Optimal Routes for Electric Vehicles Facing Uncertainty, Congestion, and Energy Constraints

by

Matthew William Fontana

B.A., Cornell University (2007)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2013

Author .....
Sloan School of Management
August 9, 2013

Certified by .....
Dimitris Bertsimas
Boeing Leaders for Global Operations Professor
Co-Director, Operations Research Center
Thesis Supervisor

Certified by .....
Thomas Magnanti
Institute Professor
Thesis Supervisor

Accepted by .....
Patrick Jaillet
Dugald C. Jackson Professor
Co-Director, Operations Research Center

Optimal Routes for Electric Vehicles Facing Uncertainty, Congestion, and Energy Constraints

by

Matthew William Fontana

Submitted to the Sloan School of Management
on August 9, 2013, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

There are many benefits of owning a battery electric vehicle, including zero tailpipe emissions, potential independence from oil, lower fuel costs, and the option to recharge the battery at home. However, a significant concern about owning a battery electric vehicle is range anxiety: the fear that the battery will run out of charge before the driver reaches his or her destination. We address range anxiety by providing a robust optimization framework to give drivers confidence that they can reach their destinations in a reasonable amount of time with enough energy in the battery, even when there is uncertainty in travel time and energy consumption on the roads. The robust optimization appropriately incorporates uncertainty without significantly increasing the complexity of the problem. This thesis describes that optimization framework and how to use it on real-world examples to find appropriate routes, with a central part being the application of robust optimization to the problem.

We develop an energy model, an optimization-based formulation using robust optimization, and algorithms to quickly find good routes for battery electric vehicles. The combination of using robust optimization, the A-Star algorithm to find shortest paths, and Lagrangian relaxation allows us to solve the problem in seconds or less. For one example start and destination, our algorithms required less than 2 seconds for each instance (energy consumption limit). In addition, for example trips, we compute a Pareto frontier to illustrate the time-energy tradeoff from driving different routes. We use Lagrangian relaxation to provide lower

bounds and estimates that suggest that our algorithms produce near-optimal solutions. We apply our methodology to example trips in Massachusetts and Michigan to demonstrate its practicality and its potential for real-world use. Future work could continue to improve the modeling accuracy and include algorithmic enhancements to further improve running time, especially for larger networks.

Thesis Supervisor: Dimitris Bertsimas
Title: Boeing Leaders for Global Operations Professor
Co-Director, Operations Research Center

Thesis Supervisor: Thomas Magnanti
Title: Institute Professor

Acknowledgments

I express gratitude to all those who have helped me in so many ways throughout my life. Whether I have known them for years or interacted with them once, each has had a positive influence that remains with me today.

I thank my advisors, Professor Thomas Magnanti and Professor Dimitris Bertsimas, for all of their support and advice. I especially appreciate Professor Magnanti's support from the beginning and his willingness to let me choose whatever research path I felt was best and Professor Bertsimas' keen insights on how to maximize the potential impact of this work. This thesis would not have been possible without them.

I am very grateful to Professor Patrick Jaillet, the third member of my thesis committee, for his suggestions, advice, and support.

I would also like to recognize additional people who helped me with my work. Oleg Gusikhin, Erica Klampfl, Perry MacNeille, Ryan McGee, and Daniel Reich from Ford Motor Company provided a focus on practicality that has helped shape the work in a more meaningful way. This research was supported by a grant from the Ford-MIT alliance. The MIT Geographic Information Systems (GIS) staff helped me with using ArcGIS and GIS data. I appreciate the hard work and support of the Operations Research Center (ORC) co-directors, Professor Bertsimas and Professor Jaillet, and the ORC staff. Andrew Carvalho, Paulette Mosley, and Laura Rose have been very helpful with all of my administrative questions.

I am grateful for the experiences I have shared with people and the friends I have made during my time at MIT. I feel fortunate to have been able to learn from my fellow students in the ORC, both from studying together and from our more informal discussions. I would like to express gratitude to the Tech Catholic Community (TCC) and my friends there. In particular, I value Father Richard Clancy's practicality and wisdom that he shared through our conversations, his homilies, and his scripture studies. I have also benefitted from the different perspectives and viewpoints of my friends outside of the TCC and the ORC.

I cannot thank my family enough for everything they have done for me. They have been a constant source of support for my entire life. I am extremely grateful to my father, Bill, and my mother, Carol, for all of the sacrifices they have made and for everything they have taught me. I feel very fortunate to have been raised by both of them. I am also very grateful for my wonderful brother, Peter, who has been a very loyal and dependable companion.

Finally, I would like to thank God for everything. "In God We Trust."

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There are many benefits to owning a battery electric vehicle, including zero tailpipe emissions, potential independence from oil, lower fuel costs, and the option to recharge the battery at home. President Barack Obama expressed the importance of electric vehicles in his State of the Union Address in 2011, "With more research and incentives, we can break our dependence on oil with biofuels, and become the first country to have a million electric vehicles on the road by 2015" ([50]).

However, sales of electric vehicles have currently been significantly less than the one million goal President Obama set for the year 2015. According to the Electric Drive Transportation Association (EDTA), fewer than 105,000 electric vehicles have been sold from December 2010 to May 2013 ([15]). The EDTA defines *electric vehicles* as plug-in hybrid electric vehicles, range extended electric vehicles, and battery electric vehicles. A *battery electric vehicle* is a car that is powered solely by a battery and does not use gasoline. According to CNBC.com, only 50,000 plug-in

vehicles have been sold since 2011, and "Supporters of electric cars point out that many would-be buyers are still dealing with range anxiety, or the feeling that they are limited to how far they can drive" ([12]). This "range anxiety" is a concern that must be addressed for battery electric vehicles to become a widely-adopted form of transportation.

To mitigate range anxiety, we combine principles from engineering and optimization techniques to provide a routing solution designed for battery electric vehicles that solves quickly, which is a requirement for an in-car GPS system. Our algorithms find fast routes while constraining total energy consumption, and we account for uncertainty in both time and energy using a robust-optimization approach. In addition, we illustrate the practicality of these algorithms using real road network data from the Boston, MA area and a 975,666-arc road network primarily representing part of Michigan.

We expect there to be uncertainty in both the travel time and energy consumption of a given route. The uncertainty could be due to traffic, weather, an accident, the behavior of other drivers, complexities of modeling the battery in the vehicle, and/or other factors. This uncertainty, if not accounted for, could cause a driver to take a route that would require more energy than is available in the battery, resulting in the driver being unable to reach his or her destination. This is probably a cause of range anxiety in potential drivers of battery electric vehicles as well. We incorporate this uncertainty in time and energy using robust optimization.

As the main contribution of this thesis, we provide a practical methodology to find time and energy efficient routes for battery electric vehicles, with the confidence that the drivers will reach their destinations, even given the uncertainty

that is on the roads. This includes developing an energy consumption model for a battery electric vehicle, a robust-optimization-based formulation of the problem, and algorithms to solve the problem quickly, and gathering the data to apply this formulation to real-world examples. We feel that our methodology has potential for real-world use.

First, we describe robust optimization, which is a central technique in this thesis (Section 1.1). Then, we provide a literature review of what has been done to solve various routing problems for electric vehicles (Section 1.2) and various shortest path problems incorporating uncertainty besides using robust optimization (Section 1.3).

Our optimization framework has three main components:

- We model the energy consumption of a battery electric vehicle using principles from engineering, and using analytics to support this model (Chapter 2).

- We use robust optimization to model the optimization problem in a way that is tractable, when augmented with Lagrangian relaxation (Chapter 3).

- We build upon known algorithms and create additional heuristics to provide a practical solution methodology (Chapter 4).

We provide examples of applying our methodology on real road networks to illustrate its effectiveness (Chapter 5). In addition, we provide an extension to the framework that incorporates traffic signals and traffic conditions, based on data , and suggest optimization-based formulations that incorporate acceleration

(the first part of Chapter 6). Finally, we provide suggestions for future work and conclude (the rest of Chapter 6).

1.1 Robust Optimization

Robust optimization accounts for the fact that parameters in an optimization problem might change. In the deterministic case, when we solve an optimization problem, we assume that all of the parameters are fixed. However, Bertsimas et al. [10] write, "Solutions to optimization problems can exhibit remarkable sensitivity to perturbations in the parameters of the problem (demonstrated in compelling fashion in [5]), thus often rendering a computed solution highly infeasible, suboptimal, or both (in short, potentially worthless)." Robust optimization has become a very appealing approach to provide solutions that are feasible even if many of the parameters change (and change significantly). Robust optimization has been applied in fields as far ranging as finance, statistics, supply chain management, and engineering. These applications, as well as theoretical developments in robust optimization, are surveyed in [10].

The general premise of robust optimization is to assume that the parameters have some uncertainty and that an adversary may select the values of the uncertain parameters, within certain restrictions. In a robust optimization approach, an algorithm (or something else) solves the optimization problem and selects a solution, and then, the adversary sees this solution, and chooses the values of the uncertain parameters in attempt to make the solution infeasible or as suboptimal as possible. Perhaps, for example, each of the uncertain parameters are within

a specific interval. A very conservative methodology would be to assume that an adversary will set each of the uncertain parameters to the worst-case value. A more realistic approach is to restrict the adversary to choosing all of the values of the uncertain parameters to be within some set, called an uncertainty set. This is the approach taken by robust optimization, and it has been applied to a wide variety of optimization problems, including linear, quadratic, discrete, and semidefinite optimization [10].

One benefit of the robust optimization approach is tractability. While there are other ways to incorporate uncertainty, they typically increase the complexity of the problem substantially. For example, using random variables to model uncertainty for a shortest path problem increases the complexity from polynomial time to NP-Hard or #P-Hard (see Section 1.3). On the other hand, robust optimization is tractable in the sense that many deterministic problems that can be solved in polynomial time have a robust counterpart that can still be solved in polynomial time. In addition, the robust counterpart can often be solved quickly in practice. Bertsimas and Sim [7, 8, 9] provide examples of problems with robust counterparts that can be solved in polynomial time and computations to illustrate the practicality of the robust optimization approach.

Another benefit of robust optimization is the probabilistic guarantees. Suppose that you would like to model uncertainty as random variables. Even though it might be hard to solve an optimization problem with random variables, a tractable approach can be to use robust optimization and then the probabilistic guarantees from it. Namely, model the problem using robust optimization, solve it, and then use a bound or result to show that the probability that the solution to the robust

optimization problem will be feasible is at least some amount. Some probabilistic guarantees are in Bertsimas and Sim [9] and Bertsimas et al. [10]. While random variables are not explicitly included in the model, they can be included in the model by choosing uncertainty sets that have some of the properties of the random variables.

Given all of the benefits of robust optimization we feel it is a useful approach for our problem of finding routes for battery electric vehicles. First, a good solution depends crucially on protecting from uncertainty, due to the requirement that the vehicle must reach the destination before the battery runs out of charge. In particular, being stuck on the road due to the battery running out of energy is a very severe consequence, people would likely be very willing take a more conservative route in order to prevent even the possibility such a disaster. Second, the robust counterpart of the polynomial-time solvable shortest path problem can be solved in polynomial time (see Bertsimas and Sim [7, 9]). In addition, robust optimization can be very practical for our problem, as we show later in this thesis. The details about how we use robust optimization to model our particular problem is described in Section 3.2, and the algorithms we use to solve it are described in Chapter 4.

1.2 Electric Vehicles and Related Routing Problems

Touati-Moungla and Jost [46] briefly describe some ways combinatorial optimization can contribute to efficient electric vehicle management, and they provide references to some relevant papers. They discuss the energy shortest path prob-

lem: find a route that maximizes a vehicle's state of charge at the destination. Each edge has a energy consumption/recharge value. The battery cannot have less than 0 energy at any time, and the battery can recharge to at most its full state. Artmeier et al. [2], address this problem, giving an $O(n^3)$ algorithm and providing some experimental results based on road network data. Sachenbacher et al. [39] present an improved algorithm, based on A-star search, that has $O(n^2)$ complexity.

Sweda and Klabjan [42] focus on finding minimum-cost paths for electric vehicles, where the vehicle may stop and recharge the battery at any node (at an additional cost). They use a dynamic programming approach to solve the problem over a directed, acyclic network.

