. Orlin. *Network flows: theory, algorithms, and applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[3] A. Almuhtady, S. Lee, E. Romeijn, M. Wynblatt, and J. Ni. A degradation-informed battery-swapping policy for fleets of electric or hybrid-electric vehicles. *Transportation Science*, 2014.

[4] R. Bauer and D. Delling. Sharc: Fast and robust unidirectional routing. *Journal of Experimental Algorithmics*, 14:4, 2009.

[5] R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner. Combining hierarchical and goal-directed speed-up techniques for dijkstra's algorithm. *Journal of Experimental Algorithmics*, 15:2–3, 2010.

[6] M. Bazaraa, J. Jarvis, and H. Sherali. *Linear programming and network flows*. Wiley Online Library, 2011.

[7] S. Bera, T. Ojha, and S. Misra. D2p: Distributed dynamic pricing policy in smart grid for phevs management. *IEEE Transactions on Parallel and Distributed Systems*, page 1, 2014.

[8] A. Bernstein. Near linear time (1+ ε)-approximation for restricted shortest paths in undirected graphs. In *Proc. of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 189–201, 2012.

[9] D. Bertsekas. *Linear network optimization: algorithms and codes*. MIT Press, Cambridge, 1991.

[10] M. Bierlaire and F. Crittin. An efficient algorithm for real-time estimation and prediction of dynamic od tables. *Operations Research*, 52(1):116–127, 2004.

[11] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[12] L. Buriol, M. Resende, and M. Thorup. Speeding up dynamic shortest-path algorithms. *INFORMS Journal on Computing*, 20(2):191–204, 2008.

[13] I. Chabini and S. Lan. Adaptations of the a* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks. *IEEE Transactions on Intelligent Transportation Systems*, 3(1):60–74, 2002.

[14] S. Chen, M. Song, and S. Sahni. Two techniques for fast computation of constrained shortest paths. *IEEE/ACM Transactions on Networking*, 16(1):105–115, 2008.

[15] V. Chocteau, D. Drake, P. R. Kleindorfer, R. J. Orsato, and A. Roset. Sustainable fleet operations: The collaborative adoption of electric vehicles. *INSEAD Faculty and Research Working Paper*, 2011.

[16] Y. Chou, H. Romeijn, and R. Smith. Approximating shortest paths in large-scale networks with an application to intelligent transportation systems. *INFORMS Journal on Computing*, 10(2):163–179, 1998.

[17] E. Clarke. Multipart pricing of public goods. *Public choice*, 11(1):17–33, 1971.

[18] A. Clauset, C. Moore, and M. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[19] K. Clement, E. Haesen, and J. Driesen. Coordinated charging of multiple plug-in hybrid electric vehicles in residential distribution grids. In *Proc. of the IEEE/PES Power Systems Conference and Exposition*, pages 1–7, 2009.

[20] K. L. Cooke and E. Halsey. The shortest route through a network with time-dependent internodal transit times. *Journal of mathematical analysis and applications*, 14(3):493–498, 1966.

[21] S. Davis, S. Diegel, and R. Boundy. *Transportation Energy Data Book: Edition 31*. Oak Ridge National Laboratory, Oak Ridge, TN, 2012.

[22] D. Delling and G. Nannicini. Core routing on dynamic time-dependent road networks. *INFORMS Journal on Computing*, 24(2):187–201, 2012.

[23] L. Di Puglia Pugliese and F. Guerriero. A survey of resource constrained shortest path problems: Exact solution approaches. *Networks*, 62(3):183–200, 2013.

[24] I. Diakonikolas and M. Yannakakis. Small approximate pareto sets for biobjective shortest paths and other problems. *SIAM Journal on Computing*, 39(4):1340–1371, 2009.

[25] DIMACS. http://www.dis.uniroma1.it/ challenge9/download.shtml, 2013.

[26] S. E. Dreyfus. An appraisal of some shortest-path algorithms. *Operations Research*, 17(3):395–412, 1969.

[27] I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42(3):135–153, 2003.

[28] J. Eisner, S. Funke, and S. Storandt. Optimal route planning for electric vehicles in large networks. In *Proc. of the 25th Assoc. Advancement Artificial Intell. Conf*, pages 1108–1113, 2011.

