http://pubsonline.informs.org/journal/trsc/

TRANSPORTATION SCIENCE
Vol. 51, No. 4, November 2017, pp. 1304–1325
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

Optimal Routing for Plug-In Hybrid Electric Vehicles

Mark M. Nejad,[a] Lena Mashayekhy,[b] Daniel Grosu,[c] Ratna Babu Chinnam[d]

[a] Department of Civil and Environmental Engineering, University of Delaware, Newark, Delaware 19716; [b] Department of Computer and Information Sciences, University of Delaware, Newark, Delaware 19716; [c] Department of Computer Science, Wayne State University, Detroit, Michigan 48202; [d] Department of Industrial and Systems Engineering, Wayne State University, Detroit, Michigan 48202

Contact: nejad@udel.edu (MMN); mlena@udel.edu (LM); dgrosu@wayne.edu (DS); r_chinnam@wayne.edu (RBC)

Received: September 28, 2014
Revised: December 13, 2015; March 28, 2016; May 9, 2016
Accepted: May 19, 2016
Published Online in Articles in Advance: March 24, 2017

https://doi.org/10.1287/trsc.2016.0706

Abstract. We propose routing algorithms for plug-in hybrid electric vehicles (PHEVs) that account for the significant energy efficiency differences of vehicle operating modes and recommend the predominant mode of operation for each road segment during route planning. This is to enhance fuel economy and reduce emissions. 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 to minimize fuel consumption. EERP can be generalized to a new class of problems in the context of network optimization, where for each arc we need to choose which resources to use to minimize the consumption of one of the resources subject to a constraint on the other resource. In this problem, the resource selection is mutually exclusive, which means we cannot choose both resources together for an arc. We prove that the EERP is NP-complete. We then propose two exact algorithms, and a fully polynomial time approximation scheme (FPTAS) to solve the EERP. We demonstrate the performance of our proposed exact and FPTAS algorithms using road network data from Southeast Michigan. The results show that incorporating our proposed algorithms during route planning leads to significant energy savings for PHEVs over simplistic routing algorithms and current practice.

Keywords: energy-efficient routing • plug-in hybrid electric vehicles • approximation schemes

1. Introduction

Electrified vehicles (EVs) 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 United States, 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 EVs. Several companies have accepted the challenge, and more models of EVs (hybrid vehicles, plug-in hybrid vehicles, pure battery electric vehicles) are being introduced every year around the world (Hybrid Vehicle Timeline 2012).

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 plug-in hybrid electric vehicle (PHEV) with a series configuration, 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 paper, we consider parallel plug-in hybrid electric vehicles, and refer to 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).

The literature offers a number of routing algorithms for BEVs (Sachenbacher et al. 2011, Eisner, Funke, and Storandt 2011, Sweda and Klabjan 2012). 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 (miles per hour), the electric mode is relatively efficient on city roads with lower speed limits (Plotkin et al. 2002). 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 Sciarretta and Guzzella (2007). 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) have 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 a 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 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 paper, 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 presegmented 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 paper, 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 United States, is currently several times cheaper than using the gasoline mode (Romm and Frank 2006, Sioshansi 2012). 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 paper, 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).

1.1. Our Contribution

We address the energy-efficient routing problem (EERP) for PHEVs by designing exact and approximation algorithms for solving it. The primary contributions of this paper 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.

The proposed algorithms are applicable to the emerging field of EV routing and to more general network optimization problems. In the context of network optimization, our proposed algorithms improve the state of the art in constrained shortest path, where for each arc we need to choose which resource to use to minimize consumption of one of the resources subject to a constraint on the other resource. In this problem, the resource selection is mutually exclusive, which means we cannot choose both resources together for an arc. The algorithms proposed in this paper can also be applied to solve such a generalized problem.

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 pseudo-polynomial 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, A is the set of arcs in the network, and V is the set of vertices in the network. 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.

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 to vehicle routing problems (Bertsekas 1991, Ahuja, Magnanti, and Orlin 1993, Bazaraa, Jarvis, and Sherali 2009).

We focus on studies of SPP that are closely related to our work, in particular, those on the resource constrained shortest path problem (RCSPP). The RCSPP 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 Di Puglia Pugliese and Guerriero (2013a). The solution methods for the RCSPP can be briefly classified into path ranking methods (e.g., Santos, Coutinho-Rodrigues, and Current 2007), Lagrangian relaxation methods (e.g., Mehlhorn and Ziegelmann 2000), and node labeling methods (e.g., Di Puglia Pugliese and Guerriero 2013b). RCSPP has been studied extensively, however, here we only focus on a special case of RCSPP that considers one resource consumption. This problem is referred 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 (Garey and Johnson 1979), even if the graph is acyclic, and all length and transition times are positive (Dumitrescu and Boland 2003). However, the RSPP is NP-complete if the graph does not contain negative length cycles, and can be solved in pseudo-polynomial time (Di Puglia Pugliese and Guerriero 2013b). For a survey on exact and approximate solution approaches for the RSPP and the RCSPP we refer the reader to Garroppo, Giordano, and Tavanti (2010). Warburton (1987) was the first to introduce a polynomial-time approximation scheme (PTAS) for the RSPP on acyclic graphs. Hassin (1992) improved the results of Warburton (1987) 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. Phillips (1993) proposed a PTAS for the RSPP, which uses a Dijkstra-based algorithm with time complexity of $O((|A||V|/\epsilon + (|V|^2/\epsilon)\log(|V|^2/\epsilon))\log\log(U/L))$. Goel et al. (2001) 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. Lorenz and Raz (2001) reduced the complexity of the results by Hassin (1992) to $O(|A||V|(1/\epsilon+\log\log(U/L)))$ by proposing an FPTAS. Ergun, Sinha, and Zhang (2002) 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. Chen, Song, and Sahni (2008) 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 Goel et al. (2001). 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 Diakonikolas and Yannakakis (2009), where they approximated the Pareto curves. Bernstein (2012) 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 Lorenz and Orda (1998), Raz and Shavitt (2000), Lu et al. (2005), and Sherali and Hill (2009).

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. As a result, in our problem two resources cannot be selected together, which is the case for RSPP (where delay and length are properties that are selected together by choosing an arc). In our case, electricity and gasoline cannot be selected together for the same 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.

Research on different decision problems related to EVs has attracted a great deal of attention in the past few years. Such research includes forecasting EV market shares (Glerum et al. 2014, Kieckhäfer, Volling, and Spengler 2014), proposing battery-swapping policies (Mak, Rong, and Shen 2013, Almuhtady et al. 2014), managing EV powertrains (Salmasi 2007), pricing EV charging (Nejad et al. 2017), and placing charging stations (Xi, Sioshansi, and Marano 2013). Sweda, Dolinskaya, and Klabjan (2017) studied the problem of finding an optimal recharging policy for a BEV along a given path. Schneider, Stenger, and Goeke (2014) studied an EV routing problem with time windows and recharging stations incorporating the possibility of recharging at any of the available stations. Lin (2014) 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. Bay and Limbourg (2015) defined the EV traveling salesman problem. They presented models and preliminary results focusing on maximizing the battery state of charge at the end of the trip, and thus, maximizing the driving range of the BEV. Several researchers investigated the powertrain management for hybrids and PHEVs from an energy management control perspective (e.g., Gong et al. 2008, Murgovski, Johannesson, and Sjöberg 2013). 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.

1.3. Organization

The rest of the paper is organized as follows. In Section 2, we introduce the energy-efficient routing problem for PHEVs. In Section 3, we describe our proposed exact algorithms for solving the EERP. In Section 4, we propose an FPTAS for solving the EERP and characterize its properties. In Section 5, we evaluate the proposed algorithms by extensive experiments on the Southeast Michigan road network. In Section 6, we conclude the paper and present possible directions for future research.

2. Energy-Efficient Routing Problem for PHEV

PHEVs can run on both electric and gasoline modes as long as there is an adequate charge in the battery and the gasoline tank is not empty, respectively. This feature necessitates routing algorithms to incorporate decisions on the vehicle's operating modes in path planning. The decision in EERPs for PHEVs involve 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 the gasoline consumption (in gallons) or battery charge consumption (in watt hours) with it. 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 types is denoted by A. We associate with each arc $(i, j)^e \in A$ a weight w^e_{ij} 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^g_{ij} 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:

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

$$y_{ij} = \begin{cases} 1 & \text{if the vehicle passes through arc } (i, j) \text{ in electric mode,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The vehicle has an available battery SOC $C \in \mathbb{Z}_{\geq 0}$ (nonnegative integer value) 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

Table 1. Notation

V	Set of nodes
A^e	Set of all arcs $(i,j)^e \in A$
A^g	Set of all arcs $(i,j)^g \in A$
w^e_{ij}	Battery charge consumption when traversing arc (i,j)
w^g_{ij}	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

is given by $\Gamma = \sum_{(i,j)\in P} w^g_{ij} x_{ij}$. Similarly, the total battery charge consumption E on a path P in G_R from O to D is given by $E = \sum_{(i,j)\in P} w^e_{ij} y_{ij}$. Table 1 summarizes the notation used throughout the paper.

We define EERP for PHEVs as follows.