Several authors consider vehicle routing problems related to electric vehicles by incorporating some form of energy consumption into an optimization model. Worley et al. [51] propose a discrete integer programming formulation for optimally locating vehicle charging stations and routing vehicles, based on the classic vehicle routing problem framework. Schneider et al. [40] formulate the electric vehicle routing problem with time windows and recharging stations and propose a heuristic to solve it that combines a variable neighborhood search algorithm with tabu search. Mirzaei and Krishnan [32] formulate a location routing problem and incorporate energy considerations of mass due to the vehicle load and rolling resistance and solve this formulation on a small example.

MacNeille et al. [30] use a method to estimate energy consumption for battery electric vehicles using simulation. They also present some qualitative results describing how energy consumption varies based in different driving conditions,

such as the road type, vehicle speed, traffic flow rate, and road grade.

Yu et al. [52] provide a method for identifying various driving patterns based on trip segment clustering, and they apply it to estimate trip energy consumption on an example.

Gonder [20] and Zhang and Vahidi [53] propose methods to determine the best utilization strategy for the battery and internal combustion engine when driving a hybrid electric vehicle to improve fuel economy. Syed et al. [43] propose an advisory system for a hybrid electric vehicle based on fuzzy logic to help drivers change their behaviors to achieve greater fuel efficiency for a predetermined route.

Quigley [37] suggests some high-level strategies for hybrid vehicle control, and Tate et al. [44] use a dynamic-programming approach for hybrid-electric vehicle control. In addition, Gong et al. [21] generate Markov chain models from drive data seeking to understand the relationship between plug-in hybrid electric vehicle performance and velocity statistical properties.

1.3 Stochastic Shortest Path Problems

While we propose using robust optimization to solve shortest path problems under uncertainty, other researchers have used other methods to incorporate randomness into shortest path problems. We prefer the robust optimization approach due to its tractability (see Section 1.1). However, we describe various shortest path problems in which arc weights are random variables: stochastic shortest path problems.

The stochastic shortest path problem is to find the shortest path when the directed graph is given and each arc (i, j) has weight L_{ij}, which is a random variable. The arc weights might not be independent. In [36], all arc weights are assumed to come from discrete random variables, each with finitely many values.

As mentioned in [36], there are 3 kinds of the stochastic shortest path problems:

Shortest Expected Path Problem: In the shortest expected path problem, the traveler has no knowledge about the weights of the network, even while traveling, so the optimal solution is to solve the deterministic shortest path problem using $E[L_{ij}]$ as the weight of arc (i, j).

Expected Shortest Path Problem: In the expected shortest path problem, the arc weights are generated randomly and once the instance is generated, the traveler knows what all of the weights of the arcs are. The problem is to determine the expected shortest-path lengths over all possible instances. Provan [36] states this problem is NP-hard to do, even when the arcs weights only take values of 0 or 1, noting that this problem is a network reliability problem and is surveyed in [4].

Shortest Path With Recourse Problem: In the shortest path with recourse problem, the traveler knows the weights of the arcs are once he or she is near them. That is, if the traveler is currently at node v in the graph, he or she knows the weights of all of the outgoing arcs from v. This problem is to compute the expected shortest path cost as well as choose a traveling policy to minimize the traveler's expected cost of going from the source to the

destination.

Note: In all cases, the directed graph is already known and is not random.

Provan [36] states that there are two ways shortest path with recourse problems are differentiated and describes each one:

1. The dependencies of the arc weights:

   Arc-Independent Weights: Each arc's weight is determined independently of the other arc weights.

   Node-Independent Weights: The weights of each of the outgoing arcs from node i are dependent, but they do not depend on the weights of outgoing arcs from any other node.

   Dependent Weights: The weights of the arcs are dependent on some or all of the other arc weights.

   Markov Weights: The arc weights are associated with a Markov process or Markov chain. Each state is a realization of the weights for all of the arcs, and each time an arc is traversed, the system moves to a different state. This process can be defined for node-independent or arc-independent models as well.

2. Whether or not the arc weights change each time the traveler is adjacent to the arc:

   No Reset: The arc weights are fixed once they are seen and do not change while the graph is being traversed.

Reset: Each arc weight is redrawn from the corresponding distribution (independently of its previous value(s)) each time the traveler visits a node incident to it.

Provan [36] makes several observations:

- The optimal recourse "path" might have cycles (and therefore be a walk), since repeating arcs can be useful to discover previously unknown weights on arcs.
- Under the reset assumption, the dependent and node-independent models regarding the arc weights are identical.
- When the graph is acyclic and weights are arc-independent or node-independent, the reset and no reset assumptions are basically identical.

Provan [36] also outlines the complexity of various versions of shortest path with recourse problems:

- Under the no reset assumption, shortest path with recourse with dependent arc weights is NP-Complete [35]. Provan [36] remarks that the NP-hardness result applies to node and arc-independent weights.
- When the weights are node-independent and arc-independent for either the reset or no reset situations, the shortest path with recourse problem over an acyclic graph can be solved in polynomial time.
- For the no reset situations, the shortest path with recourse problem with arc-independent weights is #P-hard and can be solved in polynomial space [35].

- It is not known whether the node-independent and arc-independent shortest path with recourse problems under the no reset assumption are in NP or not.

- For situations with dependent weights and no reset, Provan shows that shortest path with recourse problem is NP-complete for an acyclic graph with arc weights that are either 0 or 1 (the decision problem is to decide whether a recourse path with weight 0 exists) [36].

- The shortest path with recourse problem with reset is solvable in polynomial time if cycles are allowed. This is Provan's main contribution in [36]. However, if we require that the optimal solution is actually a path (without cycles), then the reduction from [35] shows that shortest path with recourse under the reset assumption is NP-hard.

- When there are Markov weights, the shortest path with recourse problem with no reset is NP-hard.

Under the no reset assumption, Polychronopoulos and Tsitsiklis [35] observe that although the graph is assumed fixed, we can incorporate random arc or node failures into the model. For example, use a very large arc weight to represent failure of an arc.

In the proof of Theorem 5 in [35] Polychronopoulos and Tsitsiklis show that the shortest path with recourse problem with dependent arc weights and the no reset assumption is at least as hard as the expected shortest path problem.

We can use very large arc weights and the transformation above to model the expected shortest path problem over a randomly generated graph as a shortest

path with recourse problem with dependent arc weights and the no attachment assumption. However, it might be very complicated to calculate the probabilities that edges are present for the randomly generated graph.

Kulkarni, sometimes with Adlakha, wrote papers, [26, 27, 28, 29], that solve different network optimization problems over directed graphs when the weights are independent and exponentially distributed (although the means of the exponential distributions can vary for each edge). Their work on the shortest path problem is [26]. The general approach is the following: take advantage of the memoryless property of exponential distributions and create a continuous time Markov chain (also known as a Markov process), and then use techniques for analyzing Markov processes to compute quantities of interest. For example, the expected length of a shortest path is the expected time until absorption in the appropriate Markov process. Janson [25] has also used this approach. In addition, Bailey claims that the extension of the Markov process to phase-type arc weights is direct ([3]). Ball [4] states that we should see [3] for a unified framework about this topic.

Chapter 2

Physics and Data

In this chapter, we propose an energy model, based on standard formulas from physics and engineering, to use as a base for an optimization framework (Section 2.1). Then, we apply regression to some vehicle drive data, which will show that our proposed energy model is a reasonable choice (Section 2.2).

2.1 Energy Model: An Engineering-Based Approach

We propose an energy model, E, when traveling over a straight road of length ℓ at speed v in a time interval of length Δt. The standard formulas from physics and engineering that we use can be found in various sources about electric vehicles, including Haddoun et al. [22], Husain [24] and Ehsani et al. [14].

The energy model is

$$E = \frac{v\Delta t}{\eta}\left(\frac{1}{2}\rho C_w A_f v^2 + \mu mg\cos(\alpha) + mg\sin(\alpha) + m\delta\frac{\Delta v}{\Delta t}\right) + P_{\text{acc}}\Delta t, \qquad (2.1)$$

which consists of accessory loads ($P_{\text{acc}}\Delta t$) and the following forces:

- Aerodynamic Drag $= \frac{1}{2}\rho C_w A_f v^2$, which is due to the resistance of air when the vehicle is moving (we assume a headwind velocity of 0).
- Rolling Resistance $= \mu mg\cos\alpha$, which is resistance where the tires meet the road surface.
- Climbing/Downgrade Resistance $= mg\sin\alpha$, which is the force required to go uphill/downhill.
- Acceleration/Deceleration $= m\delta\frac{\Delta v}{\Delta t}$, which is the force required to change the speed of the vehicle.

Climbing/Downgrade resistance and acceleration/deceleration can be negative.

The parameters and variables in (2.1) are:

- $\rho =$ air density (kg/m^3)
- $C_w =$ coefficient of drag (dimensionless)
- $A_f =$ frontal area of the car (m^2)
- $v =$ the velocity of the vehicle (m/sec)
- $\mu =$ coefficient of friction (dimensionless)
- $m =$ mass of the vehicle (kg)
- $g =$ gravitational constant (m/sec^2)
- $\alpha =$ the angle of the road ($-\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2}$, in radians)

- δ = a mass factor (dimensionless). For example, in [49], $\delta = \frac{4I_w}{r_w^2 m}$, where I_w is the polar moment of inertial of a wheel, and r_w is the effective wheel rolling radius
- Δt = the change in time (sec)
- Δv = the change in velocity (m/sec)
- η = an efficiency parameter to account for all complexities of the battery and various losses when transferring energy from the battery to the wheels (dimensionless)
- P_{acc} = the power required to run the the accessory loads, which include heating or air conditioning, headlights, and the radio (W)

The energy model (2.1) includes some simplifying assumptions, such as lossless regenerative braking. Regenerative braking allows an electric vehicle to recapture most of the energy that is lost in conventional (gasoline) vehicles when the brakes are applied. Regenerative braking is very efficient, as specified in a description on how electric vehicles work, "The Regenerative Braking System in an all-electric vehicle will be designed to capture over 90 percent of the energy normally lost and send it back to the battery pack to be stored for later use" ([18]).

2.2 Calibrating the Model Using Regression

We calibrate the energy consumption model using regression, by processing some vehicle drive data from a particular car into segments and applying linear regression to estimate the appropriate coefficients.

We acquired 2 weeks of drive data from a Focus Electric battery electric vehicle driven in California ([17]). The data includes readings every second for energy consumed, vehicle speed, and GPS coordinates. We combine this information with elevation data from the US Geological Survey National Elevation Dataset (obtained from the National Map Viewer, `http://nationalmap.gov/viewer.html`), which we match to the drive data using the latitude and longitude coordinates from the GPS readings. In addition, we aggregate data into "segments" where a vehicle is either accelerating or moving at constant speed.

Recalling the forces used in the engineering-based model, (2.1), we can apply linear regression to estimate the coefficients β_0, β_1, β_2, β_3, and β_4, in

$$E = \beta_0 + \beta_1 v^3 \Delta t + \beta_2 \cos(\alpha) v \Delta t + \beta_3 \sin(\alpha) v \Delta t + \beta_4 \frac{\Delta v}{\Delta t} v \Delta t. \tag{2.2}$$

For 12 hours and 40 minutes of data (4656 segments), we obtained an r-squared value of 0.92 and the estimates shown in Table 2.1, where all of the coefficients have statistically significant t-values.

Coefficient	Estimate	t-Value
β_0	11386	8
β_1	0.366	20
β_2	287	17
β_3	13916	107
β_4	1955	115

Table 2.1: The estimated values and t-values for the coefficients for the linear regression, on (2.1), applied to the vehicle drive data.

The high correlation coefficient and statistically significant t-values indicate

that the energy model provides a good fit for the data. However, the beta coefficients will depend on the particular vehicle.

Besides using this method to estimate the energy formula coefficients directly, we can see how well the parameters from a particular vehicle match the model. One could translate the coefficients in the regression model (2.2) to the vehicle parameters in the energy model (2.1). This would result in solving 5 equations with 8 unknowns, although some of the solutions might be fairly unrealistic. A better way to see how well the vehicle parameters match is to apply linear regression to the energy model

$$E = \begin{pmatrix} \beta_0 + \beta_1 \frac{1}{2\eta} \rho C_w A_f v^3 \Delta t + \beta_2 \frac{\mu m g}{\eta} \cos(\alpha) v \Delta t \\ + \beta_3 \frac{mg}{\eta} \sin(\alpha) v \Delta t + \beta_4 \frac{m\delta}{\eta} \frac{\Delta v}{\Delta t} v \Delta t \end{pmatrix} \tag{2.3}$$

and see how close β_1, β_2, β_3, and β_4 each are to 1.0, where we input proprietary values for C_w, A_f, ρ, m, μ, δ, and η into the equation (g is a standard constant). It is the same energy model as (2.2), and therefore, the quality of the fit from the regression will be the same. The only things that will change are the values of the coefficients β_1, β_2, β_3, and β_4 (β_0 will remain the same).