[29] F. Ergun, R. Sinha, and L. Zhang. An improved fptas for restricted shortest path. *Information Processing Letters*, 83(5):287–291, 2002.

[30] P. Fairley. Speed bumps ahead for electric-vehicle charging. *IEEE Spectrum*, 47(1):13–14, 2010.

[31] J. Fernández-Madrigal and J. González. Multihierarchical graph search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):103–113, 2002.

[32] C. M. Flath, J. P. Ilg, S. Gottwalt, H. Schmeck, and C. Weinhardt. Improving electric vehicle charging coordination through area pricing. *Transportation Science*, 2013.

[33] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[34] L. Foschini, J. Hershberger, and S. Suri. On the complexity of time-dependent shortest paths. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 327–341, 2011.

[35] L. Fu, D. Sun, and L. Rilett. Heuristic shortest path algorithms for transportation applications: state of the art. *Computers & Operations Research*, 33(11):3324–3343, 2006.

[36] L. Gan, U. Topcu, and S. Low. Optimal decentralized protocol for electric vehicle charging. *IEEE Transactions on Power Systems*, 28(2):940–951, 2013.

[37] S. Gao and I. Chabini. Optimal routing policy problems in stochastic time-dependent networks. *Transportation Research Part B: Methodological*, 40(2):93–122, 2006.

[38] M. Garey and D. Johnson. *Computers and intractability*, volume 174. Freeman, New York, 1979.

[39] R. Garroppo, S. Giordano, and L. Tavanti. A survey on multi-constrained optimal path computation: Exact and approximate algorithms. *Computer Networks*, 54(17):3081–3107, 2010.

[40] R. Geisberger, P. Sanders, D. Schultes, and C. Vetter. Exact routing in large road networks using contraction hierarchies. *Transportation Science*, 46(3):388–404, 2012.

[41] E. H. Gerding, V. Robu, S. Stein, D. C. Parkes, A. Rogers, and N. R. Jennings. Online mechanism design for electric vehicle charging. In *Proc. of the 10th International*

Conference on Autonomous Agents and Multiagent Systems-Volume 2, pages 811–818, 2011.

[42] E. H. Gerding, S. Stein, V. Robu, D. Zhao, and N. R. Jennings. Two-sided online markets for electric vehicle charging. In *Proc. of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 989–996. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[43] A. Gershkov and B. Moldovanu. Efficient sequential assignment with incomplete information. *Games and Economic Behavior*, 68(1):144–154, 2010.

[44] A. Glerum, L. Stankovikj, M. Thémans, and M. Bierlaire. Forecasting the demand for electric vehicles: accounting for attitudes and perceptions. *Transportation Science*, 2014.

[45] A. Goel, K. Ramakrishnan, D. Kataria, and D. Logothetis. Efficient computation of delay-sensitive routes from one source to all destinations. In *Proc. of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 854–858, 2001.

[46] A. Goldberg and C. Harrelson. Computing the shortest path: A search meets graph theory. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 156–165, 2005.

[47] A. Goldberg, H. Kaplan, and R. Werneck. Better landmarks within reach. *Experimental Algorithms*, pages 38–51, 2007.

[48] A. V. Goldberg, H. Kaplan, and R. F. Werneck. Reach for a*: Shortest path algorithms with preprocessing. *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, 74:93–139, 2009.

[49] Q. Gong, Y. Li, and Z. Peng. Trip-based optimal power management of plug-in hybrid electric vehicles. *IEEE Transactions on Vehicular Technology*, 57(6):3393–3401, 2008.

[50] T. Groves. Incentives in teams. *Econometrica: Journal of the Econometric Society*, 41(4):617–631, 1973.

[51] R. Guimera and L. Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, 2005.

[52] M. Hajiaghayi, R. Kleinberg, M. Mahdian, and D. C. Parkes. Online auctions with re-usable goods. In *Proc. of 6th ACM conf. on Electronic commerce*, pages 165–174. ACM, 2005.

[53] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, 1992.

[54] B. Heydenreich, R. Müller, and M. Uetz. Games and mechanism design in machine scheduling: an introduction. *Production and Operations Management*, 16(4):437–454, 2007.

[55] M. Hilger, E. Köhler, R. H. Möhring, and H. Schilling. Fast point-to-point shortest path computations with arc-flags. *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, 74:41–72, 2009.