Definition 3 (Energy-Efficient Routing Problem (EERP)). Given a fuel consumption network $G = (V, A)$ 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:

$$\text{Minimize} \quad \sum_{(i,j)\in A^g} w^g_{ij} x_{ij} \tag{3}$$

$$\text{Subject to:} \quad \sum_{i\in V} (x_{Oi} + y_{Oi}) = 1, \tag{4}$$

$$\sum_{i\in V} (x_{iD} + y_{iD}) = 1, \tag{5}$$

$$\sum_{i\in V} (x_{ij} + y_{ij}) - \sum_{k\in V} (x_{jk} + y_{jk}) = 0, \quad \forall\, j \in V;\ j \notin \{O, D\}, \tag{6}$$

$$\sum_{(i,j)\in A^e} w^e_{ij} y_{ij} \le C, \tag{7}$$

$$x_{ij} + y_{ij} \le 1, \quad \forall\, i, j \in V, \tag{8}$$

$$x_{ij},\ y_{ij} \in \{0,1\}, \quad \forall\, i, j \in V. \tag{9}$$

The objective function represents the total gasoline consumption for a trip from O to D. Constraint (4) ensures that there is an outgoing arc in the final path from O. Constraint (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 (6). Constraint (7) guarantees that the battery charge consumption does not exceed the available battery SOC of the vehicle. Constraints (8) ensure that for each pair of nodes at most one vehicle operating mode is selected. Constraints (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 $\mathbf{Y} = [y_{ij}]_{i,j\in V}$.

The minimum gasoline consumption satisfies the triangle inequality given any battery SOC c, where $c \le C$. That is, for all nodes $i, j \in V$ and a battery SOC c, the minimum gasoline consumption from O to j is less than or equal to the sum of the minimum gasoline consumption from O to i and from i to j.

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:

$$\text{Maximize} \quad \sum_{i=1}^{N-1} -w^g_{i,i+1} x_{i,i+1} \tag{10}$$

$$\text{Subject to:} \quad \sum_{i=1}^{N-1} w^e_{i,i+1} y_{i,i+1} \le C, \tag{11}$$

$$x_{i,i+1} + y_{i,i+1} = 1, \quad \forall\, i = 1, \ldots, N-1, \tag{12}$$

$$x_{i,i+1}, y_{i,i+1} \in \{0,1\}, \quad \forall\, i = 1, \ldots, N-1. \tag{13}$$

The objective function represents the total savings of gasoline consumption for the path $P = (1, \ldots, N)$. Constraint (11) guarantees that the battery charge consumption does not exceed the available battery SOC of the vehicle. Constraints (12) ensure that for each arc on the path, only one vehicle operating mode is selected. Constraints (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. To show that 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 F on gasoline consumption, is there a solution such that the total gasoline consumption Γ is at most F?

The first step is to prove that 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 from O to D, battery SOC C, and gasoline consumption F, it is easy to check if $(\mathbf{X}, \mathbf{Y})$ is a solution. This verification involves checking that $(\mathbf{X}, \mathbf{Y})$ is a path from O to D. In addition, it includes computing two sums and comparing the

results against F and C, that is, checking $\sum_{(i,j)\in P} w^g_{ij} x_{ij} \le F$ and $\sum_{(i,j)\in P} w^e_{ij} y_{ij} \le C$. This requires an amount of time linear in the number of arcs in G. Thus, 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 (Kellerer, Pferschy, and Pisinger 2004). 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 of k items, where each item $s_{ij} \in T$ is from one set S_i with a total weight at most C (i.e., $\sum_{i=1}^{k} \sum_{j\in S_i} \sum_{s_{ij}\in T} w_{ij} \le C$), such that the corresponding profit is at least R (i.e., $\sum_{i=1}^{k} \sum_{j\in S_i} \sum_{s_{ij}\in T} v_{ij} \ge R$)?

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 the 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 as for 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 $\sum_{i=1}^{N-1} \sum_{j\in S_i} \sum_{s_{ij}\in T} w_{ij} \le C$ and $\sum_{i=1}^{N-1} \sum_{j\in S_i} \sum_{s_{ij}\in T} v_{ij} \ge -F$. 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 $\sum_{i=1}^{N-1} \sum_{j\in S_i} \sum_{s_{ij}\in T} w_{ij} \le C$ and $\sum_{i=1}^{N-1} \sum_{j\in S_i} \sum_{s_{ij}\in T} v_{ij} \ge -F$. 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. 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 F 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 F?

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 from O to D, battery SOC C, and gasoline consumption F, it is easy to check if $(\mathbf{X}, \mathbf{Y})$ is a solution. This verification involves checking that $(\mathbf{X}, \mathbf{Y})$ is a path from O to D. Then, it includes computing two sums and comparing the results against F and C, that is, checking $\sum_{(i,j)\in P} w^g_{ij} x_{ij} \le F$ and $\sum_{(i,j)\in P} w^e_{ij} y_{ij} \le C$. 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 MCKP-D. We consider an instance of the EERP-D on a given path. Such an instance is the same decision problem as the EERP-PATH-D. As in Theorem 1, we proved that there is such a polynomial time reduction from an instance of MCKP-D to EERP-PATH-D, thus, there exists a polynomial time reduction from an instance of MCKP-D to an instance of EERP-D.

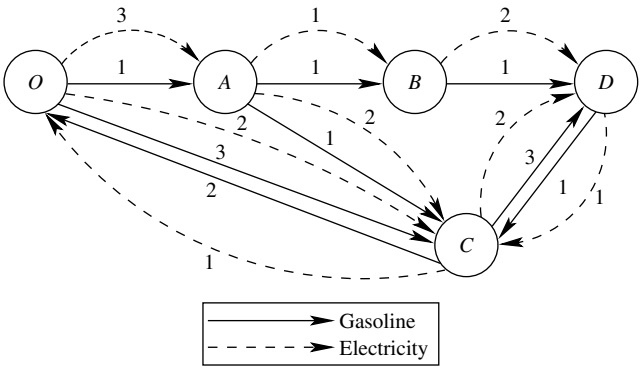
Therefore, EERP-D is NP-complete as its special case EERP-PATH-D is NP-complete. Since EERP-D is the decision version of EERP it follows that EERP is also NP-complete. □

2.1. Illustrative Example

To illustrate the EERP, we present an example shown in Figure 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 electric 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 (Dijkstra 1959) on only the 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

Figure 1. EERP: Illustrative Example of a Fuel Consumption Network



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.

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 paper 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 of the nodes for a specific battery SOC. However, the obtained results do not guarantee the optimality of the solution. This is because of 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 an 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.

3.1. Basic Algorithm: Exact-EER-I

In this section, we propose an exact algorithm, Exact-EER-I, to solve the EERP. Exact-EER-I iteratively increases the SOC and finds the minimum gasoline consumption for that SOC. In each iteration, Exact-EER-I solves the problem for a fixed SOC c in two phases. In the first phase, the algorithm employs a dynamic programming recurrence to find the gasoline consumption to reach a node for a specific battery SOC. In the second phase, it uses the latest updates of the first phase to find the optimal gasoline consumption for that SOC.

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) $c=1,\dots,C$ 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 $(i,j)\in A_R$ that initially employed gasoline mode:

$$\Gamma(j,c)=\min_{(i,j)\in A^e;\, w^e_{ij}\le c}\{\Gamma(i,c-w^e_{ij}),\Gamma(j,c-1)\}. \quad (14)$$

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 arc (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 $\Gamma(j,c)$. This initial value may be reduced in the second phase.

In the second phase of iteration c, Exact-EER-I investigates 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 precomputed optimal All gasoline paths from these promising nodes to j. We denote by $\Delta(i\to j)$ the minimum gasoline consumption between nodes i and j without consuming electricity. Computing $\Delta(i\to j)$, $i,j\in V$, 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 (Johnson 1977). In this phase, Exact-EER-I updates $\Gamma(j,c)$ by considering

only gasoline consumption on the path $(i \to j)$, for all $i \in V$, as follows:

$$\Gamma(j,c) = \min_{i \in V}\{\Gamma(j,c), \Gamma(i,c) + \Delta(i \to j)\}. \quad (15)$$

That is, $\Gamma(j,c)$ is updated for all nodes $i \in V$ 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 (i.e., $\Delta(i \to j)$). The minimum among all these paths from O to j that go through i gives the optimal value of $\Gamma(j,c)$ for SOC c.

Note that Equation (14) does not assume any ordering of the nodes. If gasoline consumption of a node changes in an iteration by Equation (14), the dynamic program in the first phase cannot capture its effect on the gasoline consumption of other nodes. As a result, the recurrence in Equation (14) cannot guarantee optimal results. However, the second phase of the algorithm implementing the recurrence in Equation (15) goes through all of the nodes, and it updates the gasoline consumption of each node considering all of the recent changes in the first phase of this iteration. Even though Equation (15) does not consider any ordering of the nodes, Exact-EER-I finds the optimal gasoline consumption for each node by the end of this phase in each iteration. This is because of the fact that any new changes in gasoline consumption for a node through Equation (15) do not affect the gasoline consumption of any other nodes since these changes are implicitly considered in Δ. Therefore, $\Gamma(j,c)$ in the second phase is independent of any new changes to $\Gamma(i,c)$ for all $i \in V$ in the second phase. As a result, the obtained results of Exact-EER-I are optimal. For example, consider that the optimal gasoline consumption to reach j is obtained by going through a path with gasoline consumption that starts at node h and ends at node j. We assume node i is on this path from h to j, and Exact-EER-I calculates $\Gamma(i,c)$ after finding $\Gamma(j,c)$. In this case, $\Gamma(i,c)$ will be updated based on the gasoline consumption of a path from h to i. The value of $\Gamma(i,c)$ decreases, however, $\Gamma(j,c)$ does not need to consider such a decrease since it has already been considered through the path from h to j. This is a key property of Exact-EER-I, that is, it does not consider any ordering of the nodes while finding optimal solutions.