The 95% confidence interval for the smallest of the estimated values of β_1, β_2, β_3 and β_4 is $[0.675, 0.701]$, and the 95% confidence interval for the largest of the estimated four values is $[1.813, 2.295]$. This indicates that the vehicle values are reasonable, but are not a perfect fit to the model, as the values β_1, β_2, β_3 and β_4 are close to 1.0. If the completely accurate and realistic energy model was exactly equation (2.3), then β_1, β_2, β_3 and β_4 would all be 1.0. However, this

model is a simplification, and therefore we expect there to be some variation in the coefficients from the regression. The value for β_0 would be interpreted as the estimated energy cost (say in accessory loads) per segment. Since the mean segment time for the trip in the dataset is 9.8 seconds, this results in an accessory load (P_{acc}) of 1162 W, which is a reasonable value.

A regression model with a more clear interpretation of β_0 than the model (2.2) would be to fit energy per unit time, using the same vehicle data:

$$\frac{E}{\Delta t} = \begin{pmatrix} \beta_0 + \beta_1 v^3 + \beta_2 \cos(\alpha) v \\ + \beta_3 \sin(\alpha) v + \beta_4 \dfrac{\Delta v}{\Delta t} v \end{pmatrix}. \tag{2.4}$$

In this case, β_0 would be the value of P_{acc} to use in the problem.

The r-squared value of 0.64 and the negative estimated value of β_2 in Table 2.2 indicate that the regression model (2.4) is not as well-suited to the data (the data was processed to be in segments of varying lengths of time).

Coefficient	Estimate	t-Value
β_0	2020	71
β_1	0.0645	25
β_2	-116	-43
β_3	672	46
β_4	137	62

Table 2.2: The estimated values for the coefficients for the linear regression, on (2.4), applied to the vehicle drive data.

Chapter 3

Optimization Model

For the optimization model we use the energy function

$$E_{ij}(v_{ij}) = \frac{\ell_{ij}}{\eta}\left(\frac{1}{2}\rho C_w A_f v_{ij}^2 + \mu mg\cos(\alpha_{ij}) + mg\sin(\alpha_{ij})\right) + P_{\text{acc}}\left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij}\right). \quad (3.1)$$

In equation (3.1), we exclude acceleration from model (2.1) to eliminate the dependence between energy consumption and changes in velocity, since we expect the total energy consumption from acceleration on a route to be small because of the high efficiency of regenerative braking. We show, under some simplifying assumptions, that there is zero net energy consumption due to acceleration over a path (Theorem 3.1). Besides excluding acceleration, equation (3.1) uses the relationship between arc length, ℓ_{ij}, and velocity times time in transit, both directly and through τ_{ij}, which is an extra cost in time for arc (i, j), to account for stopping or turns.

Theorem 3.1 (**Ignoring Acceleration Under Some Assumptions**). Suppose that we make the following assumptions regarding acceleration in $[0, T]$:

- The vehicle starts at velocity 0.
- The vehicle ends at velocity 0.
- At any instant of time, the vehicle can do one of three things:
  1. Remain at a constant velocity
  2. Increase velocity (accelerate) at a specified rate $a > 0$
  3. Decrease velocity (decelerate) at the rate $-a < 0$

  and the vehicle changes whether it accelerates, decelerates, or remains at a constant speed a finite number of times (technically, the number of changes in vehicle acceleration only has to be a set of measure 0).
- Regenerative braking is always 100% effective.

Under these assumptions, for any path and any choice of speeds, the total energy consumed due to acceleration is 0.

Note that the velocity of the vehicle, $v(t)$, is allowed to be 0 at more times than just the start and the end of the trip.

Proof of Theorem 3.1. A sketch of the proof is in Figure 3.1.

Suppose that for any fixed path and any choice of velocities, $v(t)$, the vehicle

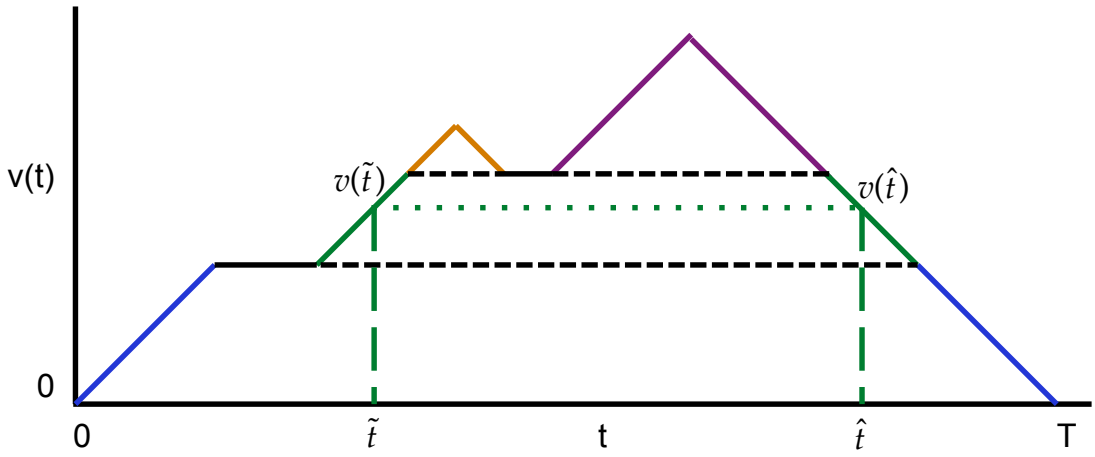

Figure 3.1: A Sketch of the Proof of Theorem 3.1: The vehicle spends energy when accelerating, but will regain it back later when decelerating from the same speed. For example, the vehicle is accelerating at time $\tilde{t}$ and the matched time it is decelerating is time $\hat{t}$.

starts at time $t = 0$ and ends at time $t = T$. Then the total energy consumed is

$$E = \int_0^T \left(\frac{v(t)}{\eta} \left(\begin{array}{l} \frac{1}{2}\rho C_w A_f v(t)^2 + \mu mg\cos(\alpha(t)) \\ + mg\sin(\alpha(t)) + m\delta\frac{dv(t)}{dt} \end{array} \right) + P_{\text{acc}} \right) dt$$

$$= \left(\begin{array}{l} \int_0^T \left(\frac{v(t)}{\eta} \left(\begin{array}{l} \frac{1}{2}\rho C_w A_f v(t)^2 + \mu mg\cos(\alpha(t)) \\ + mg\sin(\alpha(t)) \end{array} \right) + P_{\text{acc}} \right) dt \\ + \int_0^T \left(m\delta v(t)\frac{dv(t)}{dt} \right) dt \end{array} \right).$$

Given our assumptions, $v(t)$ is technically not differentiable everywhere within

$[0, T]$, and we will replace $\frac{dv(t)}{dt}$ with the more accurate notation

$$f_a(t) = \begin{cases} a & \text{if the vehicle is accelerating at time } t \\ 0 & \text{if the vehicle is neither accelerating nor decelerating at time } t \\ -a & \text{if the vehicle is decelerating at time } t \end{cases}$$

Therefore, proving the desired result means showing that

$$\int_0^T (v(t) f_a(t))\, dt = 0,$$

since

$$\int_0^T \left(m\delta v(t) \frac{dv(t)}{dt} \right) dt = \int_0^T (m\delta v(t) f_a(t))\, dt = m\delta \int_0^T (v(t) f_a(t))\, dt.$$

Because the vehicle must accelerate, remain at constant speed, or decelerate, $v(t)$ is continuous and $v(t)f_a(t)$ is bounded and a continuous function of t except at a finite set of points (or a set of measure 0). Therefore, $v(t)f_a(t)$ is Riemann integrable.

The rest of the proof is as follows. For any time $\tilde{t}$ where the vehicle is currently at speed $\tilde{v} = v(\tilde{t})$ and is accelerating at rate a, we are going to "match" $\tilde{t}$ with the first $\hat{t} > \tilde{t}$ where $v(\hat{t}) = v(\tilde{t})$ and the vehicle is decelerating at time $\hat{t}$ (and show that such a $\hat{t}$ exists). Then, we will partition time into small enough intervals, assign each of the matched velocities to the appropriate interval, and use the definition of Riemann integrability to show $\int_0^T (v(t)f_a(t))\, dt = 0$ (the interval

containing time $\tilde{t}$ will cancel out the interval containing time $\hat{t}$).

Consider any time $\tilde{t}$ where the vehicle is currently at speed $\tilde{v} = v(\tilde{t})$ and is accelerating at rate a. Because $v(t)$ is a continuous function and $v(t)$ is increasing at time $\tilde{t}$, for some small $\epsilon > 0$, there is a $\delta > 0$ such that $v(\tilde{t} + \delta) = \tilde{v} + \epsilon > \tilde{v}$. Since $v(\tilde{t} + \delta) = \tilde{v} + \epsilon > \tilde{v}$, $v(T) = 0$, and $v(t)$ is continuous, by the intermediate value theorem, there exists at least one t^* in $[\tilde{t} + \delta, T]$ such that $v(t^*) = \tilde{v}$. Pick $\hat{t}$ to be the smallest t^* in $[\tilde{t} + \delta, T]$ that has $v(\hat{t}) = \tilde{v}$.

Because $\hat{t}$ is the first time after time $\tilde{t}$ the vehicle reached speed $\tilde{v}$, the vehicle must have been decelerating at time $\hat{t}$. If the vehicle was neither decelerating or accelerating at $\hat{t}$, then there would be a $\delta > 0$ such that $v(\hat{t} - \delta) = \tilde{t}$ (as the rate of change at of v(t) at $\hat{t}$ would have been 0), contradicting the fact that $\hat{t}$ is the first time after time $\tilde{t}$ the vehicle reached speed $\tilde{v}$. If the vehicle was accelerating at time $\hat{t}$, then $v(t)$ is increasing at time $\hat{t}$, and for some small $\epsilon_2 > 0$, there is a $\delta_2 > 0$ such that $v(\hat{t} - \delta_2) = \tilde{v} - \epsilon_2 < \tilde{v}$. Since $v(\hat{t} - \delta_2) = \tilde{v} - \epsilon_2 < \tilde{v}$, $v(\tilde{t} + \delta) = \tilde{v} + \epsilon > \tilde{v}$, and $v(t)$ is continuous, by the intermediate value theorem, there exists at least one t^* in $[\tilde{t} + \delta, \hat{t} - \delta_2]$ such that $v(t^*) = \tilde{v}$, which contradicts the fact that $\hat{t}$ is the first time after time $\tilde{t}$ the vehicle reached speed $\tilde{v}$. This argument also shows that there is no other t in $(\tilde{t}, \hat{t})$ with $v(t) = \tilde{v}$.

Using the previous argument, we can effectively pair each time the vehicle is at speed $v(\tilde{t})$ and is accelerating to a time $\hat{t}$ where the vehicle is decelerating and at speed $v(\tilde{t})$, and this way of pairing up times ensures that no time t^* is matched more than once. We can also use the same argument to pair any time $\hat{t}$ where the vehicle is at speed $v(\hat{t})$ and is decelerating to a time $\tilde{t} < \hat{t}$ where the vehicle is accelerating and $v(\tilde{t}) = v(\hat{t})$.

Partition time into intervals $[t_{k-1}, t_k)$ with maximum size $\Delta t = \max_{k=1,\ldots,K} \{t_k - t_{k-1}\}$ and represent $\int_0^T (v(t)f(t))\,dt$ as (using the definition of the Riemann integral)

$$\int_0^T (v(t)f_a(t))\,dt = \lim_{\Delta t \to 0} \sum_{k=1}^{K} (v(\tilde{t}_k)f_a(\tilde{t}_k)\,(t_k - t_{k-1}))$$

where:

- K is the number of intervals of the chosen partition of $[0, T]$.
- $\tilde{t}_k$ is a particular time in the interval $[t_{k-1}, t_k)$.