[56] J. Huang, M. Leng, L. Liang, and J. Liu. Promoting electric automobiles: supply chain analysis under a governments subsidy incentive scheme. *IIE Transactions*, 45(8):826–844, 2013.

[57] Hybrid Vehicle Timeline, 2012.

[58] O. Ibarra and C. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–468, 1975.

[59] G. Jagadeesh, T. Srikanthan, and K. Quek. Heuristic techniques for accelerating hierarchical routing on road networks. *IEEE Transactions on Intelligent Transportation Systems*, 3(4):301–309, 2002.

[60] C. Jin, J. Tang, and P. Ghosh. Optimizing electric vehicle charging: a customer's perspective. *IEEE Transactions on Vehicular Technology*, 62(7):2919–2927, 2013.

[61] S. Jung and S. Pramanik. An efficient path computation model for hierarchically structured topographical road maps. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1029–1046, 2002.

[62] G. S. Kasbekar and S. Sarkar. Spectrum auction framework for access allocation in cognitive radio networks. *IEEE/ACM Transactions on Networking*, 18(6):1841–1854, 2010.

[63] D. E. Kaufman and R. L. Smith. Fastest paths in time-dependent networks for intelligent vehicle-highway systems application. *Journal of Intelligent Transportation Systems*, 1(1):1–11, 1993.

[64] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.

[65] K. Kieckhäfer, T. Volling, and T. S. Spengler. A hybrid simulation approach for estimating the market share evolution of electric vehicles. *Transportation Science*, 2014.

[66] R. Lavi and N. Nisan. Competitive analysis of incentive compatible on-line auctions. *Theoretical Computer Science*, 310(1-3):159–180, 2004.

[67] W.-Y. Lin, G.-Y. Lin, and H.-Y. Wei. Dynamic auction mechanism for cloud resource allocation. In *Proc. of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 591–592, 2010.

[68] Z. Lin. Optimizing and diversifying electric vehicle driving range for us drivers. *Transportation Science*, 2014.

[69] D. Lorenz and A. Orda. Qos routing in networks with uncertain parameters. *IEEE/ACM Transactions on Networking*, 6(6):768–778, 1998.

[70] D. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 28(5):213–219, 2001.

[71] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel. Delay efficient sleep scheduling in wireless sensor networks. In *Proc. of the IEEE INFOCOM 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 2470–2481, 2005.

[72] I. Mahdavi, M. Nejad, and F. Adbesh. Designing customer-oriented catalogs in e-crm using an effective self-adaptive genetic algorithm. *Expert Systems with Applications*, 38(1):631–639, 2011.

[73] H.-Y. Mak, Y. Rong, and Z.-J. M. Shen. Infrastructure planning for electric vehicles with battery swapping. *Management Science*, 59(7):1557–1575, 2013.

[74] L. Mashayekhy, M. Nejad, and D. Grosu. A truthful approximation mechanism for autonomic virtual machine provisioning and allocation in clouds. In *Proc. of the ACM Cloud and Autonomic Computing Conf.*, pages 1–10, 2013.

[75] L. Mashayekhy, M. Nejad, and D. Grosu. A framework for data protection in cloud federations. In *Proc. 43rd IEEE Intl. Conf. on Parallel Processing*, 2014.

[76] L. Mashayekhy, M. Nejad, and D. Grosu. A two-sided market mechanism for trading big data computing commodities. In *Proc. of the IEEE Intl. Conf. on Big Data*, 2014.

[77] L. Mashayekhy, M. Nejad, and D. Grosu. Cloud federations in the sky: Formation game and mechanism. *IEEE Transactions on Cloud Computing*, PrePrints, 2014.

[78] L. Mashayekhy, M. Nejad, and D. Grosu. Physical machine resource management in clouds: A mechanism design approach. *IEEE Transactions on Cloud Computing*, PrePrints, 2014.

[79] L. Mashayekhy, M. Nejad, and D. Grosu. A PTAS mechanism for provisioning and allocation of heterogeneous cloud resources. *IEEE Transactions on Parallel and Distributed Systems*, PrePrints, 2014.

[80] L. Mashayekhy, M. Nejad, D. Grosu, D. Lu, and W. Shi. Energy-aware scheduling of mapreduce jobs. In *Proc. of the 3rd IEEE Intl. Congress on Big Data*, pages 32–39, 2014.