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, $\Delta(i \to j)$, are precomputed.

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

1: $\Delta(i \to j) \leftarrow$ 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: $\quad\Gamma(j,0) \leftarrow \Delta(O \to j)$
4: $\Gamma(O,0) \leftarrow 0$
5: **for all** $c = 1$ to C **do**
6: $\quad$**for all** nodes $j \in V$ **do**
7: $\quad\quad\Gamma(j,c) \leftarrow \Gamma(j,c-1)$
8: $\quad\quad$**for all** arcs $(i,j) \in A^e$ **do**
9: $\quad\quad\quad$**if** $w^e_{ij} \leq c$ **then**
10: $\quad\quad\quad\quad\Gamma(j,c) = \min\{\Gamma(j,c), \Gamma(i, c - w^e_{ij})\}$
11: $\quad$**for all** nodes $j \in V$ **do**
12: $\quad\quad$**for all** nodes $i \in V$, $i \neq j$ **do**
13: $\quad\quad\quad\Gamma(j,c) = \min\{\Gamma(j,c), \Gamma(i,c) + \Delta(i \to j)\}$
14: $\Gamma^* = \Gamma(D,C)$
15: Find $(\mathbf{X}^*, \mathbf{Y}^*)$ by looking backward
16: Find P^* from $(\mathbf{X}^*, \mathbf{Y}^*)$
17: **Output:** $\Gamma^*, (\mathbf{X}^*, \mathbf{Y}^*), P^*$.

Exact-EER-I starts by setting $\Delta(i \to j)$ to the minimum gasoline consumption between nodes i and j (line 1). Then, it sets $\Gamma(j,0)$ to $\Delta(O \to j)$ 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 $\Delta(i \to j)$.

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 backtracking 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 precomputed all-to-all shortest paths of the gasoline consumption network. 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.

3.1.1. Illustrative Examples of Exact-EER-I. To illustrate how Exact-EER-I works, we present two examples in Figures 2 and 3. We use the same fuel consumption network as in Figure 1. These figures show dynamic programming tables, where there is a row for each node of the network, and a column for each battery SOC $c = 0, 1, 2, 3$. However, each figure shows a different node selection by the Exact-EER-I algorithm.

We first discuss the example presented in Figure 2. 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 because of 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., $\Delta(i \to j)$).

Figure 2. Illustrative Example I: Dynamic Programming Table of Exact-EER-I

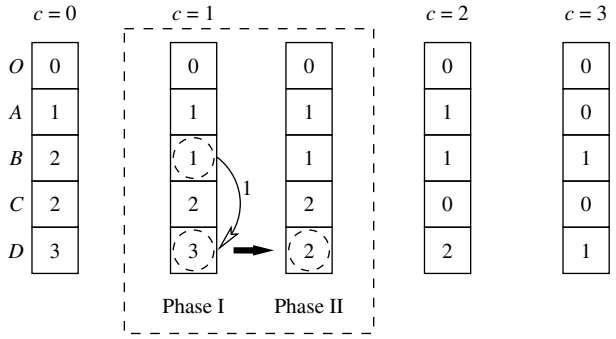
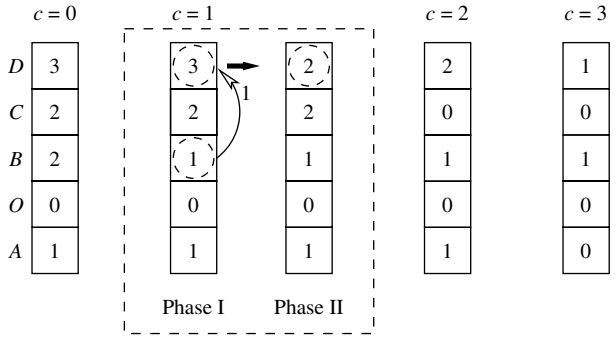


Figure 3. Illustrative Example II: Dynamic Programming Table of Exact-EER-I (with a Different Node Selection)



For brevity, we illustrate the calculations for only the cells that have changed their values for column $c = 1$ (these cells are highlighted by 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 subpath $O \rightarrow B$ from two to one 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 subpath $O \rightarrow B$ (in phase one), and traversing from B to D using all gasoline mode (i.e., $\Delta(B \rightarrow D)$), reducing the cost from three units in column $c = 0$ represented by the path $((O,A)^g, (A,B)^g,(B,D)^g)$ to two 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.

To show that Exact-EER-I does not assume any ordering of the nodes in the fuel consumption network, in Figure 3, we present the example with another node selection by the algorithm. Exact-EER-I initializes the first column of the dynamic programming table (i.e., $c = 0$) with the costs from the precomputed all-to-all shortest paths of the gasoline consumption network (i.e., $\Delta(i \rightarrow j)$). In addition, it sets $\Gamma(O,0)$ to 0. Then, Exact-EER-I iteratively increases the SOC c from 1 to 3 and updates the minimum gasoline consumption function in the table (lines 5–13). Note that Exact-EER-I does not assume any ordering for node selection in lines 6, 8, 11, and 12. In the following, we explain the details. For $c = 1$, the algorithm selects a node $j \in V$ arbitrarily (e.g., D) (line 6), and it sets $\Gamma(D,1)$ to $\Gamma(D,0) = 3$ (line 7). Then, Exact-EER-I analyzes all incoming arcs to D, CD, and BD, to update the table (lines 8–10). None of w^e_{CD} and w^e_{BD} are less than or equal to 1, which is a current value of c. As a result, $\Gamma(D,1)$ does not change. Exact-EER-I selects another node $j \in V$ arbitrarily (e.g., C) (line 6), and it initializes $\Gamma(C,1)$ with $\Gamma(C,0) = 2$ (line 7). None of the incoming arcs to node C can use the 1 unit of available battery SOC to reduce $\Gamma(C,1)$, thus, $\Gamma(C,1)$ remains the same. The algorithm selects another node $j \in V$ arbitrarily (e.g., B) (line 6), and it initializes $\Gamma(B,1)$ with $\Gamma(B,0) = 2$ (line 7). Since w^e_{AB} is 1, Exact-EER-I checks whether using electricity from A to B reduces the gasoline consumption to reach B (line 10). In this case, $\Gamma(B,1)$ is set to $\Gamma(A,0) = 1$. Then, the algorithm selects O (line 6), and $\Gamma(O,1)$ remains 0. Finally, the algorithm selects node A and initializes $\Gamma(A,1)$ with $\Gamma(A,0) = 1$ (line 7). The incoming arc to node A cannot use the 1 unit of available battery SOC, thus, $\Gamma(A,1)$ remains the same. By the end of phase one of iteration $c = 1$, Exact-EER-I investigates the possibility of updating the table (lines 11–13). Since the algorithm can use such an opportunity for just node D during this iteration, we present phase two for node D. When node D is selected, for all other nodes $i \in V$ the algorithm checks if reaching to node i with available battery SOC $c = 1$, and then using gasoline mode from i to D will reduce the total gasoline consumption to reach node D. When node A is selected (line 12), $\Gamma(A,1)$ is 1 and $\Delta(A \rightarrow D) = 2$, which is not less than $\Gamma(D,1) = 3$. Moreover, $\Gamma(D,1)$ remains the same value also when nodes C and O are selected. However, when node B is selected (line 12), $\Gamma(B,1)$ is 1 and $\Delta(B \rightarrow D) = 1$. As a result, a path from O to B with electricity mode with a path from B to D with gasoline mode leads to less total gasoline consumption to reach D. Therefore, $\Gamma(D,1)$ will be updated to 2.

For $c = 2$, the algorithm selects a node $j \in V$ arbitrarily (e.g., B) (line 6), and it sets $\Gamma(B,2)$ to $\Gamma(B,1) = 1$ (line 7). Then, Exact-EER-I analyzes the incoming arc to B to update $\Gamma(B,2)$. However, since $\Gamma(A,1) = 1, \Gamma(B,2)$ remains the same. Exact-EER-I selects node C arbitrarily, and it initializes $\Gamma(C,2)$ with $\Gamma(C,1) = 2$ (line 7).

Then, the algorithm checks all incoming arcs to C (AC, OC, and DC) (lines 8–10). For arc OC, Exact-EER-I reduces $\Gamma(C,2)$ by $\Gamma(O,0)=0$. This means that the algorithm selects traversing arc OC by electric mode leading to the total gasoline consumption 0 from O to C. None of the other nodes can benefit from the additional battery SOC in this iteration. In addition, there is no opportunity to reduce the gasoline consumption in phase two of this iteration, therefore we skip this phase for $c=2$.

For $c=3$, the algorithm selects a node $j \in V$ arbitrarily (e.g., D) (line 6), and it sets $\Gamma(D,3)$ to $\Gamma(D,2)=2$ (line 7). For the incoming arc CD, Exact-EER-I checks $\Gamma(C,3-2)=2$, and it cannot reduce $\Gamma(D,3)$. However, for the incoming arc BD, Exact-EER-I checks $\Gamma(B,3-2)=1$, and then reduces $\Gamma(D,3)$ to 1. Exact-EER-I selects node A with initial $\Gamma(A,3)=1$. Using the incoming arc OA with electricity consumption 3 units, $\Gamma(A,3)$ is updated to 0. None of the other nodes can benefit from the additional battery SOC in this iteration. In addition, there is no opportunity to reduce the gasoline consumption in phase two of this iteration, therefore we skip this phase for $c=3$.