For any partition chosen to compute the integral $\int_0^T (v(t)f_a(t))\,dt$, select a refinement (a finer sub-partition) where for each $v(\tilde{t})$ where $f_a(\tilde{t}) = a$, there is another interval containing $\hat{t} > \tilde{t}$ where $v(\hat{t}) = v(\tilde{t})$ and $f_a(\hat{t}) = -a$ (we can find the particular $\hat{t}$ by the previous argument that paired times). Select the refinement in this way with the additional requirement that all intervals are of the same length Δt (we can do this for small enough Δt). For convenience in writing out the sum corresponding to the Riemann integral $\int_0^T (v(t)f_a(t))\,dt$, let v_r be the r^{th} speed and n_r be the number of intervals where $v(\tilde{t}_k) = v_r$ and the vehicle is accelerating, for $r = 1, \ldots, R$. By the previous argument that paired times when the vehicle was accelerating and decelerating, there will be n_r times (and therefore

n_r intervals) where the vehicle is at speed v_r and is decelerating. Therefore,

$$\begin{aligned}\int_0^T (v(t)f_a(t))\,dt &= \lim_{\Delta t \to 0} \sum_{k=1}^{K} (v(\tilde{t}_k)f_a(\tilde{t}_k)\,(t_k - t_{k-1})) \\ &= \lim_{\Delta t \to 0} \sum_{r=1}^{R} (n_r v_r a \Delta t - n_r v_r a \Delta t) \\ &= \lim_{\Delta t \to 0} \sum_{r=1}^{R} (0) = 0. \end{aligned}$$

□

As a first step to incorporate the energy model (3.1) into the optimization framework, we discuss a deterministic optimization-based model (Section 3.1.). Then, we expand the deterministic model to incorporate uncertainty using robust optimization (Section 3.2). Next, we apply Lagrangian relaxation to the optimization model (Section 3.3). Finally, we present a result that shows the equivalence between two robust optimization formulations (Section 3.4).

3.1 Deterministic Framework

Our deterministic optimization model is

$$
\begin{aligned}
\min_{v_{ij}, x_{ij}} \quad & \sum_{(i,j)\in A} \left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij} \right) x_{ij} \\
\text{s.t.} \quad & \sum_{(i,j)\in A} \big(E_{ij}(v_{ij}) \big)\, x_{ij} \le E_0 \\
& \underline{v}_{ij} \le v_{ij} \le \overline{v}_{ij}, \quad \forall (i,j) \in A \\
& \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} 1 & \text{if } i \text{ is the source node} \\ -1 & \text{if } i \text{ is the destination node} \\ 0 & \text{otherwise} \end{cases} \\
& x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A.
\end{aligned}
\tag{3.2}
$$

A is the set of arcs in the network, $E_{ij}(v_{ij})$ is the energy function (3.1), and E_0 is the maximum energy consumption, which could be taken or estimated from the vehicle's current state of charge.

We will also write "$X =$ set of paths" to represent the constraints for the shortest path polytope (and requiring x_{ij} to be binary):

$$
\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} 1 & \text{if } i \text{ is the source node} \\ -1 & \text{if } i \text{ is the destination node} \\ 0 & \text{otherwise} \end{cases}
$$

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

Problem (3.2) is a constrained shortest path problem with the additional requirement of finding velocities, and therefore, we expect it to have many similarities to the constrained shortest path problem. We prove that problem (3.2) is NP-Hard in Section 3.1.2.

3.1.1 About Constrained Shortest Path Problems

There is quite a bit of literature about the constrained shortest path problem, although we will not survey it here. The main facts about the constrained shortest path problem are:

- It is NP-Complete (Garey and Johnson [19]). It is a reduction from partition, and Garey and Johnson cite a private communication with N. Megiddo in 1977 as the reference for this.
- It can be solved exactly in pseudopolynomial time, such as with dynamic programming. One source is [23].
- It has a fully-polynomial time approximation scheme (FPTAS) [23].

The algorithms given in Hassin [23] are for acyclic graphs, but the authors state that the generalization to general graphs is straightforward.

Formally, the decision version of constrained shortest path problem can be stated as the following.

Definition 3.1 (**Constrained Shortest Path Problem (Decision Version)**). An instance of the decision version of the *constrained shortest path problem* is the following:

Input: A directed graph $G(N, A)$, with a source node and sink node, arc costs c_{ij}, and arc times t^c_{ij}. There is a cost limit C and time limit T.

Output: A path from the source and sink node such that the total cost is no more than C and the total time is no more than T, or a statement that no such path exists.

■

3.1.2 NP-Hardness of the Deterministic Electric Vehicle Routing Problem

Problem (3.2) is NP-Hard, as it is a generalization of the NP-Hard constrained shortest path problem ([19]). Since any constrained shortest path problem can be transformed into problem (3.2): set $\underline{v}_{ij} = \overline{v}_{ij}$, for all $(i, j) \in A$, and choose the data and parameters appropriately.

Here are the formal details for the NP-hardness.

The decision version of the electric vehicle routing problem is in Definition 3.2.

Definition 3.2 (Electric Vehicle Routing Problem (Decision Version)). An instance of the decision version of the *electric vehicle routing problem* is the following:

Input: A directed graph $G(N, A)$, with a source node and sink node, arc lengths ℓ_{ij}, extra times τ_{ij}, energy consumption functions $E_{ij}(v_{ij})$ (per arc) with the appropriate input constants, and lower and upper bounds on the velocities: $\underline{v}_{ij}$ and $\overline{v}_{ij}$. There is a time limit T and energy consumption limit E_0.

Output: A path from the source and sink node and arc speeds v_{ij} such that the total energy consumption is no more than E_0, the total travel time is no more than T, and v_{ij} are within the specified velocity bounds, or a statement that no such path and velocities exist.

■

Theorem 3.2. The decision version of the electric vehicle routing problem (Definition 3.2) is NP-Complete.

Proof. Proof of Theorem 3.2

The decision version of the electric vehicle routing problem (Definition 3.2) is in NP, as feasible solutions can be verified in polynomial time. Namely, a path that requires a travel time at most T and energy consumption at most E_0, with the specified velocities can be described in at most $|A|$ variables for the path and at most $|A|$ variables for the velocities. The travel time and energy consumption of the path with velocities can be checked in polynomial time. In addition, we can check that the velocities are within the specified bounds and that the solution is a path from the source node to sink node in polynomial time.

Consider an instance of the constrained shortest path problem (a graph $G(N, A)$, costs c_{ij}, times t^c_{ij}, cost limit C, and time limit T). We can transform it into the electric vehicle routing problem in the following way:

- Keep the directed graph $G(N, A)$ the same, with the same source and sink

node.

- Set the energy consumption limit to T.
- Set the travel time limit to C.

From the formulation, we have that:

$$t_{ij}(v_{ij}) = \frac{\ell_{ij}}{v_{ij}} + \tau_{ij} \tag{3.3}$$

$$E_{ij}(v_{ij}) = \left(\begin{array}{l} \frac{\ell_{ij}}{\eta} \left(\frac{1}{2} \rho C_w A_f v_{ij}^2 + \mu m g \cos(\alpha_{ij}) + m g \sin(\alpha_{ij}) \right) \\ + P_{\text{acc}} \left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij} \right) \end{array} \right). \tag{3.4}$$

Now we select the parameters carefully. We first set the velocity lower bounds $\underline{v}_{ij} = \hat{v}_{ij}$ and the upper bounds $\overline{v}_{ij} = \hat{v}_{ij}$, simplify the selection of velocities to one parameter per arc: $\hat{v}_{ij}$. With this choice, the parameters that are arc-dependent are $\hat{v}_{ij}$, ℓ_{ij}, τ_{ij}, and α_{ij}. We will need to carefully choose the values of at least two of these (per arc), and we can basically set all of the rest of the parameters as we choose. Therefore,

- Set $P_{\text{acc}} = 0$.
- Use any positive values for ρ, C_w, A_f, μ, and m. We will set $C_w = 2$, $m = 1/g$, and the rest of the values to 1.
- Set $\alpha_{ij} = 0$, for all $(i, j) \in A$.
- Set $\hat{v}_{ij} = 1$, for all $(i, j) \in A$.

- Pick P_{acc} to be 0, although P_{acc} only needs to be small enough such that $t_{ij}^c - P_{\text{acc}} c_{ij}$ is positive for all $(i,j) \in A$. This ensures that ℓ_{ij} will be positive.

- Set $\tau_{ij} = c_{ij} - \ell_{ij}$. This is from solving

$$\frac{\ell_{ij}}{\hat{v}_{ij}} + \tau_{ij} = c_{ij}$$

and using $\hat{v}_{ij} = 1$.

- Set arc lengths $\ell_{ij} = \dfrac{(t_{ij}^c)\eta}{2}$.
This is from solving

$$\begin{aligned}
& E_{ij}(\hat{v}_{ij}) = t_{ij}^c \\
\Rightarrow \quad & \left(\begin{array}{l} \dfrac{\ell_{ij}}{\eta} \left(\dfrac{1}{2} \rho C_w A_f \left(\hat{v}_{ij}\right)^2 + \mu m g \cos\left(\alpha_{ij}\right) + m g \sin\left(\alpha_{ij}\right) \right) \\ + P_{\text{acc}} \left(\dfrac{\ell_{ij}}{\hat{v}_{ij}} + \tau_{ij} \right) \end{array} \right) = t_{ij}^c \\
\Rightarrow \quad & \left(\begin{array}{l} \dfrac{\ell_{ij}}{\eta} \left(\dfrac{1}{2}(1)(2)(1)\,(1)^2 + (1)(1/g) g \cos\left(0\right) + (1/g) g \sin\left(0\right) \right) \\ + 0 \left(\dfrac{\ell_{ij}}{1} + c_{ij} - \ell_{ij} \right) \end{array} \right) = t_{ij}^c \\
\Rightarrow \quad & \left(\frac{\ell_{ij}}{\eta} (2) \right) = t_{ij}^c \\
\Rightarrow \quad & \ell_{ij} = \frac{(t_{ij}^c)\eta}{2}
\end{aligned}$$

- Pick η to be positive, but small enough such that $\ell_{ij} \leq c_{ij}$ for all $(i,j) \in A$. $\eta = \min_{(i,j)\in A} \left\{ \dfrac{2c_{ij}}{t_{ij}^c} \right\}$ will suffice. This ensures that τ_{ij} will be nonnegative for all $(i,j) \in A$.

The objective function is then min $\{\sum_{(i,j)\in A} c_{ij}x_{ij}\}$, and the energy consumption constraint is $\sum_{(i,j)\in A} t_{ij}x_{ij} \leq T$.

□

3.2 Incorporating Uncertainty Using Robust Optimization

We describe how to apply robust optimization to find routes for battery electric vehicles under uncertainty in time and energy. A general description of robust optimization is in Section 1.1.