[81] L. Mashayekhy, M. Nejad, D. Grosu, and A. Vasilakos. An online mechanism for resource allocation and pricing in clouds. *IEEE Transactions on Computers (accepted)*, PrePrints, 2015.

[82] L. Mashayekhy, M. Nejad, D. Grosu, and A. V. Vasilakos. Incentive-compatible online mechanisms for resource provisioning and allocation in clouds. In *Proc. of the 7th IEEE Intl. Conf. on Cloud Computing*, pages 312–319, 2014.

[83] L. Mashayekhy, M. Nejad, D. Grosu, Q. Zhang, and W. Shi. Energy-aware scheduling of mapreduce jobs for big data applications. *IEEE Transactions on Parallel and Distributed Systems*, PrePrints, 2014.

[84] K. Mehlhorn and M. Ziegelmann. Resource constrained shortest paths. In *Proc. of the 8th Annual European Symposium on Algorithms*, pages 326–337, 2000.

[85] A. Mu'Alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. *Games and Economic Behavior*, 64(2):612–631, 2008.

[86] N. Murgovski, L. Johannesson, and J. Sjöberg. Engine on/off control for dimensioning hybrid electric powertrains via convex optimization. *IEEE Transactions on Vehicular Technology*, 62(7):2949–2962, 2013.

[87] G. Nannicini, D. Delling, L. Liberti, and D. Schultes. Bidirectional a search for time-dependent fast paths. *Experimental Algorithms*, pages 334–346, 2008.

[88] NAVTEQ. http://www.navteq.com/, 2013.

[89] M. Nejad, L. Mashayekhy, R. Chinnam, and D. Grosu. Online scheduling and pricing for electric vehicle charging. *IIE Transactions (under second review)*, 2015.

[90] M. Nejad, L. Mashayekhy, R. Chinnam, and A. Phillips. Hierarchical Time-Dependent Shortest Path for Routing on Dynamic Road Networks under ITS. *IIE Transactions*, PrePrints, 2015.

[91] M. Nejad, L. Mashayekhy, and D. Grosu. A family of truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds. In *Proc. of the 6th IEEE Intl. Conf. on Cloud Computing*, pages 188–195, 2013.

[92] M. Nejad, L. Mashayekhy, and D. Grosu. Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds. *IEEE Transactions on Parallel and Distributed Systems*, 26(2):594 – 603, 2015.

[93] M. Nejad, L. Mashayekhy, D. Grosu, and R. Chinnam. Optimal routing for plug-in hybrid electric vehicles. *INFORMS Transportation Science (under second review)*, 2014.

[94] M. Nejad, L. Mashayekhy, A. Taghavi, and R. Chinnam. State space reduction in modeling traffic network dynamics for dynamic routing under its. In *Proceedings of the 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 277–282, 2011.

[95] M. M. Nejad, L. Mashayekhy, and R. B. Chinnam. Effects of traffic network dynamics on hierarchical community-based representations of large road networks. In *Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems*, pages 1900–1905, 2012.

[96] M. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2004.

[97] M. Newman. Communities, modules and large-scale structure in networks. *Nature Physics*, 8(1):25–31, 2011.

[98] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[99] G. Palla, A. Barabasi, and T. Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, 2007.

[100] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[101] D. C. Parkes. Online mechanisms. In N. Nisan, T. Roughgarden, Éva Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, 2007.

[102] D. C. Parkes and S. Singh. An MDP-based approach to online mechanism design. In *Proc. of 17th Annual Conf. on Neural Information Processing Systems*, 2003.

[103] C. Phillips. The network inhibition problem. In *Proc. of the 25th annual ACM symposium on Theory of computing*, pages 776–785, 1993.

[104] S. Plotkin, D. Santini, A. Vyas, J. Anderson, M. Wang, D. Bharathan, and J. He. Hybrid electric vehicle technology assessment: methodology, analytical issues, and interim results. Technical report, Argonne National Lab., IL (US), 2002.

[105] P. Pons and M. Latapy. Computing communities in large networks using random walks. *Computer and Information Sciences-ISCIS 2005*, pages 284–293, 2005.

[106] R. Porter. Mechanism design for online real-time scheduling. In *Proc. of 5th ACM conf. on Electronic commerce*, pages 61–70. ACM, 2004.