The optimal value of $\Gamma^*(D,3)$ is 1, which represents the path $((O,A)^g,(A,B)^e,(B,D)^e)$.

3.1.2. Properties of Exact-EER-I. In this section, we analyze the properties of the proposed Exact-EER-I. We first discuss the time complexity of Exact-EER-I. Then, we prove that Exact-EER-I finds the optimal solution to the EERP.

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. In general, the capacity C is not bounded by a polynomial in the number of arcs or nodes in G. Since the execution time of the Exact-EER-I algorithm depends on C, Exact-EER-I is a pseudo-polynomial time algorithm. An algorithm has pseudo-polynomial time complexity if its running time is polynomial in the numeric value of the input, but it is exponential in the size of the input (i.e., number of bits required to represent the input).

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

Proof. We prove the theorem by induction. We denote by $\Gamma^*(j,c)$ the optimal solution, for every node j and battery SOC c.

Base case. For $c=0$, $\Gamma(j,0)=\Gamma^*(j,0)$ is the optimal gasoline consumption to reach node j, for all $j \in V$. This is because of the fact that $\Gamma(j,0)=\Delta(O \to j)$, where $\Delta(O \to j)$ is the minimum gasoline consumption from O to j without consuming electricity.

Inductive hypothesis. We assume that for $c=n$, $\Gamma(j,n)=\Gamma^*(j,n)$ is the optimal gasoline consumption to reach node j, for all $j \in V$ given battery SOC n.

Inductive step. We need to prove that for $c=n+1$, $\Gamma(j,n+1)=\Gamma^*(j,n+1)$. We have the following two cases depending on which fuel consumption mode (either gasoline or electric) has been used for traversing the incoming arc to j.

In case one, Exact-EER-I checks all paths for which the incoming arc to j (e.g., (i,j)) is *traversed* by electric mode. Based on lines 8–10, we have $\Gamma(j,n+1)=\min_{(i,j)\in A^e; w_{ij}^e \le n+1}\Gamma(i,n+1-w_{ij}^e)$. Since $n+1-w_{ij}^e \le n$, we have $\Gamma(i,n+1-w_{ij}^e)=\Gamma^*(i,n+1-w_{ij}^e)$ based on the inductive hypothesis. That is, the gasoline consumption to reach node i with SOC $n+1-w_{ij}^e$ has its optimal value in the table. That means, for node j Exact-EER-I checks the optimal path to reach any node i, for all nodes i that have an outgoing arc to j. As a result, the minimum of all optimal gasoline consumption for all nodes i incident to j with SOC $n+1-w_{ij}^e$ is the minimum value. Since Exact-EER-I checks all arcs that reach j in electric mode, and it finds the minimum value among all paths, $\Gamma(j,n+1)$ has the minimum value when traversing the last arc by electric mode. This value can be reduced only in case two, where the incoming arc to j (e.g., (i,j)) is traversed by gasoline mode. This is because of the fact that at this point, only paths for which the incoming arc to j is traversed by gasoline have not been checked.

In case two, Exact-EER-I checks all paths for which the incoming arc to j (e.g., (i,j)) is traversed by gasoline mode. Based on lines 11–13, we have $\Gamma(j,n+1)=\min_{h\in V}\{\Gamma(h,n+1)+\Delta(h \to j)\}$. Exact-EER-I checks all paths from O to j passing through any node h, where h is the first node of a subpath of the path that uses gasoline mode to reach j. If $h=O$, $\Gamma(h,n+1)=\Gamma^*(h,n+1)=0$. Otherwise, node h is the last node whose incoming arc in the path is traversed by electric mode. Based on case one, $\Gamma(h,n+1)=\Gamma^*(h,n+1)$ since the incoming arc to h is traversed by electric mode, and $\Gamma^*(h,n+1)$ does not change during phase two. This is because of the fact that all paths to reach h by traversing its incoming arcs are investigated in phase one. Exact-EER-I checks all such paths from any node $h \in V$ to reach j, and it finds the minimum value of $\Gamma^*(h,n+1)+\Delta(h \to j)$, where $\Delta(h \to j)$ is the minimum gasoline consumption from h to j. Since the triangle inequality holds in the gasoline consumption paths, Exact-EER-I calculates the minimum gasoline consumption from O to j by finding the minimum of all paths that go through any node h. As a result, $\Gamma(j,n+1)$ has the minimum value when traversing the last arc by gasoline mode.

The obtained gasoline consumption $\Gamma(j, n+1)$ is the minimum value of case one and two. Since we check all paths to reach j considering traversing the incoming arc to j by either electric (case one) or gasoline (case two) mode, we have $\Gamma(j, n+1) = \Gamma^*(j, n+1)$.

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

3.2. Improved Algorithm: Exact-EER-II

In this section, 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 precomputing all-pairs shortest paths for the gasoline network, which is a very time consuming task.

Exact-EER-II uses a dynamic programming table Γ with $C+1$ columns and $|V|$ rows, and similar to Exact-EER-I, it iteratively increases the SOC and finds the minimum gasoline consumption for that SOC. In each iteration, Exact-EER-II employs a two-phase approach to solve the problem for a fixed SOC c and updates each column of the dynamic programming table. In the first phase, Exact-EER-II employs the same dynamic programming recurrence used in Exact-EER-I to find the gasoline consumption to reach a node for a specific battery SOC. In addition, it builds a priority queue that will be used for node selection in the second phase. In the second phase, it uses the priority queue to capture the latest updates in the first phase to find the optimal gasoline consumption for that SOC. Exact-EER-II reduces the number of calculations while guaranteeing optimality.

The first phase of iteration c of Exact-EER-II is identical to the first phase procedure of Exact-EER-I for finding the gasoline consumption to reach a node with battery SOC c. In this phase, Exact-EER-II computes $\Gamma(j, c)$ for all nodes $j \in V$ by considering whether using the electric mode on any of the incoming arcs to node j leads to lower gasoline consumption. However, Exact-EER-II gives priorities to nodes to be explored in the second phase. The priority of a node is based on $\Gamma(j, c)$. Exact-EER-II uses a priority queue, $\mathcal{Q}$, to store the gasoline consumptions computed in this phase as node priorities. Priority queue $\mathcal{Q}$ will be used in the second phase of Exact-EER-II when considering paths that use gasoline in traversing the last arc by selecting the node with the minimum gasoline consumption computed in the first phase. Using the priority queue, Exact-EER-II imposes an order on the nodes to be explored for the update process in the second phase to guarantee the optimality of the solution.

In the second phase of iteration c, Exact-EER-II uses the stored values of the priority queue, $\mathcal{Q}$, and the values of the dynamic programming table. The priority queue gives priorities to the nodes to be explored in this phase. A priority is given to a node based on the gasoline consumption computed in the first phase, and it may change during the second phase. The gasoline consumption of a node with a lower priority in $\mathcal{Q}$ can be updated through an All gasoline path from a higher priority node to that node. However, a node with a lower priority in $\mathcal{Q}$ cannot improve the gasoline consumption of a node with a higher priority. This is the reason that Exact-EER-II finds the optimal gasoline consumption of nodes without the need to cycle numerous times. In the second phase, Exact-EER-II extracts the nodes from $\mathcal{Q}$ based on their priority. 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. Exact-EER-II 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 priorities of the remaining nodes in $\mathcal{Q}$ are updated accordingly as follows:

$$\Gamma(k, c) = \min_{(j,k)\in A^g} \{\Gamma(k, c), \Gamma(j, c) + w^g_{jk}\}. \tag{16}$$

That is, $\Gamma(k, c)$ is updated for all nodes $k \in \mathcal{Q}$ that have an incoming arc from j by considering the path with minimum gasoline consumption from O to j, $\Gamma(j, c)$, and then traversing arc (j, k) using only gasoline mode (i.e., w^g_{jk}). Traversing through these arcs (i.e., (j, k)) by gasoline mode forms a gasoline subpath from j to another node (e.g., node h). That means, $\Gamma(h, c)$ is the sum of the minimum gasoline consumption of a path from O to j computed as $\Gamma(j, c)$ and the minimum gasoline consumption of a path from j to h using only gasoline mode (computed by $\sum_{(a,b)\in(j\to h)} w^g_{ab}$ based on updated priorities of nodes).

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

for all $c = 0$ to C **do**
  Create an empty priority queue $\mathcal{Q}$
  for all nodes $j \in V$ **do**
    if $c = 0$ **then**
      $\Gamma(j, c) \leftarrow \infty$
    else
      $\Gamma(j, c) \leftarrow \Gamma(j, c-1)$
    if $j = O$ **then**
      $\Gamma(O, c) \leftarrow 0$
    for all arcs $(i, j) \in A^e$ **do**
      if $w^e_{ij} \le c$ **then**
        $\Gamma(j, c) = \min\{\Gamma(j, c), \Gamma(i, c - w^e_{ij})\}$
    $\mathcal{Q}$.enqueue(j)
    $\mathcal{Q}$.updateKey($j, \Gamma(j, c)$)
  while $\mathcal{Q}$ is not empty **do**
    $(j, \Gamma(j, c)) = \mathcal{Q}$.extractMin( )
    for all arcs $(j, k) \in A^g$ **and** $k \in \mathcal{Q}$ **do**
      $\Gamma(k, c) = \min\{\Gamma(k, c), \Gamma(j, c) + w^g_{jk}\}$
      $\mathcal{Q}$.updateKey($k, \Gamma(k, c)$)
$\Gamma^* = \Gamma(D, C)$

21: Find $(\mathbf{X}^*, \mathbf{Y}^*)$ by looking backward
22: Find P^* from $(\mathbf{X}^*, \mathbf{Y}^*)$
23: **Output:** $\Gamma^*, (\mathbf{X}^*, \mathbf{Y}^*), P^*$.