We assume that there is an uncertainty in time and energy that is not velocity dependent, and we use approaches from Bertsimas and Sim [7, 8] to formulate and solve the corresponding optimization problem. The robust-optimization approach assumes that each arc (i, j) has an uncertainty in time, Δt_{ij}, and in energy, Δe_{ij}, where typically $0 \leq \Delta t_{ij} \leq \sigma_{ij}$ and $0 \leq \Delta e_{ij} \leq \tilde{\sigma}_{ij}$. A conservative approach to handling this uncertainty is to assume the worst: a driver (or an algorithm) selects a path P, and then an adversary controls all of the uncertainty to maximize $\sum_{(i,j)\in P} \Delta t_{ij}$ and $\sum_{(i,j)\in P} \Delta e_{ij}$. The adversary can maximize each sum separately. In this case, the adversary would set $\Delta t_{ij} = \sigma_{ij}$ and $\Delta e_{ij} = \tilde{\sigma}_{ij}$ for all arcs $(i, j) \in P$ (this is the same as setting $\Delta t_{ij} = \sigma_{ij}$ and $\Delta e_{ij} = \tilde{\sigma}_{ij}$ for all arcs $(i, j) \in A$ before a path is selected). A less conservative approach is to restrict the power of the adversary to select the uncertainties in times and energy within uncertainty sets. U_t represents the uncertainty set for time and U_e the uncertainty set for energy. This is the approach we take, and we formulate the robust optimization model as

$$
\begin{aligned}
\min_{v_{ij},x_{ij}} \max_{\boldsymbol{\Delta t}\in U_t} \quad & \sum_{(i,j)\in A}\left(\frac{\ell_{ij}}{v_{ij}}+\tau_{ij}\right)x_{ij}+\sum_{(i,j)\in A}\left(\Delta t_{ij}\right)x_{ij} \\
\text{s.t.} \quad & \sum_{(i,j)\in A}\left(E_{ij}(v_{ij})\right)x_{ij}+\sum_{(i,j)\in A}(\Delta e_{ij})x_{ij}\leq E_0, \quad \forall \boldsymbol{\Delta e}\in U_e \\
& \underline{v}_{ij}\leq v_{ij}\leq \overline{v}_{ij}, \quad \forall (i,j)\in A \\
& \boldsymbol{x}\in X = \text{set of paths},
\end{aligned}
\tag{3.5}
$$

which can be restated as

$$
\begin{aligned}
\min_{v_{ij},x_{ij}} \quad & \sum_{(i,j)\in A}\left(\frac{\ell_{ij}}{v_{ij}}+\tau_{ij}\right)x_{ij}+\max_{\boldsymbol{\Delta t}\in U_t}\left\{\sum_{(i,j)\in A}\left(\Delta t_{ij}\right)x_{ij}\right\} \\
\text{s.t.} \quad & \sum_{(i,j)\in A}\left(E_{ij}(v_{ij})\right)x_{ij}+\max_{\boldsymbol{\Delta e}\in U_e}\left\{\sum_{(i,j)\in A}\left(\Delta e_{ij}\right)x_{ij}\right\}\leq E_0 \\
& \underline{v}_{ij}\leq v_{ij}\leq \overline{v}_{ij}, \quad \forall (i,j)\in A \\
& \boldsymbol{x}\in X = \text{set of paths}.
\end{aligned}
\tag{3.6}
$$

To complete the formulation (3.6), we need to specify the uncertainty sets U_t and U_e. For example, we could assume the worst-case scenario for time and set $U_t = \left\{\Delta t_{ij} | 0 \leq \Delta t_{ij} \leq \sigma_{ij}\right\}$. Then $\max_{\boldsymbol{\Delta t}\in U_t}\left\{\sum_{(i,j)\in A}\left(\Delta t_{ij}\right)x_{ij}\right\} = \sum_{(i,j)\in A}\left(\sigma_{ij}\right)x_{ij}$, which is the case when the adversary sets $\Delta t_{ij} = \sigma_{ij}$ for all arcs $(i,j) \in A$. U_e could be defined similarly. However, we choose more restrictive uncertainty sets to obtain less conservative solutions to (3.6).

The uncertainty sets we use are:

Polyhedral Uncertainty: $U_t = \left\{ \Delta t_{ij} | 0 \leq \Delta t_{ij} \leq \sigma_{ij}, \sum_{(i,j)\in A} \frac{\Delta t_{ij}}{\sigma_{ij}} \leq \Gamma_t \right\}$, where Γ_t is a parameter (assumed integer) that restricts the power of the adversary. In a network optimization context, this is equivalent allowing the adversary to set $\Delta t_{ij} = \sigma_{ij}$ for Γ_t arcs and setting $\Delta t_{ij} = 0$ for the rest. For polyhedral uncertainty,

$$\max_{\boldsymbol{\Delta t}\in U_t} \left\{ \sum_{(i,j)\in A} \left(\Delta t_{ij}\right) x_{ij} \right\} = \max_{\{S_t | S_t \subset A, |S_t| \leq \Gamma_t\}} \left\{ \sum_{(i,j)\in S_t} \left(\sigma_{ij}\right) x_{ij} \right\}.$$

Ellipsoidal Uncertainty: $U_t = \left\{ \Delta t_{ij} | \, ||\Sigma_t^{-1/2} \boldsymbol{\Delta t}||_2 \leq \Omega_t \right\}$, where Σ_t is a diagonal matrix with entries σ_{ij} along the diagonal and zeros everywhere else. Namely, Σ_t is an ellipse and its radius is determined by a parameter Ω_t. For ellipsoidal uncertainty,

$$\max_{\boldsymbol{\Delta t}\in U_t} \left\{ \sum_{(i,j)\in A} \left(\Delta t_{ij}\right) x_{ij} \right\} = \max_{\left\{\boldsymbol{\Delta t_{ij}} | \, ||\Sigma_t^{-1/2} \boldsymbol{\Delta t_{ij}}||_2 \leq \Omega_t\right\}} \left\{ \sum_{(i,j)\in A} \left(\Delta t_{ij}\right) x_{ij} \right\}.$$

The specifications for the kind of uncertainty sets in energy, U_e, is similar, and they would have parameters for energy instead of time: $\tilde{\sigma}_{ij}$, Σ_e, Ω_e, and Γ_e.

The robust optimization formulation (3.6) is more general than the deterministic framework, because setting $U_t = \emptyset$ and $U_e = \emptyset$, produces the deterministic model (3.2).

3.3 Lagrangian Relaxation

For many of our algorithms, we use the Lagrangian relaxation, which for problem (3.6) can be stated as:

$$
\begin{aligned}
L(\lambda) = \min_{v_{ij}, x_{ij}} \quad & \sum_{(i,j)\in A} \left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij} \right) x_{ij} + \max_{\boldsymbol{\Delta t} \in U_t} \left\{ \sum_{(i,j)\in A} \left(\Delta t_{ij} \right) x_{ij} \right\} \\
& + \lambda \left(\sum_{(i,j)\in A} \left(E_{ij}(v_{ij}) \right) x_{ij} + \max_{\boldsymbol{\Delta e} \in U_e} \left\{ \sum_{(i,j)\in A} \left(\Delta e_{ij} \right) x_{ij} \right\} - E_0 \right) \qquad (3.7) \\
\text{s.t.} \quad & \underline{v}_{ij} \le v_{ij} \le \overline{v}_{ij}, \quad \forall (i,j) \in A \\
& \boldsymbol{x} \in X = \text{set of paths.}
\end{aligned}
$$

Note:

- The Lagrangian multiplier λ is required to be nonnegative.
- Because $L(\lambda)$ is a lower bound on the objective value of problem (3.6), solving $\max_{\lambda \ge 0} \{L(\lambda)\}$ obtains the best lower bound.
- The solution to $\max_{\lambda \ge 0} \{L(\lambda)\}$ is guaranteed to be a path with velocities within the specified bounds, but it might be infeasible due to a violation of the energy constraint.
- $L(\lambda)$ is a piecewise-linear concave function of λ.
- Suppose that $\boldsymbol{x}^*$ and $\boldsymbol{v}^*$ are an optimal solution to $L(\lambda)$. Then a subgradient

of $L(\lambda)$ at λ is

$$\sum_{(i,j)\in A} \left(E_{ij}(v_{ij}^*)\right) x_{ij}^* + \max_{\mathbf{\Delta e}\in U_e} \left\{ \sum_{(i,j)\in A} \left(\Delta e_{ij}\right) x_{ij}^* \right\} - E_0.$$

We can solve the optimization for $L(\lambda)$ in the following way:

1. Solve the deterministic case, where there is no uncertainty. The details are in Section 4.1.2.

   (a) The objective function for $L(\lambda)$ is coupled only by the path variables. Namely, given any particular choice of path, the problem of finding optimal velocities is decomposable into $|A|$ simpler subproblems, which are easy to solve. It turns out that we can find the optimal velocity for each arc regardless of the path chosen.

   (b) Use the optimal velocities for each arc as input and solve a shortest path problem to evaluate $L(\lambda)$.

2. For the cases with uncertainty, use the techniques for deterministic case (the uncertainty does not depend on the variables v_{ij}), and then incorporate techniques to solve the robust version of a shortest path problem. See Section 4.2.2 for more details for the case of polyhedral uncertainty and Section 4.3.2 for more details for the case of ellipsoidal uncertainty.

We optimize the Lagrangian relaxation using subgradient ascent, as described in Section 16.3 of Ahuja et al. [1]. To obtain an initial upper bound on the objective function of the full optimization problem (such as (3.6)), we solve the appropri-

ate minimize-energy optimization problem (see Sections 4.1.1, 4.2.1, and 4.3.1). If the optimal minimum energy-solution requires more than the allowed amount of energy, we know that there is no feasible solution. The traversal time required for this solution is the value of the upper bound. Update the upper bound with the objective value (traversal time) of any better feasible solution that is found along the way. Subgradient ascent is a heuristic method, although $L_{\text{det}}(\lambda)$ for any $\lambda \geq 0$ is guaranteed to be a lower bound on the objective value of the optimal solution. Because the subgradient will rarely be 0, we have added the additional stopping criteria of terminating the algorithm after a predefined number of iterations (as mentioned in Ahuja et al. [1]), which we typically set to 10.

The detailed method of subgradient ascent is described below, (applied from [1]).

1. Initialization:

   - $\lambda_k = 0$
   - $\overline{\lambda}_k = 2$
   - θ_k is the stepsize on iteration k to be determined later.
   - $UB =$ upper bound on objective value
   - Solve the minimum-energy shortest path problem for the appropriate case. This will always generate a feasible solution, if one exists. The traversal time required of this solution is a value for UB.

2. Suppose that we can evaluate the value of the Lagrangian function $L(\lambda_k)$ and obtain a subgradient $g(\lambda_k)$ at λ_k. How to do this is explained in the

sections for each particular formulation (Sections 4.1.2, 4.2.2, and 4.3.2).

3. Subgradient Ascent: Given λ_k, UB, $L(\lambda_k)$, and $g(\lambda_k)$:

   - Stepsize $\theta_k = \dfrac{\overline{\lambda}_k \left(UB - L(\lambda_k)\right)}{||g(\lambda_k)||^2}$
   - Update: $\lambda_{k+1} = \min\left\{\lambda_k + \theta_k g(\lambda_k),\, 0\right\}$.

4. Stopping Criterion and Parameter Updates:

   - If after a few iterations of finding $L(\lambda_k)$ there is consistently no objective value improvement, decrease $\overline{\lambda}_k$ by a factor of 2: ($\overline{\lambda_k} = \overline{\lambda_k}/2$)
   - Terminate the algorithm when $||g(\lambda_k)||^2 < \epsilon$, for some small $\epsilon > 0$. However, since this condition rarely happens, also terminate the algorithm after completing set number of iterations. Return the final value of λ_k as λ^*.
   - Update UB with the objective value (traversal time) of any better feasible solution that is found along the way.

5. Error Bounds: Suppose we terminate with λ^* and a best feasible solution $(\boldsymbol{v}^*, \boldsymbol{x}^*)$. Then we have

   $$L(\lambda^*) \leq OPT \leq UB$$

   And that the solution $(\boldsymbol{v}^*, \boldsymbol{x}^*)$ has an error of at most

   $$\frac{UB - L(\lambda^*)}{L(\lambda^*)}.$$

3.4 Showing Equivalence of Robust Optimization Problems

In this section, we provide the theorems and proofs that demonstrate that robust optimization problems such as (3.5) are equivalent when we add the additional requirement that the adversary modifies the same arcs for both time and energy (instead of letting the adversary choose the arcs for time and energy separately), as in problem (3.8), for some fairly general kinds of uncertainty sets U_t and U_e. Typically U_t and U_e will be the same or related for the equivalence to hold.

$$\begin{aligned}
\min_{v_{ij},x_{ij}} \quad \max_{\substack{\boldsymbol{\Delta t}\in U_t \\ \boldsymbol{\Delta e}\in U_e \\ \Delta t_{ij}=0\Leftrightarrow\Delta e_{ij}=0,\forall(i,j)\in A}} \quad & \sum_{(i,j)\in A}\left(\frac{\ell_{ij}}{v_{ij}}+\tau_{ij}\right)x_{ij}+\sum_{(i,j)\in A}\left(\Delta t_{ij}\right)x_{ij} \\
\text{s.t.} \quad & \sum_{(i,j)\in A}\left(E_{ij}(v_{ij})\right)x_{ij}+\sum_{(i,j)\in A}(\Delta e_{ij})x_{ij}\le E_0 \qquad (3.8)\\
& \underline{v}_{ij}\le v_{ij}\le \overline{v}_{ij}, \quad \forall(i,j)\in A \\
& \boldsymbol{x}\in X=\text{set of paths},
\end{aligned}$$

The results in this section are of theoretical interest because they provide insight about the robust optimization modeling framework. The results are based on the idea that violating one constraint is considered equally as bad as violating any other constraint. The results emphasize that the robust optimization is required to protect against all possible scenarios, while the adversary may focus the uncertainty on violating only one constraint or just in making the solution as suboptimal as possible (and ignoring the other scenarios that are not beneficial

to the adversary). In addition, the results show that we may formulate a robust optimization problem in multiple ways, and we may pick whichever formulation that we feel is easier to solve or has a more relevant interpretation of the uncertainty. For example, for the problem of routing battery electric vehicles, the equivalence means that the robust optimization problem (3.5) is the same if we add the additional restriction that the adversary chooses either the uncertainty in time or energy of an arc to be nonzero, it must choose *both* the uncertainty time and uncertainty in energy of that arc to each be nonzero.