[107] priuschat.com, 2014.

[108] J. S. Provan. A polynomial-time algorithm to find shortest paths with recourse. *Networks*, 41(2):115–125, 2003.

[109] L. Pugliese and F. Guerriero. A reference point approach for the resource constrained shortest path problems. *Transportation Science*, 47(2):247–265, 2013.

[110] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663, 2004.

[111] R. Rajagopalan, K. Mehrotra, C. Mohan, and P. Varshney. Hierarchical path computation approach for large graphs. *IEEE Transactions on Aerospace and Electronic Systems*, 44(2):427–440, 2008.

[112] D. Raz and Y. Shavitt. Optimal partition of qos requirements with discrete cost functions. In *Proc. of the IEEE INFOCOM 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 613–622, 2000.

[113] V. Robu, E. H. Gerding, S. Stein, D. C. Parkes, A. Rogers, and N. R. Jennings. An online mechanism for multi-unit demand and its application to plug-in hybrid electric vehicle charging. *Journal of Artificial Intelligence Research*, 2013.

[114] V. Robu, S. Stein, E. H. Gerding, D. C. Parkes, A. Rogers, and N. R. Jennings. An online mechanism for multi-speed electric vehicle charging. In *Auctions, Market Mechanisms, and Their Applications*, pages 100–112. Springer, 2012.

[115] J. Romm and A. Frank. Hybrid vehicles gain traction. *Scientific American*, 294(4):72–79, 2006.

[116] M. Sachenbacher, M. Leucker, A. Artmeier, and J. Haselmayr. Efficient energy-optimal routing for electric vehicles. In *Proc. of the Assoc. Advancement Artificial Intell. Conf*, pages 1402–1407, 2011.

[117] S. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23(1):116–127, 1976.

[118] L. Santos, J. Coutinho-Rodrigues, and J. Current. An improved solution algorithm for the constrained shortest path problem. *Transportation Research Part B: Methodological*, 41(7):756–771, 2007.

[119] M. Schneider, A. Stenger, and D. Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 2014.

[120] D. Schrank and T. Lomax. 2012 urban mobility report. Technical report, Texas A&M Univ. Syst., College Station, TX, 2012.

[121] D. Schultes. *Route Planning in Road Networks*. PhD thesis, Universität Karlsruhe, 2008.

[122] A. Sciarretta and L. Guzzella. Control of hybrid electric vehicles. *IEEE Control systems*, 27(2):60–70, 2007.

[123] H. Sherali and J. Hill. Reverse time-restricted shortest paths: Application to air traffic management. *Transportation Research Part C: Emerging Technologies*, 17(6):631–641, 2009.

[124] R. Sioshansi. Or forum-modeling the impacts of electricity tariffs on plug-in hybrid electric vehicle charging, costs, and emissions. *Operations Research*, 60(3):506–516, 2012.

[125] Q. Song and X. Wang. Efficient routing on large road networks using hierarchical communities. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):132–140, 2011.

[126] S. Stein, E. Gerding, V. Robu, and N. R. Jennings. A model-based online mechanism with pre-commitment and its application to electric vehicle charging. In *Proc. of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 669–676. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[127] S. Stüdli, W. Griggs, E. Crisostomi, and R. Shorten. On optimality criteria for reverse charging of electric vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 15(1), 2014.

[128] O. Sundstrom and C. Binding. Flexible charging optimization for electric vehicles considering distribution grid constraints. *IEEE Transactions on Smart Grid*, 3(1):26–37, 2012.

[129] T. Sweda and D. Klabjan. Finding minimum-cost paths for electric vehicles. In *Proc. of the IEEE Intl. Electric Vehicle Conf*, pages 1–4, 2012.

[130] M. Vasirani and S. Ossowski. A computational monetary market for plug-in electric vehicle charging. In *Auctions, Market Mechanisms, and Their Applications*, pages 88–99. Springer, 2012.

[131] V. Vazirani. *Approximation algorithms.* Springer, 2004.

[132] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[133] S. T. Waller and A. K. Ziliaskopoulos. On the online shortest path problem with limited arc cost dependencies. *Networks*, 40(4):216–227, 2002.

[134] A. Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35(1):70–79, 1987.