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 $(i, j) \in A_R$ 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, $\mathcal{Q}$ (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 (priority) associated with it that represents the total gasoline consumption from O to j, $\Gamma(j, c)$, considering the maximum battery SOC c. A node with the smallest key has the highest priority. 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 dynamic programming table. In this phase, Exact-EER-II checks whether using the gasoline mode on any of the outgoing arcs from node j leads to less gasoline consumption. Then, the keys of the other nodes in $\mathcal{Q}$ are updated accordingly. By the end of this iteration, the algorithm updates the values of gasoline consumption by considering all paths to reach all nodes with SOC c. 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 $\Gamma(D, C)$.

3.2.1. Properties of Exact-EER-II. In this section, we analyze the properties of the proposed Exact-EER-II. We discuss the time complexity of Exact-EER-II, and we then prove that Exact-EER-II finds the optimal solution to the EERP.

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 because of the fact that the algorithm builds a table, where the rows are the nodes and the columns are the 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 $O(\sum_{j=1}^{|V|} \log|V| + \text{degree}(j) + 1 \cdot \text{degree}(j)) = O(|A| + |V|\log|V|)$, 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. We prove the theorem by induction. We denote by $\Gamma^*(j, c)$ the optimal solution, for every node j and battery SOC c.

Base case. For $c = 0$, $\Gamma(j, 0) = \Gamma^*(j, 0)$ is the optimal gasoline consumption to reach node j, for all $j \in V$. This is because of the fact that lines 15 to 19 of Exact-EER-II is similar to Dijkstra's algorithm, and it finds the minimum gasoline consumption from O to j by only considering the gasoline consumption network.

Inductive hypothesis. We assume that for $c = n$, $\Gamma(j, n) = \Gamma^*(j, n)$ is the optimal gasoline consumption to reach node j, for all $j \in V$ given battery SOC n.

Inductive step. We need to prove that for $c = n + 1$, $\Gamma(j, n + 1) = \Gamma^*(j, n + 1)$. We have the following two cases depending on which fuel consumption mode (either gasoline or electric) has been used to traverse the incoming arc to j.

In case one, Exact-EER-II checks all paths for which the incoming arc to j (e.g., (i, j)) is *traversed* by electric mode. Based on the inductive hypothesis and similar arguments as in case one of the proof of Theorem 3, $\Gamma(j, n + 1)$ has the minimum value when traversing the last arc by electric mode.

In case two, Exact-EER-II checks all paths for which the incoming arc to j (e.g., (i, j)) is traversed by gasoline mode. In doing so, Exact-EER-II finds $\Gamma(j, c) = \min_{(i,j)\in A^g}\{\Gamma(j, c), \Gamma(i, c) + w_{ij}^g\}$. When node j is extracted from $\mathcal{Q}$ (line 16), Exact-EER-II has already checked all paths to reach node j for which arc (i, j) (for all arcs $(i, j) \in A^g$) is traversed by gasoline mode (lines 17–19), and the minimum gasoline consumption

is stored in $\mathcal{Q}$ (line 19). Therefore, $\Gamma(j, n+1)$ has the minimum value when traversing the last arc by gasoline mode.

The obtained gasoline consumption $\Gamma(j, n+1)$ is the minimum value of cases one and two. Since we check all paths to reach j traversing the incoming arc to j by either electricity (case one) or gasoline (case two) mode, we have $\Gamma(j, n+1) = \Gamma^*(j, n+1)$.

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

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 pseudo-polynomial. In this section, we propose a 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 $P = NP$ (Vazirani 2004).

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 $1/\epsilon$ that satisfies

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

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

All of the FPTASs proposed in the literature are based on dynamic programming formulations (Woeginger 2000). 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 Sahni (1976), 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 (e.g., Mashayekhy, Nejad, and Grosu 2015). The method of reducing the state space was first introduced by Ibarra and Kim (1975). 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 $1/\epsilon$, where ϵ is a given error parameter. To design an FPTAS, we need to design an exact algorithm that uses rounded values and finds the optimal solution on 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 cannot use rounding on electricity consumption. As a result, we need to design an exact algorithm (Exact-EER-GC) that given a gasoline capacity finds the optimal battery consumption. Therefore, in the problem statement we swap electricity and gasoline as objective and constraint. Our FPTAS uses rounding on gasoline consumption and applies Exact-EER-GC to find optimal results on the rounded values. Our proposed FPTAS checks for a value of gasoline capacity whether the obtained optimal battery consumption exceeds the battery SOC C. Based on the result, our proposed FPTAS calculates new upper or lower bound values for the gasoline capacity. This procedure continues until the proposed FPTAS finds close upper and lower bounds on the gasoline consumption leading to ϵ error guarantee. Finally, considering our approach, our proposed FPTAS always satisfies the battery SOC constraint.

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.

4.1. Exact-EER-GC Algorithm

In this section, 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. However, Exact-EER-GC finds a path from O to D given an available gasoline capacity such that the total electricity consumption E is minimized. To use Exact-EER-GC in our FPTAS algorithm, Exact-EER-GC needs to determine the path with the minimum gasoline consumption less than a given gasoline capacity F and with the total battery charge consumption satisfying the battery SOC constraint C. This is because of the fact that FPTAS updates the lower and upper bounds on gasoline consumption using the obtained output of Exact-EER-GC.

Exact-EER-GC employs a dynamic programming table E with $F+1$ columns and $|V|$ rows. We denote by $E(j, f)$ 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 f.

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

$$E(j, f) = \min_{(i,j)\in A^g; w^g_{ij} \leq f} \{E(i, f - w^g_{ij}), E(j, f-1)\}. \quad (17)$$

To consider paths that use electricity in traversing the last arc, Exact-EER-GC uses a priority queue $\mathcal{Q}$ to select the node with the minimum electricity consumption computed in the first phase.

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

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

The node with the smallest total electricity consumption (e.g., j) is always the first node to be extracted from $\mathcal{Q}$ and its $E(j, f)$ 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:

$$E(k, f) = \min_{(j,k) \in A^e} \{E(k, f), E(j, f) + w^e_{jk}\}. \tag{18}$$

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 F. 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 also has an additional input parameter, the gasoline capacity F.

Exact-EER-GC builds a dynamic programming table, where rows are nodes and columns are gasoline consumption (lines 2–23). The algorithm initializes $E(j, f)$ for each node j and gasoline consumption f (lines 5–9). Then, it updates $E(j, f)$ if traversing an arc $(i, j) \in A_R$ 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 $E(j, f)$ 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 f while traversing the last arc (i, j) with gasoline. The obtained gasoline consumption to reach node j is added to a priority queue, $\mathcal{Q}$ (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, $E(j, f)$, considering the maximum gasoline consumption f. 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 $E(j, f)$ is finalized in the dynamic programming 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 $E(D, f)$ satisfies the battery SOC constraint (lines 20–23), if not, E^* has its initial value, which is $+\infty$. Based on the determined value f, Γ^* is set to f (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 $O(F(|A| + |V| \log |V|))$, where $|V|$ is the number of nodes, $|A|$ is the number of arcs, and F 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 F 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.

4.2. FPTAS-EER Algorithm

In this section, 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 $(i,j) \in A^g$ are updated as follows:

$$w_{ij}'^g \leftarrow \left(\left\lfloor \frac{w_{ij}^g |V|}{F\delta} \right\rfloor + 1 \right) \frac{F\delta}{|V|}, \tag{19}$$

where δ is an adaptive factor for the approximation, set to $\sqrt{U/L} - 1$, and F 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: $F = \sqrt{LU/(1+\delta)}$. During the procedure of closing the gap to less than L, δ is calculated based on the new values of L and U.

Equation (19) updates all gasoline consumption weights of the network such that the FPTAS algorithm uses equi-sized units of gasoline consumption (i.e., $F\delta/|V|$). As a result, the dynamic programming table uses these equi-sized units to avoid checking all values that are considered for the columns. The main idea of our approach is to bring the running time of the dynamic programming on the rounded input (i.e., gasoline weights) down to polynomial. Such an approach is called rounding and scaling. The exact solution obtained by Exact-EER-GC has pseudo-polynomial complexity that is proportional to F. However, the scaled version of the problem based on rounding can be solved by Exact-EER-GC optimally in polynomial time. Then, the solution is rounded back to the original gasoline consumption values with a controlled bounded error. Furthermore, we have chosen the rounding to control the error that is introduced by the updates of the gasoline consumption weights in such a way that we can prove that FPTAS-EER will lose only a bounded amount of value.

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.

The algorithm starts with initial values of L and U, and then successively narrows down this range of possible values for the optimal gasoline consumption. This is achieved by calling Exact-EER-GC and narrowing down the range $[L,U]$ based on the results of Exact-EER-GC in each iteration. When L and U are within a constant factor of each other, FPTAS-EER uses the dynamic programming procedure of Exact-EER-GC to obtain the approximation result.

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 the gasoline consumption network in $O(|A| + |V|\log|V|)$.