3.4.1 An Example Equivalence of Coupled and Decoupled System for Projections

First, consider an example. Let U be some (uncertainty) set that contains pairs of vectors of the form (a^1, a^2). Consider the projections of U onto a^1 and a^2:

- $U^1_{\text{proj}} = \{a^1 \text{ s.t. } (a^1, a^2) \in U\}$
- $U^2_{\text{proj}} = \{a^2 \text{ s.t. } (a^1, a^2) \in U\}$

Consider two systems

"Independent System"

$$f_1(a^1, x) + g_1(y) \leq d \quad \forall a^1 \in U^1_{\text{proj}}$$
$$f_2(a^2, x) + g_2(y) \leq h \quad \forall a^2 \in U^2_{\text{proj}}$$

"Coupled System"

$$\left.\begin{array}{l} f_1(a^1, x) + g_1(y) \leq d \\ f_2(a^2, x) + g_2(y) \leq h \end{array}\right\} \quad \forall (a^1, a^2) \in U$$

Where:

- $f_1(a, x)$ and $f_2(a, x)$ are functions from $\mathbb{R}^{2n} \to \mathbb{R}$.
- $g_1(y)$ and $g_2(y)$ are functions from $\mathbb{R}^n$ to $\mathbb{R}$.
- All of the other data are vectors.

In the independent system, the inequalities must hold for all coefficients a separately in the set U. That is, (a^1, a^2) need not be in U. Therefore, the coupled system is contained in the independent system, as $(a^1, a^2) \in U$ implies $a^1 \in U^1_{\text{proj}}$ and $a^2 \in U^2_{\text{proj}}$.

Any inequality

$$f(s, x) + g(y) \leq d$$

holds for all vectors s in some set $\tilde{U}$ if and only if it holds for

$$s^*(x, y) = \underset{s \in \tilde{U}}{\operatorname{argmax}} \{f(s, x) + g(y)\}$$

where $s^*(x, y)$ depends on x and y.

Let

$$U_{EP}^{i} = \left\{ \begin{array}{c} s^*(x,y) \in U_{\text{proj}}^{i} \text{ s.t. } s^*(x,y) = \underset{s \in U_{\text{proj}}^{i}}{\text{argmax}} \{f_i(s,x) + g_i(y)\} \\ \\ \text{for some } x \text{ and } y \end{array} \right\}$$

$$\forall i = 1, 2$$

Therefore, we can rewrite the independent and coupled systems as follows:

Equivalent Independent System

$$f_1(a^1, x) + g_1(y) \leq d \quad \forall a^1 \in U_{EP}^{1}$$
$$f_2(a^2, x) + g_2(y) \leq h \quad \forall a^2 \in U_{EP}^{2}$$

Equivalent Coupled System

$$\left. \begin{array}{l} f_1(a^1, x) + g_1(y) \leq d \\ f_2(a^2, x) + g_2(y) \leq h \end{array} \right\} \quad \forall (a^1, a^2) \in U \text{ s.t. } a^1 \in U_{EP}^{1} \text{ or } a^2 \in U_{EP}^{2}$$

The equivalent coupled system contains each inequality in the equivalent independent system and additional (redundant) inequalities, although the (non-redundant) inequalities might be ordered differently in the two systems.

Therefore, these systems are the same.

3.4.2 General Results

Notation:

- U is some uncertainty set that contains vectors in $\mathbb{R}^n$

- $\boldsymbol{a}$ and $\boldsymbol{a^i}$ are vectors in U, for $i = 0, \ldots, N$
- $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors in $\mathbb{R}^n$
- X and Y are feasible regions in $\mathbb{R}^n$, where usually $\boldsymbol{x} \in X$ and $\boldsymbol{y} \in Y$
- $f_i(\boldsymbol{a}, \boldsymbol{x})$ are functions mapping $\mathbb{R}^{2n}$ to $\mathbb{R}$, where $\boldsymbol{a} \in U$ and $\boldsymbol{x} \in X$, for $i = 0, \ldots, N$
- $g_i(\boldsymbol{y})$ are functions mapping $\mathbb{R}^n$ to $\mathbb{R}$, where $\boldsymbol{y} \in Y$, for $i = 0, \ldots, N$.
- b_i are real numbers, for $i = 1, \ldots, N$

Theorem 3.3. Consider the decoupled and coupled robust optimization problems:

Decoupled:

$$\begin{aligned} \min_{\substack{\boldsymbol{x}\in X \\ \boldsymbol{y}\in Y}} \max_{\boldsymbol{a^0}\in U} \quad & f_0(\boldsymbol{a^0}, \boldsymbol{x}) + g_0(\boldsymbol{y}) \\ \text{s.t.} \quad & f_i(\boldsymbol{a^i}, \boldsymbol{x}) + g_i(\boldsymbol{y}) \leq b_i \quad \forall \boldsymbol{a^i} \in U, \quad \forall i = 1, \ldots, N \end{aligned} \tag{3.9}$$

Coupled:

$$\begin{aligned} \min_{\substack{\boldsymbol{x}\in X \\ \boldsymbol{y}\in Y}} \max_{\boldsymbol{a}\in U} \quad & f_0(\boldsymbol{a}, \boldsymbol{x}) + g_0(\boldsymbol{y}) \\ \text{s.t.} \quad & f_i(\boldsymbol{a}, \boldsymbol{x}) + g_i(\boldsymbol{y}) \leq b_i \quad \forall i = 1, \ldots, N \end{aligned} \tag{3.10}$$

The decoupled and coupled robust optimization problems are equivalent. Namely, they have the same feasible regions and optimal solutions.

Proof of Theorem 3.3. We first show that the decoupled and coupled problems have the same feasible region.

Suppose that $(\boldsymbol{x}, \boldsymbol{y})$ is feasible for the decoupled problem. Then it is also feasible for the coupled problem, as $\boldsymbol{x} \in X$, $\boldsymbol{y} \in Y$, and because the coupled problem is the same as adding the restriction $\boldsymbol{a^1} = \boldsymbol{a^2} = \cdots = \boldsymbol{a^N}$ to the uncertainty set for the decoupled problem.

Now suppose that $(\boldsymbol{x}, \boldsymbol{y})$ is not feasible for the decoupled problem. If $\boldsymbol{x} \notin X$ or $\boldsymbol{y} \notin Y$, then it is infeasible for the coupled problem, so suppose that $\boldsymbol{x} \in X$ and $\boldsymbol{y} \in Y$. Because $\boldsymbol{x} \in X$ and $\boldsymbol{y} \in Y$ are not feasible for the decoupled problem, there exists an $\boldsymbol{a^i} \in U$ such that

$$f_i(\boldsymbol{a^i}, \boldsymbol{x}) + g_i(\boldsymbol{y}) > b_i.$$

Choosing $\boldsymbol{a} = \boldsymbol{a^i} \in U$, along with $(\boldsymbol{x}, \boldsymbol{y})$ produces the infeasible solution for the coupled problem, as

$$f_i(\boldsymbol{a}, \boldsymbol{x}) + g_i(\boldsymbol{y}) > b_i.$$

Therefore, the feasible regions of the decoupled and coupled problems are the same. Call this feasible region A.

The decoupled and coupled optimization problems can be written as:

Decoupled:

$$\begin{aligned} \min_{\substack{\boldsymbol{x}\in X\\ \boldsymbol{y}\in Y}} \max_{\boldsymbol{a^0}\in U} \quad & f_0(\boldsymbol{a^0}, \boldsymbol{x}) + g_0(\boldsymbol{y}) \\ \text{s.t.} \quad & (\boldsymbol{x}, \boldsymbol{y}) \in A \end{aligned} \tag{3.11}$$

Coupled:

$$\begin{aligned} \min_{\substack{\boldsymbol{x}\in X\\ \boldsymbol{y}\in Y}} \max_{\boldsymbol{a}\in U} \quad & f_0(\boldsymbol{a}, \boldsymbol{x}) + g_0(\boldsymbol{y}) \\ \text{s.t.} \quad & (\boldsymbol{x}, \boldsymbol{y}) \in A \end{aligned} \tag{3.12}$$

These optimization problems are the same, and therefore have the same optimal solutions. □

Notation:

- $\boldsymbol{a^i}$ are vectors with real number entries, with dimension d_i, for $i = 0, \ldots, N$. Note that the $\boldsymbol{a^i}$ can be of different dimensions
- U is some uncertainty set that contains vectors (of vectors) $\left(\boldsymbol{a^0}, \boldsymbol{a^1}, \ldots, \boldsymbol{a}^n\right)$
- $U^i_{\text{proj}} = \{\boldsymbol{a^i}$ s.t. there exist $\boldsymbol{a^0}, \boldsymbol{a^1}, \ldots, \boldsymbol{a^{i-1}}, \boldsymbol{a^{i+1}}, \ldots, \boldsymbol{a^N}$ where $(\boldsymbol{a^0}, \boldsymbol{a^1}, \ldots, \boldsymbol{a^N}) \in U\}$. Namely, U^i_{proj} is the projection of U onto the subspace spanned by the $\boldsymbol{a^i}$. Here, $i = 0, \ldots, N$
- $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors in $\mathbb{R}^n$
- X and Y are feasible regions in $\mathbb{R}^n$, where usually $\boldsymbol{x} \in X$ and $\boldsymbol{y} \in Y$
- $f_i(\boldsymbol{a^i}, \boldsymbol{x})$ are functions mapping $\mathbb{R}^{d_i+n}$ to $\mathbb{R}$, where $\boldsymbol{a^i} \in U^i_{\text{proj}}$ and $\boldsymbol{x} \in X$, for $i = 0, \ldots, N$
- $g_i(\boldsymbol{y})$ are functions mapping $\mathbb{R}^n$ to $\mathbb{R}$, where $\boldsymbol{y} \in Y$, for $i = 0, \ldots, N$
- b_i are real numbers, for $i = 1, \ldots, N$

Theorem 3.4. Consider the decoupled and coupled robust optimization problems:

Decoupled:

$$\begin{aligned} \min_{\substack{x\in X\\ y\in Y}} \max_{a^0\in U^0_{\text{proj}}} \quad & f_0(a^0, x) + g_0(y) \\ \text{s.t.} \quad & f_i(a^i, x) + g_i(y) \le b_i \quad \forall a^i \in U^i_{\text{proj}}, \quad \forall i = 1, \dots, N \end{aligned} \tag{3.13}$$

Coupled:

$$\begin{aligned} \min_{\substack{x\in X\\ y\in Y}} \max_{(a^0, a^1, \dots, a^n)\in U} \quad & f_0(a^0, x) + g_0(y) \\ \text{s.t.} \quad & f_i(a^i, x) + g_i(y) \le b_i \quad \forall i = 1, \dots, N \end{aligned} \tag{3.14}$$

The decoupled and coupled robust optimization problems are equivalent. Namely, they have the same feasible regions and optimal solutions.

Proof of Theorem 3.4. The proof is similar to the proof of Theorem 3.3 and to the proof for the example in Section 3.4.1. □

Chapter 4

Algorithms

Our ultimate goal is to solve the electric vehicle routing problem under uncertainty and use techniques from robust optimization to formulate and solve problem (3.6). The objective, broadly speaking, is to select a route and vehicle speed to minimize travel time while ensuring that total energy consumption does not exceed a pre-specified limit.

Our main approach for solving the deterministic and robust problems as follows:

1. Choose some reasonable solutions to the problem, such as a minimum-time or minimum-energy solution. Such "candidate solutions" are easy to calculate, although they might not be feasible.

2. Because the mathematical formulation is NP-Hard, we use Lagrangian relaxation as a heuristic.

3. In general, solving a Lagrangian relaxation does not guarantee a feasible so-

lution. However, we guarantee a feasible solution to problem (3.7) by finding an initial feasible solution using an appropriate minimum-energy solution, and then keeping track of whether or not the algorithm finds better feasible solutions in the process of solving the Lagrangian relaxation.