[135] L. Wei and Y. Guan. Optimal control of plug-in hybrid electric vehicles with market impact and risk attitude. *Transportation Science*, 2014.

[136] G. Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (fptas)? *INFORMS Journal on Computing*, 12(1):57–74, 2000.

[137] L. Zhang, Z. Li, and C. Wu. Dynamic resource provisioning in cloud computing: A randomized auction approach. In *Proc. of IEEE INFOCOM*, pages 433–441, 2014.

[138] X. Zhou, S. Gandhi, S. Suri, and H. Zheng. ebay in the sky: strategy-proof wireless spectrum auctions. In *Proc. of the 14th ACM Intl. Conf. on Mobile Comp. and Networking*, pages 2–13, 2008.

ABSTRACT

FRONTIERS IN OPERATIONS RESEARCH FOR OVERCOMING BARRIERS TO VEHICLE ELECTRIFICATION

by

MAHYAR MOVAHED NEJAD

August 2015

Advisor: Dr. Ratna Babu Chinnam

Major: Industrial Engineering

Degree: Doctor of Philosophy

Electric vehicles (EVs) hold many promises including diversification of the transportation energy feedstock and reduction of greenhouse gas and other emissions. However, achieving large-scale adoption of EVs presents a number of challenges resulting from a current lack of supporting infrastructure and difficulties in overcoming technological barriers. This dissertation addresses some of these challenges by contributing to the advancement of theories in the areas of network optimization and mechanism design.

To increase the electric driving range of plug-in hybrid electric vehicles (PHEVs), we propose a power-train energy management control system that exploits energy efficiency differences of the electric machine and the internal combustion engine during route planning. We introduce the Energy-Efficient Routing problem (EERP) for PHEVs, and formulate this problem as a new class of the shortest path problem. We prove that the EERP is NP-complete. We then propose two exact algorithms that find optimal solutions by exploiting the transitive structure inherent in the network. To tackle the intractability of the problem, we proposed a Fully Polynomial Time Approximation Scheme (FPTAS). From a theoretic perspective, the proposed two-phase approaches improve the state-of-the-art to optimally solving shortest path problems on general constrained multi-graph networks. These novel approaches are scalable and offer broad potential in many network optimization problems. In the context of vehicle routing, this is the first study to take into account energy efficiency

difference of different operating modes of PHEVs during route planning, which is a high level power-train energy management procedure.

Another challenge for EV adoption is the inefficiency of current charging systems. In addition, high electricity consumption rates of EVs during charging make the load management of micro grids a challenge. We proposed an offline optimal mechanism for scheduling and pricing of electric vehicle charging considering incentives of both EV owners and utility companies. In the offline setting, information about future supply and demand is known to the scheduler. By considering uncertainty about future demand, we then designed a family of online mechanisms for real-time scheduling of EV charging. A fundamental problem with significant economic implications is how to price the charging units at different times under dynamic demand. We propose novel bidding based mechanisms for online scheduling and pricing of electric vehicle charging. The proposed preemption-aware charging mechanisms consider incentives of both EV drivers and grid operators. We also prove incentive-compatibility of the mechanisms, that is, truthful reporting is a dominant strategy for self-interested EV drivers. The proposed mechanisms demonstrate the benefits of electric grid load management, revenue maximization, and quick response, key attributes when providing online charging services.

AUTOBIOGRAPHICAL STATEMENT

Mahyar Movahed Nejad received his BSc degree in mathematics from Iran University of Science and Technology. He received his MSc degree in socio-economic systems engineering from Mazandaran University of Science and Technology. He received his second MSc degree in computer science from Wayne State University. He is currently a PhD candidate in industrial and systems engineering at Wayne State University. His research interests research interests include network optimization, algorithmic game theory, integer programming, cloud computing, distributed data analytics, and electric vehicles. His papers appeared in venues such as IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, IIE Transactions, IEEE CLOUD, and IEEE BigData. Mahyar received several best paper awards including: the 2014 INFORMS ENRE best student paper award, and runner-up awards for the 2014 POMS College of Sustainable Operations Best Student Paper Competition and the 2014 INFORMS Service Science Best Paper Award. He is a student member of the IEEE, ACM, INFORMS, and IIE. Following graduation, Mahyar will join the University of Oklahoma as an assistant professor.