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: $\quad \delta = \sqrt{U/L} - 1$
5: $\quad F = \sqrt{LU/(1+\delta)}$
6: $\quad$ **for all** arcs $(i,j) \in A^g$ **do**
7: $\quad\quad w_{ij}'^g \leftarrow \left\lfloor \frac{w_{ij}^g |V|}{F\delta} \right\rfloor + 1$
8: $\quad F' \leftarrow \lfloor |V|/\delta \rfloor + 1$
9: $\quad (E, (\mathbf{X},\mathbf{Y}), P)$ $= \text{Exact-EER-GC}(G(V,A'), OD, C, F')$
10: $\quad$ **if** $E \le C$ **then**
11: $\quad\quad U \leftarrow (1+\delta)F$
12: $\quad$ **else**
13: $\quad\quad L \leftarrow F$
14: **for all** arcs $(i,j) \in A^g$ **do**
15: $\quad w_{ij}'^g \leftarrow \left\lfloor \frac{w_{ij}^g |V|}{\epsilon L} \right\rfloor + 1$
16: $F' \leftarrow \lfloor |V|/\epsilon \rfloor + 1$
17: $(\hat{E}, (\hat{\mathbf{X}},\hat{\mathbf{Y}}), \hat{P}) = \text{Exact-EER-GC}(G(V,A'), OD, C, F')$
18: Calculate $\hat{\Gamma}$ based on original value of $\hat{P}$
19: **Output:** $\hat{\Gamma}, (\hat{\mathbf{X}},\hat{\mathbf{Y}}), \hat{P}$.

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 F is updated 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 $w_{ij}'^g$ stored as A', and a gasoline capacity F' as the input parameters (line 9). If there is no feasible solution, then the lower bound is updated to F, otherwise the upper bound is updated to $(1+\delta)F$. Finding a feasible solution by Exact-EER-GC

means that with a given total gasoline consumption F', we have $E(D, F') \le C$.

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).

4.3. Properties of FPTAS-EER

In this section, 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 $1/\epsilon$. We then compare the time complexity of FPTAS and Exact-EER-II.

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

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 $1/\epsilon$.

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 $L \le \Gamma \le U$. If $U \le (1+\epsilon)L$, then Γ is within $1+\epsilon$ of the optimal solution Γ^*. If $U > (1+\epsilon)L$, we can select F such that $L < F < U(1+\epsilon)^{-1}$. Using Exact-EER-GC with selected F can improve the bounds such that either U is decreased to $F(1+\epsilon)$ (line 11 in Algorithm 4) or L is increased to F (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^g based on the determined lower bound L. Since FPTAS-EER replaces the arcs' gasoline consumption by $w'^g_{ij} \leftarrow (\lfloor w^g_{ij}|V|/\epsilon L \rfloor + 1)(\epsilon L/|V|)$, for each arc, we have $|w^g_{ij} - w'^g_{ij}| \le \epsilon L/|V|$. That means, there is an error of at most $\epsilon L/|V|$ for each arc. In the worst case, where the number of arcs in the path is $|V|-1$, the total error is $\epsilon L(|V|-1)/|V|$, which is less than ϵL. Since $L \le \Gamma^*$, we have $\epsilon L < \epsilon\Gamma^*$. Therefore, Γ is within $1+\epsilon$ of the optimal solution Γ^*.

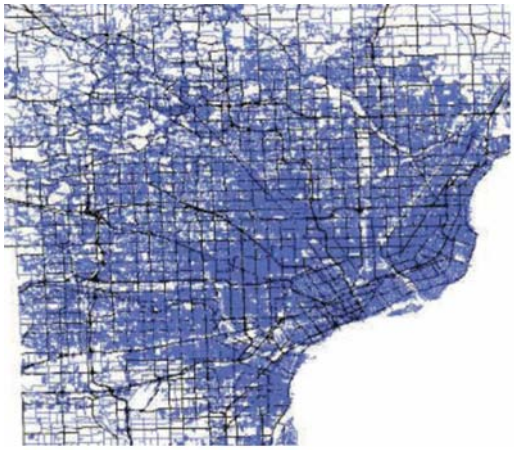
We now show that the time complexity of FPTAS-EER is polynomial in the number of nodes, the number of arcs, and $1/\epsilon$. The time complexity of FPTAS-EER is given by the time complexity of its three major parts. In the first part, the upper bound U is determined by finding the shortest path from O to D on the 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. Even though Exact-EER-GC is a pseudo-polynomial time algorithm because its execution time depends on F, FPTAS-EER calls Exact-EER-GC with parameter $\lfloor |V|/\delta \rfloor + 1$ as the input value instead of F leading to a polynomial time Exact-EER-GC. FPTAS-EER 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 number of iterations is in $O(\log\log(U/L))$ by using δ to reduce the gap. Then, 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 i is in $O(\log\log(U/L))$ and $\sum_i 1/\delta_i$ 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 $|V|/\epsilon < C$. This is derived 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 $|V|/\epsilon \ge C$, 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.

5. Experimental Results

We extract real road network features of Southeast Michigan from data provided by NAVTEQ (2014) using

Figure 4. (Color online) Southeast Michigan Road Network, 465,938 Road Segments (Arcs), 168,806 Cross Sections (Nodes)



ArcGIS Desktop 10.1. Figure 4 shows the full road network of Southeast Michigan. The extracted data from the 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 3.3 GHz with 48 GB RAM. In this section, we describe the experimental setup and analyze the experimental results.

5.1. Generating Multigraph Road Networks

Given the absence of gasoline and battery consumption data for any production PHEV for all of 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 (Davis, Diegel, and Boundy 2012). In particular, we employ the following expression that estimates the gasoline mode fuel economy (mpg) as a function of the 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., priuschat.com 2014). 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.

5.2. Analysis of Results

We compare the performance of Exact-EER-I, Exact-EER-II, and FPTAS-EER on different network sizes. In Section 5.2.1, we present the results for Exact-EER-II on the whole Southeast Michigan network. In Section 5.2.2, we present the results for all of the proposed algorithms on a small network selected from the 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 a reasonable amount of time because of its large memory requirements, and the setting for the Southeast Michigan network is such that $|V|/\epsilon < C$ for the chosen range of C, therefore, as discussed in Section 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 the Southeast Michigan network.

5.2.1. Large Scale Networks. We evaluate the performance of Exact-EER-II on the 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 6 different classes of OD pairs distance shown in Table 2: 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, 1,000 Wh, 2,000 Wh, 3,000 Wh, 4,000 Wh, and 5,000 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, the algorithm selects only the electric mode for all of the arcs in the path to minimize gasoline consumption. In fact, there is no point in minimizing gasoline consumption in this scenario. The amount of gasoline consumption is always zero in this case, and the remaining SOC is the SOC at the beginning of the trip minus the electricity consumption of the path. It is for this reason that we choose smaller available battery SOC for shorter distances. When the OD distance is not within all electric range, the algorithm uses the most or all of the available SOC. In this case, the remaining SOC, at the end of the trip, is either zero or not greater than (or equal to) the electricity consumption of any arc

Table 2. Available Battery SOC in Each Class of OD Distance

Attribute	Value					
OD distance (mile)	0–5	5–10	10–20	20–30	30–40	Above 40
SOC (Wh)	200	1,000	2,000	3,000	4,000	5,000

(i.e., w^e_{ij}) in the optimal path that is traversed by gasoline mode (i.e., $x_{ij} = 1$). Formally, the remaining SOC is bounded by

$$\min_{(i,j) \in P^* \text{ and } x_{ij}=1} w^e_{ij}. \tag{20}$$

That means, if the remaining SOC had been greater than the bound, it would have been used to traverse at least one more arc by electric mode.

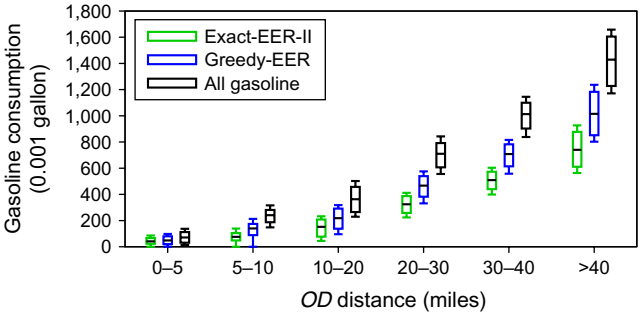
We compare the performance of our proposed algorithm, Exact-EER-II, with that of a greedy based algorithm called Greedy-EER. In addition, we present the minimum gasoline consumption without considering electric mode for the selected OD pairs, called All gasoline. The Greedy-EER algorithm incorporates a greedy approach to solve the EERP as follows. It first finds the route with minimum gasoline consumption without considering the electric mode. The route can be found by using Dijkstra's algorithm on a gasoline consumption network with a time complexity of $O(|A^g| + |V| \log |V|)$. 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. To find an upper bound on the solution of Greedy-EER, we need to find in the worst case how much gasoline savings the solution of Greedy-EER can have by using the battery SOC at the beginning of the trip. An upper bound of Greedy-EER is given by

$$\Delta(O \to D) - \frac{C}{\max_{\forall (i,j) \in P} (w^e_{ij}/w^g_{ij})}, \tag{21}$$

where P is the path with minimum gasoline consumption (i.e., $\Delta(O \to D)$) and w^e_{ij}/w^g_{ij} is the ratio of the electricity consumption to gasoline consumption on an arc with worst electric machine performance. Several production vehicles are known to currently employ this greedy approach. While the 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 5 shows the distributions of the gasoline consumption in 0.001 gallon units obtained by the algorithms. This figure presents the distribution of the results with minimum, 10th percentile, average, 90th percentile, and maximum values. The results show that the range of the gasoline consumption for 0–5 miles and 5–10 miles classes is smaller than other classes. This is because of the fact that these two classes contain OD pairs with difference in distances within 5 miles, whereas the classes of 10–20, 20–30, and 30–40 contain OD pairs with difference in distances within 10 miles. We observe an increase in the range again for the above 40 miles class since this class contains OD pairs

Figure 5. (Color online) Distributions (Minimum, 10th Percentile, Average, 90th Percentile, and Maximum) of the Gasoline Consumption (0.001 Gallon)



with difference in distances within 19 miles. Higher OD distance difference in a class results in a higher range of gasoline consumption. In addition, we observe that the All gasoline algorithm has a higher range of gasoline consumption in all classes compared to Exact-EER-II and Greedy-EER. This is because of the fact that Exact-EER-II and Greedy-EER can utilize the battery SOC for a part of the route and decrease the range of gasoline consumption in a class. The results show that Greedy-EER can be far from the optimal solution provided by Exact-EER-II. With an increase in OD distance and battery SOC, the gap between Greedy-EER and Exact-EER-II increases. This is because of 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 to reduce gasoline consumption.