4. We evaluate the quality of the feasible solution by comparing it to the lower bound provided by solving the Lagrangian. Technically speaking, we get a lower bound from the Lagrangian when we solve the appropriate subproblem optimally. When we solve that subproblem heuristically, as for the robust optimization case, we are not guaranteed a lower bound from Lagrangian relaxation, and can only use it to get an estimate of a lower bound on the objective function.

Some additional comments:

- Typically, heuristic methods are applied to each of the steps (depending on the complexity of the problem), so optimal solutions at each step are not guaranteed.

- In all of the Lagrangian relaxations and candidate solutions, the optimal velocity of any arc is independent of the path chosen and the velocities on other arcs. Therefore, optimal velocities can be found by solving a convex optimization problem for each arc.

- Solving shortest path problems is a subroutine in all the instances. To solve it more quickly, we formulate the problem in a way that has nonnegative

arc weights and use A-star to find a shortest path from a given source to a given destination.

We first discuss the algorithms for the deterministic model (Section 4.1). We then present algorithms for the cases involving robust optimization: polyhedral uncertainty (Section 4.2) and ellipsoidal uncertainty (Section 4.3). Next, we describe how to use an A-star algorithm to solve the shortest path problems in various algorithm subroutines more quickly (Section 4.4). We conclude this chapter with pseudo-polynomial algorithms for the electric vehicle routing problem (Section 4.5).

4.1 Deterministic Model: No Uncertainty

As a preliminary case, we consider the optimization model without uncertainty.

4.1.1 Candidate Solutions

We start by finding three solutions that could be candidates for a feasible or optimal solution:

Minimize Time: We ignore the energy constraint, and minimize the total traversal time. Arc velocities are set to their upper bounds, and the problem becomes a shortest path problem with arc costs as traversal times.

Minimize Length: We ignore the energy constraint, and minimize the total length of the route. This problem becomes a shortest path problem with arc costs as arc lengths. Because velocity does not matter, arc velocities are set to their upper bounds.

Minimize Energy: Instead of using an energy constraint, we minimize the energy consumption of the route. We can find the arc velocities using the techniques in the proof of Theorem 4.1 (in Section 4.1.2). We do this by solving the convex optimization problem $\min_{\underline{v}_{ij} \le v_{ij} \le \overline{v}_{ij}} \left\{ E_{ij}(v_{ij}) \right\}$. The optimal velocity for arc (i, j) is either $\underline{v}_{ij}$, $\overline{v}_{ij}$, or the solution to $\dfrac{\partial}{\partial v_{ij}} E(v_{ij}) = 0$, which is $\sqrt[3]{\eta \left(\dfrac{P_{\text{acc}}}{\rho C_w A_f} \right)}$. If $\sqrt[3]{\eta \left(\dfrac{P_{\text{acc}}}{\rho C_w A_f} \right)}$ is not in $\left[\underline{v}_{ij}, \overline{v}_{ij} \right]$, then one of the two endpoints of $\left[\underline{v}_{ij}, \overline{v}_{ij} \right]$ must be optimal. With the optimal velocities as input, the minimum-energy problem becomes a shortest path problem with arc costs as energy consumption.

We obtain additional information from the candidate solutions:

Feasible Solution: If the energy limit is less than the energy consumption of the minimum-energy solution, then the optimization problem has no feasible solution. Otherwise, the minimum-energy solution is a feasible solution to the problem.

Unconstrained Optimal Solution: The most energy that could be optimal to consume is the energy required for the minimum-time solution. If this solution is feasible, then the minimum-time solution is optimal for the deterministic optimization problem.

4.1.2 Solving the Lagrangian Relaxation

The Lagrangian relaxation of the deterministic optimization problem is

$$\begin{aligned} L_{\text{det}}(\lambda) = \min_{v_{ij}, x_{ij}} \quad & \sum_{(i,j)\in A} \left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij} \right) x_{ij} + \lambda \left(\sum_{(i,j)\in A} \left(E_{ij}(v_{ij}) \right) x_{ij} - E_0 \right) \\ \text{s.t.} \quad & \underline{v}_{ij} \le v_{ij} \le \overline{v}_{ij}, \quad \forall (i,j) \in A \\ & \boldsymbol{x} \in X = \text{set of paths.} \end{aligned} \tag{4.1}$$

There are two parts to solving the Lagrangian relaxation at a fixed value of λ: finding velocities and finding a path.

Theorem 4.1. The optimal velocity, v_{ij}^*, to problem (4.1), for each arc (i,j), is the velocity with the smallest objective value chosen from the following 3 choices:

Lower Bound: $\underline{v}_{ij}$

Upper Bound: $\overline{v}_{ij}$

Solution to the Unconstrained Case: $\sqrt[3]{\eta \left(\frac{1}{\lambda \rho C_w A_f} + \frac{P_{\text{acc}}}{\rho C_w A_f} \right)}$. This velocity is a possible solution only if it is in the interval of feasible velocities $[\underline{v}_{ij}, \overline{v}_{ij}]$.

Proof of Theorem 4.1. For the time being, suppose that we fix any $\boldsymbol{x} \in X \subset$

$\{0, 1\}^{|A|}$. Then the optimization problem becomes:

$$\begin{aligned} \min_{v_{ij}} \quad & \sum_{(i,j)\in A} \left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij} \right) x_{ij} + \lambda \left(\sum_{(i,j)\in A} \left(E_{ij}(v_{ij}) \right) x_{ij} - E_0 \right) \\ \text{s.t.} \quad & \underline{v}_{ij} \leq v_{ij} \leq \overline{v}_{ij}, \quad \forall (i,j) \in A. \end{aligned}$$

This problem is separable by arc, and reduces to solving $|A|$ problems of the form

$$\begin{aligned} \min_{v_{ij}} \quad & \left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij} \right) x_{ij} + \lambda \left(E_{ij}(v_{ij}) x_{ij} \right) \\ \text{s.t.} \quad & \underline{v}_{ij} \leq v_{ij} \leq \overline{v}_{ij}. \end{aligned}$$

If $x_{ij} = 0$, any arc velocity is optimal (with total cost 0), and therefore the optimal velocity for $x_{ij} = 1$ will be optimal for all cases. When $x_{ij} = 1$, for the particular choice of energy function (3.1), this problem becomes (after removing some constant terms that do not alter the optimal solution):

$$\begin{aligned} \min_{v_{ij}} \quad & \frac{\ell_{ij}}{v_{ij}} \left(1 + \lambda P_{\text{acc}} \right) + \lambda \left(\frac{\ell_{ij}}{\eta} \left(\frac{1}{2} \rho C_w A_f v_{ij}^2 \right) \right) \\ \text{s.t.} \quad & \underline{v}_{ij} \leq v_{ij} \leq \overline{v}_{ij}. \end{aligned} \tag{4.2}$$

This is a convex optimization problem, and we can arrive at the same optimal solution in two ways:

- Find all possible solutions that satisfy the Karush-Kuhn Tucker conditions, and select the solution with the smallest objective value.

- Minimize an unconstrained convex function of one variable, the velocity v_{ij}^*,

by setting the derivative equal to 0:

$$-\ell_{ij}\frac{(1+\lambda P_{acc})}{v_{ij}^2}+\lambda\left(\frac{\ell_{ij}}{\eta}\left(\rho C_w A_f v_{ij}\right)\right)=0,$$

which yields $v_{ij}^* = \sqrt[3]{\eta\left(\frac{1}{\lambda\rho C_w A_f}+\frac{P_{\text{acc}}}{\rho C_w A_f}\right)}$. If v_{ij}^* is not in $\left[\underline{v}_{ij},\overline{v}_{ij}\right]$, then one of the two endpoints of $\left[\underline{v}_{ij},\overline{v}_{ij}\right]$ must be optimal.

If $\lambda = 0$, the optimal solution to problem (4.2) is to choose $v_{ij} = \overline{v}_{ij}$. □

Since the optimal velocities are valid for all choices of x_{ij}, we can fix them at their optimal value and, the problem becomes a shortest path problem.

While we can find all of the arc velocities first and then solve a shortest path problem, it is more efficient to find the velocities on an as-needed basis. That is, we solve the shortest path problem without finding the optimal velocities. Whenever the algorithm examines an arc, we compute optimal velocity for that arc (using Theorem 4.1) and use it to evaluate the traversal time and energy consumption for that arc.

We optimize the Lagrangian relaxation using subgradient ascent, and a subgradient of $L_{\text{det}}(\lambda)$ at λ is

$$\sum_{(i,j)\in A}\left(E_{ij}(v_{ij}^*)\right)x_{ij}^* - E_0,$$

with optimal velocities and path v_{ij}^* and x_{ij}^*.

To guarantee a feasible solution (if one exists), we do the following:

- Check that the minimum-energy solution is feasible. If it is, we use it as an

initial feasible solution for subgradient ascent. If it is not feasible, then we say that problem (3.2) has no feasible solution.

- Each time we evaluate $L_{\text{det}}(\lambda)$ at some value of λ, we check whether the path is feasible and has a smaller traversal time (objective value for the original problem) than the current best feasible solution. If it is better, it becomes the the current best feasible solution.

4.2 Robust Optimization Model: Polyhedral Uncertainty

To solve robust optimization problems under polyhedral uncertainty, we will need to be able to incorporate it into the problem in way that is tractable (the main framework is discussed in Section 3.2). The problem formulation with Lagrangian relaxation is in Section 4.2.2, and the main idea is to simplify the problem by applying the techniques in Bertsimas and Sim [7] to

$$\max_{\boldsymbol{\Delta t} \in U_t} \left\{ \sum_{(i,j)\in A} \left(\Delta t_{ij}\right) x_{ij} \right\} = \max_{\{S_t | S_t \subset A, |S_t| \leq \Gamma_t\}} \left\{ \sum_{(i,j)\in S_t} \left(\sigma_{ij}\right) x_{ij} \right\}.$$

We use the fact that the set of paths is a subset of $\{0, 1\}^{|A|}$ to simplify the formulation of polyhedral uncertainty (see the proof of Theorem 4.2 in Section 4.2.2):

$$\max_{\{S_t | S_t \subset A, |S_t| \leq \Gamma_t\}} \left\{ \sum_{(i,j)\in S_t} \left(\sigma_{ij}\right) x_{ij} \right\} = \min_{\{\theta_t \geq 0\}} \left\{ \sum_{(i,j)\in A} \left(\max\left\{\sigma_{ij} - \theta_t, 0\right\}\right) x_{ij} + \Gamma_t \theta_t \right\}.$$

Under this transformation, we will typically use a subroutine to find the value of the theta variable as part of solving the desired optimization problem. To explain the solution method for the subroutine, suppose we want to solve the optimization problem with L theta variables:

$$\begin{aligned} \min_{x_{ij},\theta_l} \quad & \boldsymbol{c}'\boldsymbol{x} + \sum_{l=1}^{L} \left(\sum_{(i,j)\in A} \left(\max\left(\sigma_{ij}^{l} - \theta_l, 0\right) x_{ij} \right) + \Gamma_l \theta_l \right) \\ \text{s.t.} \quad & \theta_l \geq 0, \, \forall l = 1, \ldots, L \\ & \boldsymbol{x} \in X = \text{set of paths.} \end{aligned} \tag{4.3}$$

For any fixed value of each θ_l, this optimization problem is a shortest path problem. When $L = 1$, Bertsimas and Sim [7] provide ways to solve problems of this type to optimality. In addition, for the minimum cost flow problem (more general than the shortest path problem), Bertsimas and Sim [7], showed that an equivalent objective function is convex in θ. We propose an algorithm that is different than any of the algorithms in Bertsimas and Sim [7] to provide a good solution even faster, as well as for $L \geq 1$. This heuristic algorithm is algorithm 1.

Proof of optimality in step 2(b) of algorithm 1. Given a fixed $\boldsymbol{x}^k$, representing the path P^k, problem (4.3) is separable and can be decomposed into L convex optimizations problems, which we can solve for each θ_l independently of the others:

$$\begin{aligned} \min_{\theta_l} \quad & \sum_{(i,j)\in A} \left(\max\left(\sigma_{ij}^{l} - \theta_l, 0\right) x_{ij}^{k} \right) + \Gamma_l \theta_l \\ \text{s.t.} \quad & \theta_l \geq 0. \end{aligned}$$

Algorithm 1 Algorithm for solving problem (4.3).