Figure 6 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 the CPLEX 12 solver cannot find the optimal solution for the EERP for any of the selected OD pairs even after 3,600 seconds. Greedy-EER is a Dijkstra-based algorithm with a time complexity of $O(|A^g| + |V| \log |V|)$. Since the time

Figure 6. (Color online) Execution Time of Exact-EER-II

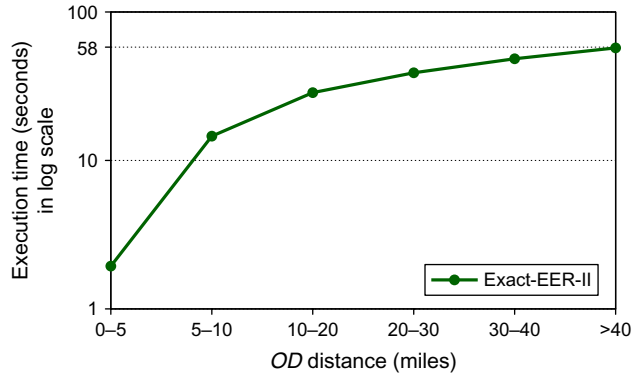


Figure 7. (Color online) Normalized Gasoline Consumption Ratio of Greedy-EER

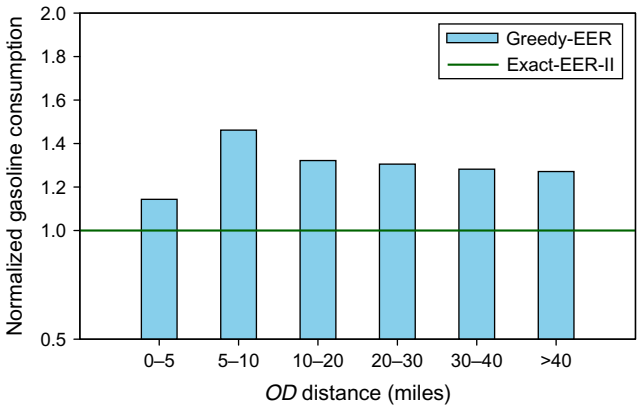
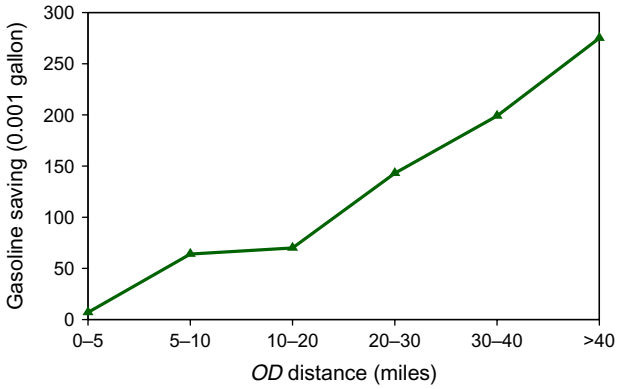


Figure 8. (Color online) Gasoline Savings of Exact-EER-II Compared to Greedy-EER



complexity of Greedy-EER depends only on the number of arcs and nodes and not on the battery SOC C, its execution time is negligible, and we do not present it here.

In Figures 7 and 8, we compare the gasoline consumption obtained by Exact-EER-II and Greedy-EER. Figure 7 shows the normalized gasoline consumption of Greedy-EER and Exact-EER-II over the optimal solution obtained by Exact-EER-II for each OD distance class, where the *normalized gasoline consumption* is the ratio of the gasoline consumption obtained by Greedy-EER and the optimal gasoline consumption (obtained by Exact-EER-II). Note that the normalized optimal results remain the same in all classes. This figure shows how many times larger the gasoline consumption obtained by Greedy-EER is than the optimal result for each class. In the less than five miles class, the ratio is 1.14, which is the lowest ratio among all classes of OD pairs. This is because of the 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 1,000, this ratio increases to 1.46. This is because of 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 8 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.

Figure 9. (Color online) Percentage of Gasoline Savings of Exact-EER-II Compared to All Gasoline

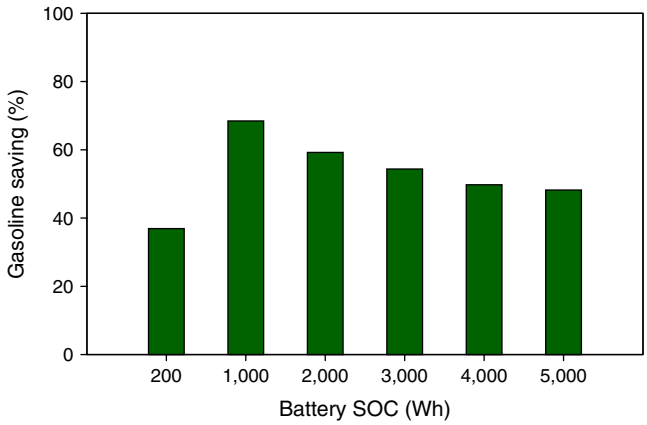


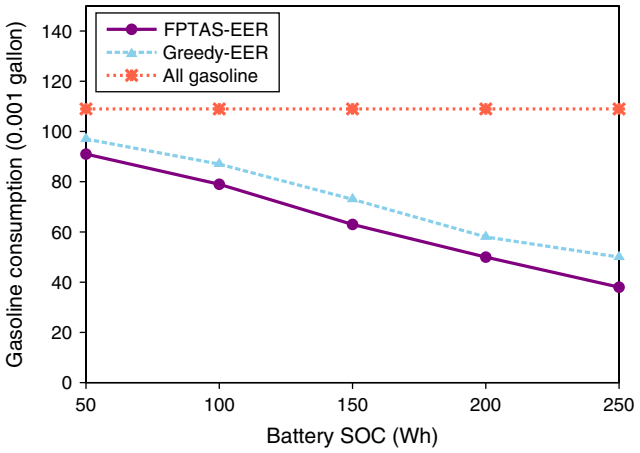
Figure 9 shows the percentage of gasoline saving obtained by Exact-EER-II compared to that of the All gasoline because of the use of battery SOC. This figure shows how much more gasoline could have been consumed if there was no battery SOC at the beginning of the trip. Since there is a significant increase in the amount of battery SOC for the class of 5–10 miles compared to that of 0–5 miles, Exact-EER-II covers more arcs of the path using electricity. In this case, 68.46% of the path is traversed with electricity.

For all of 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.

5.2.2. 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 NAVTEQ (2014) 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 Section 5.2.1), and the minimum gasoline consumption only on the gasoline consumption network, called All gasoline. We randomly select

Figure 10. (Color online) Effect of Battery SOC on the Gasoline Consumption



100 OD pairs by considering several battery SOC between 50,000 to 250,000 mWh. For FPTAS-EER, we set $\epsilon = 0.1$.

We perform sensitivity analysis for the battery SOC parameter to analyze the effects of change in battery SOC on gasoline consumption (Figure 10) and execution time (Figure 11) of our proposed algorithms. Figure 10 shows the average gasoline consumption in 0.001 gallon units obtained by FPTAS-EER, Greedy-EER, and All gasoline algorithms. For all of 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 because of the small range of battery SOC and the size of the network.

Figure 11 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 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 pseudo-polynomial 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.

Figure 11. (Color online) Effect of Battery SOC on the Execution Time

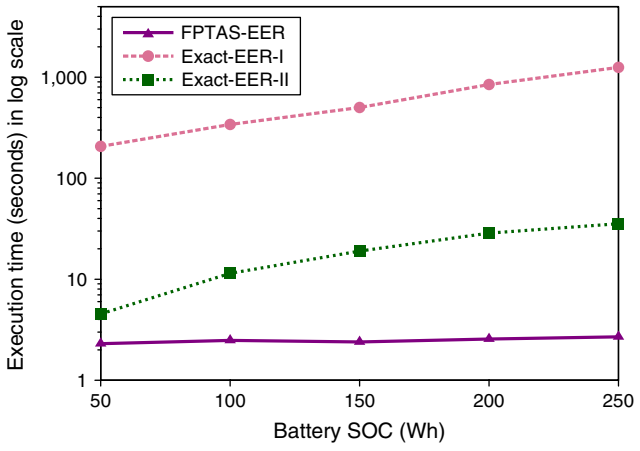
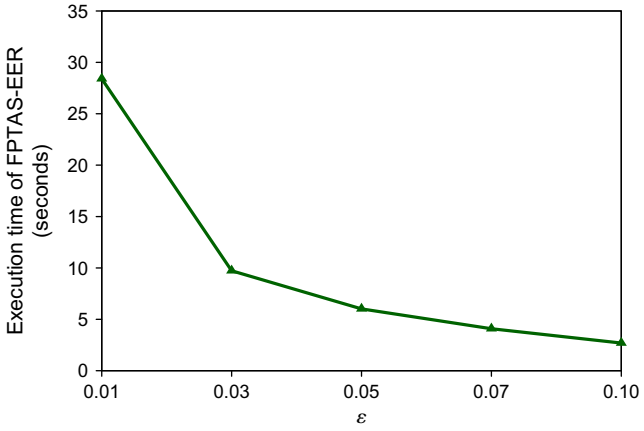


Figure 12. (Color online) Effect of ϵ on Execution Time of FPTAS-EER



We also perform sensitivity analysis for the error bound parameter ϵ. The execution time of FPTAS-EER for values of ϵ between $[0.01, 0.1]$ and battery SOC of 250,000 mWh is presented in Figure 12. The results show that by increasing the error bound, the execution time of FPTAS-EER decreases. In addition, the figure shows that the execution time of FPTAS-EER is polynomial in $1/\epsilon$.

From all of 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.

6. Conclusion

In this paper, we introduced the EERP for PHEVs. We presented the hardness proof of the EERP. We then

proposed two exact pseudo-polynomial 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 multigraph networks. In the context of vehicle routing, this is the first study to take into account the 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 the 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.

It would be interesting to extend this research 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 can be incorporating stochastic features into the problem. 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.

Acknowledgments

The authors would like to thank the editor and the anonymous reviewers for their helpful and constructive suggestions.

References

Ahuja RK, Magnanti TL, Orlin JB (1993) *Network Flows: Theory, Algorithms, and Applications* (Prentice Hall, Englewood Cliffs, NJ).

Almuhtady A, Lee S, Romeijn E, Wynblatt M, Ni J (2014) A degradation-informed battery-swapping policy for fleets of electric or hybrid-electric vehicles. *Transportation Sci.* 48(4):609–618.

Bay M, Limbourg S (2015) TSP model for electric vehicle deliveries, considering speed, loading and road grades. *Proc. 6th Internat. Workshop Freight Transportation Logist., Ajaccio, France*, 439–442.

Bazaraa MS, Jarvis JJ, Sherali HD (2009) *Linear Programming and Network Flows*, 3rd ed. (John Wiley & Sons, Hoboken, NJ).

Bernstein A (2012) Near linear time $(1 + \varepsilon)$-approximation for restricted shortest paths in undirected graphs. *Proc. 23rd Annual ACM-SIAM Sympos. Discrete Algorithms, Kyoto, Japan*, 189–201.

Bertsekas DP (1991) *Linear Network Optimization: Algorithms and Codes* (MIT Press, Cambridge, MA).

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

Davis SC, Diegel SW, Boundy RG (2012) *Transportation Energy Data Book*, 31st ed. (Oak Ridge National Laboratory, Oak Ridge, TN).

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

Di Puglia Pugliese L, Guerriero F (2013b) A reference point approach for the resource constrained shortest path problems. *Transportation Sci.* 47(2):247–265.

Diakonikolas I, Yannakakis M (2009) Small approximate Pareto sets for biobjective shortest paths and other problems. *SIAM J. Comput.* 39(4):1340–1371.

Dijkstra EW (1959) A note on two problems in connexion with graphs. *Numerische Mathematik* 1(1):269–271.

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

Eisner J, Funke S, Storandt S (2011) Optimal route planning for electric vehicles in large networks. *Proc. 25th Assoc. Advancement Artificial Intelligence Conf., San Francisco*, 1108–1113.

Ergun F, Sinha R, Zhang L (2002) An improved FPTAS for restricted shortest path. *Inform. Processing Lett.* 83(5):287–291.

Garey MR, Johnson DS (1979) *Computers and Intractability*, Vol. 174 (Freeman, New York).

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

Glerum A, Stankovikj L, Thémans M, Bierlaire M (2014) Forecasting the demand for electric vehicles: Accounting for attitudes and perceptions. *Transportation Sci.* 48(4):483–499.

Goel A, Ramakrishnan KG, Kataria D, Logothetis D (2001) Efficient computation of delay-sensitive routes from one source to all destinations. *Proc. 20th Annual Joint Conf. IEEE Comput. Comm. Soc., Anchorage, AK*, 854–858.

Gong Q, Li Y, Peng ZR (2008) Trip-based optimal power management of plug-in hybrid electric vehicles. *IEEE Trans. Vehicular Tech.* 57(6):3393–3401.

Hassin R (1992) Approximation schemes for the restricted shortest path problem. *Math. Oper. Res.* 17(1):36–42.

Hybrid Vehicle Timeline (2012) http://www.hybridcenter.org/hybrid-timeline.html.

Ibarra OH, Kim CE (1975) Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM* 22(4):463–468.

Johnson DB (1977) Efficient algorithms for shortest paths in sparse networks. *J. ACM* 24(1):1–13.

Kellerer H, Pferschy U, Pisinger D (2004) *Knapsack Problems* (Springer, Berlin).

Kieckhäfer K, Volling T, Spengler TS (2014) A hybrid simulation approach for estimating the market share evolution of electric vehicles. *Transportation Sci.* 48(4):651–670.

Lin Z (2014) Optimizing and diversifying electric vehicle driving range for U.S. drivers. *Transportation Sci.* 48(4):635–650.

Lorenz DH, Orda A (1998) QoS routing in networks with uncertain parameters. *IEEE/ACM Trans. Networking* 6(6):768–778.

Lorenz DH, Raz D (2001) A simple efficient approximation scheme for the restricted shortest path problem. *Oper. Res. Lett.* 28(5): 213–219.

Lu G, Sadagopan N, Krishnamachari B, Goel A (2005) Delay efficient sleep scheduling in wireless sensor networks. *Proc. IEEE INFOCOM 24th Annual Joint Conf. IEEE Comput. Comm. Soc., Miami*, 2470–2481.

Mak H, Rong Y, Shen ZM (2013) Infrastructure planning for electric vehicles with battery swapping. *Management Sci.* 59(7): 1557–1575.

Mashayekhy L, Nejad M, Grosu D (2015) A PTAS mechanism for provisioning and allocation of heterogeneous cloud resources. *IEEE Trans. Parallel Distributed Systems* 26(9):2386–2399.

Mehlhorn K, Ziegelmann M (2000) Resource constrained shortest paths. Paterson M, ed. *Algorithms–ESA 2000*, Lecture Notes Comput. Sci., Vol. 1879 (Springer-Verlag, Berlin Heidelberg), 326–337.

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

NAVTEQ (2014) http://www.navteq.com/.

Nejad M, Mashayekhy L, Chinnam R, Grosu D (2017) Online scheduling and pricing for electric vehicle charging. *IIE Trans.* 49(2):178–193.

Phillips CA (1993) The network inhibition problem. *Proc. 25th Annual ACM Sympos. Theory Comput., San Diego*, 776–785.

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

priuschat.com (2014) http://priuschat.com/threads/toyota-prius-c-53-mpg-city-46-mpg-hwy-under-19-000.102114/page-15.

Raz D, Shavitt Y (2000) Optimal partition of QoS requirements with discrete cost functions. *IEEE J. Selected Areas Comm.* 18(12): 2593–2603.

Romm JJ, Frank AA (2006) Hybrid vehicles gain traction. *Sci. Amer.* 294(4):72–79.

Sachenbacher M, Leucker M, Artmeier A, Haselmayr J (2011) Efficient energy-optimal routing for electric vehicles. *Proc. Assoc. Advancement Artificial Intell. Conf., San Francisco*, 1402–1407.

Sahni SK (1976) Algorithms for scheduling independent tasks. *J. ACM* 23(1):116–127.

Salmasi FR (2007) Control strategies for hybrid electric vehicles: Evolution, classification, comparison, and future trends. *IEEE Trans. Vehicular Tech.* 56(5):2393–2404.

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

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

Sciarretta A, Guzzella L (2007) Control of hybrid electric vehicles. *IEEE Control Systems* 27(2):60–70.

Sherali HD, Hill JM (2009) Reverse time-restricted shortest paths: Application to air traffic management. *Transportation Res. Part C: Emerging Tech.* 17(6):631–641.

Sioshansi R (2012) OR forum—modeling the impacts of electricity tariffs on plug-in hybrid electric vehicle charging, costs, and emissions. *Oper. Res.* 60(3):506–516.

Sweda TM, Klabjan D (2012) Finding minimum-cost paths for electric vehicles. *Proc. IEEE Internat. Electric Vehicle Conf., Greenville, SC*, 1–4.

Sweda TM, Dolinskaya IS, Klabjan D (2017) Optimal recharging policies for electric vehicles. *Transportation Sci.* 51(2):457–479.

Vazirani VV (2004) *Approximation Algorithms* (Springer-Verlag, New York).

Warburton A (1987) Approximation of Pareto optima in multiple-objective, shortest-path problems. *Oper. Res.* 35(1):70–79.

Woeginger GJ (2000) When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS J. Comput.* 12(1): 57–74.

Xi X, Sioshansi R, Marano V (2013) Simulation–optimization model for location of a public electric vehicle charging infrastructure. *Transportation Res. Part D: Transport Environment* 22(5):60–69.