1. Initialization (make an initial guess):

   - For each $l = 1, \ldots, L$, guess an initial value for θ_l, and call this guess θ_l^0. We recommend choosing some $\theta_l \in \left[0, \max_{(i,j)\in A}\{\sigma_{ij}^l\}\right]$, because θ_l is required to be nonnegative and $\max\left(\sigma_{ij}^l - \theta_l, 0\right)$ is 0 for $\theta_l \geq \max_{(i,j)\in A}\{\sigma_{ij}^l\}$ ($\sigma_{ij}^l - \theta_l \leq 0$ for such a value of θ_l). The values for each initial θ_l^0 can be different.
   - Alternatively, one could choose a path $\boldsymbol{x}^0 \in X$ as an initial guess and then use that path to set θ_l^0 using the procedure in step 2 in this algorithm.

2. Repeat until convergence (Until $\theta_l^k = \theta_l^{k+1}$ for all $l = 1, \ldots, L$):

   (a) Input the values of θ_l^k into problem (4.3) and solve the shortest path problem to obtain path $\boldsymbol{x}^k$.

   (b) Given $\boldsymbol{x}^k$, representing the path P^k, set the values of θ_l. An optimal value of θ_l is the $(\Gamma_l + 1)^{\text{th}}$ largest value of σ_{ij}^l for arcs on the chosen path. Instead, for the purposes of potentially avoiding degeneracy, choose θ_l to be the average between (midpoint of) the $(\Gamma_l + 1)^{\text{th}}$ largest value of σ_{ij}^l and the $(\Gamma_l)^{\text{th}}$ largest value of σ_{ij}^l of arcs the path P^k. If $\Gamma_l = 0$, set $\theta_l^{k+1} = \max_{(i,j)\in P^k}\{\sigma_{ij}^l\}$, and if $\Gamma_l \geq \left|P^k\right|$, set $\theta_l^{k+1} = 0$.

For a fixed value of $\boldsymbol{x}$ (and so a chosen path P^k), the rate of change of the objective function with respect to θ_l is

$$(-1)\left(\text{Number of arcs in path } P^k \text{ such that } \sigma_{ij}^l > \theta_l\right) + \Gamma_l,$$

and the set of optimal solutions is when this rate of change is 0. The rate of change is 0 is when there are exactly Γ_l arcs in in path P^k such that $\sigma_{ij}^l > \theta_l$. This happens for any value of θ_l that is at least the $(\Gamma_l + 1)^{\text{th}}$ largest value of σ_{ij}^l on path P_k^k and is (strictly) less than the $(\Gamma_l)^{\text{th}}$ largest value of σ_{ij}^l of arcs the path P^k. As a special case, if the $(\Gamma_l + 1)^{\text{th}}$ largest and $(\Gamma_l)^{\text{th}}$ largest values of σ_{ij}^l on path P^k are the same, then this value is the (only) optimal value of θ_l.

To see why the optimal solutions are when the rate of change is 0, one can take the derivative of the objective function at θ_l, although technically the objective function is not differentiable at all values of θ_l. To be more rigorous, use subgradients. A subgradient of $\max\left(\sigma_{ij}^l - \theta_l, 0\right)$ is -1 if $\sigma_{ij}^l > \theta_l$, and 0 otherwise, and the subgradient (derivative) of $\Gamma_l\theta_l$ is Γ_l. The sum of these subgradients is $(-1)\left(\text{Number of arcs in path } P^k \text{ such that } \sigma_{ij}^l > \theta_l\right) + \Gamma_l$ and it is a subgradient of the objective function. Also, because Γ_l and σ_{ij}^l are nonnegative, the values of θ_l are nonnegative when the rate of change is 0.

The special cases are:

- If $\Gamma_l = 0$, then the subgradient of the objective function is always nonpositive and an optimal solution is to set $\theta_l = \max_{(i,j)\in P^k}\{\sigma_{ij}^l\}$ (any θ_l at least $\max_{(i,j)\in P^k}\{\sigma_{ij}^l\}$ would be optimal).
- If $\Gamma_l \geq \left|P^k\right|$, the subgradient of the objective function is always nonnegative,

and the optimal solution is to set $\theta_l^* = 0$.

□

The objective value of problem (4.3) is nonincreasing throughout algorithm 1. To see this, let

$$Z_{\text{poly}}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{c}'\boldsymbol{x} + \sum_{l=1}^{L} \left(\sum_{(i,j)\in A} \left(\max\left(\sigma_{ij}^l - \theta_l^k, 0\right) x_{ij} \right) + \Gamma_l \theta_l^k \right).$$

Then we have for any integer $k \geq 0$,

$$\begin{aligned} Z_{\text{poly}}(\boldsymbol{x}^k, \boldsymbol{\theta}^k) &\geq \min_{\boldsymbol{x}\in X} \left\{ Z_{\text{poly}}(\boldsymbol{x}^k, \boldsymbol{\theta}^k) \right\} \\ &= Z_{\text{poly}}(\boldsymbol{x}^{k+1}, \boldsymbol{\theta}^k) \\ &\geq \min_{\theta_l \geq 0, \forall l=1,\ldots,L} \left\{ Z_{\text{poly}}(\boldsymbol{x}^{k+1}, \boldsymbol{\theta}^k) \right\} \\ &= Z_{\text{poly}}(\boldsymbol{x}^{k+1}, \boldsymbol{\theta}^{k+1}). \end{aligned}$$

4.2.1 Candidate Solutions

Besides the candidate solutions for the deterministic case, we consider two other candidate solutions:

Minimize Time—Robust Case with Polyhedral Uncertainty: We ignore the energy constraint, and minimize time, using the polyhedral uncertainty set, which becomes an optimization problem of the form (4.3). Like the deterministic case, the optimal choice of arc velocities is to set each arc velocity to its corresponding upper bound (go as fast as possible). We use a one-variable

version of algorithm 1 to find (heuristically) the value of the theta-variable and the corresponding path.

In more detail, we want to solve:

$$\begin{aligned} \min_{v_{ij}, x_{ij}} \quad & \sum_{(i,j)\in A} \left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij} \right) x_{ij} + \max_{\{S | S\subseteq A, |S|\leq \Gamma_t\}} \left\{ \sum_{(i,j)\in S} \left(\sigma_{ij}\right) x_j \right\} \\ \text{s.t.} \quad & \underline{v}_{ij} \leq v_{ij} \leq \overline{v}_{ij}, \quad \forall (i,j) \in A \\ & \boldsymbol{x} \in X \subset \{0,1\}^m \end{aligned}$$

Since time is minimized when going fastest, we will set $v_{ij} = \overline{v}_{ij}$ and solve the optimization problem:

$$\begin{aligned} \min_{x_{ij}} \quad & \sum_{(i,j)\in A} \left(\frac{\ell_{ij}}{\overline{v}_{ij}} + \tau_{ij} \right) x_{ij} + \max_{\{S | S\subseteq A, |S|\leq \Gamma_t\}} \left\{ \sum_{(i,j)\in S} \left(\sigma_{ij}\right) x_j \right\} \\ \text{s.t.} \quad & \boldsymbol{x} \in X \subset \{0,1\}^m. \end{aligned} \tag{4.4}$$

Now we can apply the theorems and methods of proof in Section 3 of [7] to solve the problem (4.4) (see the proof of Theorem 4.2). The result is the equivalent optimization problem

$$\begin{aligned} \min_{x_{ij}, \theta_t} \quad & \sum_{(i,j)\in A} \left(\frac{\ell_{ij}}{\overline{v}_{ij}} + \tau_{ij} \right) x_{ij} + \sum_{(i,j)\in A} \max\left(\sigma_{ij} - \theta_t, 0\right) x_{ij} + \Gamma_t \theta_t \\ \text{s.t.} \quad & \theta \geq 0 \\ & \boldsymbol{x} \in X \subset \{0,1\}^m. \end{aligned}$$

We solve this problem using the heuristic algorithm described in Section

4.2.2, although it is possible to solve the problem using either of the approaches from Bertsimas and Sim [7] that were previously described. We use the A-star algorithm to solve the shortest path problems that are part of the robust optimization algorithm (see Section 4.4.2).

To compute $\max_{\{S|S\subseteq A,|S|\leq\Gamma_t\}}\left\{\sum_{(i,j)\in S}(\sigma_{ij})x_{ij}\right\}$, examine all of the arcs j on the path $x_{ij}=1$ and add together the Γ_t largest values of σ_{ij}.

Minimize Energy—Robust Case with Polyhedral Uncertainty: We ignore the energy constraint, and minimize energy, using the polyhedral uncertainty set, which becomes an optimization problem similar to (4.3). Because the uncertainty does not depend on arc velocity, the optimal choice of arc velocities is the same as the deterministic case (solve $\min_{\underline{v}_{ij}\leq v_{ij}\leq\overline{v}_{ij}}\left\{E_{ij}(v_{ij})\right\}$). Then, we use a one-variable version of algorithm 1 to find (heuristically) the value of the theta-variable and the corresponding path.

In more detail, we want to solve:

$$\begin{aligned}\min_{v_{ij},x_{ij}}\quad & \sum_{(i,j)\in A}E_{ij}(v_{ij})x_{ij}+\max_{\{S|S\subseteq A,|S|\leq\Gamma_t\}}\left\{\sum_{(i,j)\in S}\left(\tilde{\sigma}_{ij}\right)x_j\right\}\\ \text{s.t.}\quad & \underline{v}_{ij}\leq v_{ij}\leq\overline{v}_{ij},\quad\forall(i,j)\in A\\ & \boldsymbol{x}\in X\subset\{0,1\}^m\end{aligned}\tag{4.5}$$

Apply the theorems and methods of proof in Section 3 of [7] to obtain the equivalent optimization problem to problem (4.4) (see the proof of Theorem

4.2):

$$\begin{aligned}
\min_{x_{ij}, v_{ij}, \theta_e} \quad & \sum_{(i,j)\in A} E_{ij}(v_{ij})x_{ij} + \sum_{(i,j)\in A} \max\left(\tilde{\sigma}_{ij} - \theta_e, 0\right)x_{ij} + \Gamma_e\theta_e \\
\text{s.t.} \quad & \underline{v}_{ij} \leq v_{ij} \leq \overline{v}_{ij}, \quad \forall (i,j) \in A \\
& \theta_e \geq 0 \\
& \boldsymbol{x} \in X \subset \{0,1\}^m.
\end{aligned}$$

We now select the optimal velocities v_{ij}^* as in the deterministic case (see Theorem 4.1), and substitute them into the optimization problem to obtain

$$\begin{aligned}
\min_{x_{ij}, \theta_e} \quad & \sum_{(i,j)\in A} E_{ij}(v_{ij}^*)x_{ij} + \sum_{(i,j)\in A} \max\left(\tilde{\sigma}_{ij} - \theta_e, 0\right)x_{ij} + \Gamma_e\theta_e \\
\text{s.t.} \quad & \theta_e \geq 0 \\
& \boldsymbol{x} \in X \subset \{0,1\}^m.
\end{aligned}$$

We solve this problem using the heuristic algorithm described in Section 4.2.2, although it is possible to solve the problem using either of the approaches from Bertsimas and Sim [7] that were previously described. We use the A-star algorithm to solve the shortest path problems that are part of the robust optimization algorithm (see Section 4.4.2).

To compute $\max_{\{S|S\subseteq A, |S|\leq\Gamma_e\}} \left\{\sum_{(i,j)\in S} (\tilde{\sigma}_{ij})x_{ij}\right\}$, look at all of the arcs j on the path $x_{ij} = 1$ and add together the Γ_e largest values of σ_{ij}.

As with the deterministic case, the optimal robust minimum-energy solution is the path with the least (planned or anticipated) energy consumption, which will

always be feasible, if a feasible solution exists. Also, the anticipated energy of the optimal robust minimum-time solution is the most energy any optimal solution would anticipate consuming.

4.2.2 Solving the Lagrangian Relaxation

The full formulation (problem (3.7)) can be transformed into the following equivalent problem in Theorem 4.2. It can be solved using algorithm 2.

Theorem 4.2 (**Bertsimas and Sim [7]**). Problem (3.7) is equivalent to solving the following optimization problem:

$$
\begin{aligned}
L_{\text{poly\_u}}(\lambda) = \min_{v_{ij}, x_{ij}, \theta_t, \theta_e} \quad & \sum_{(i,j)\in A} \left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij} \right) x_{ij} + \lambda \sum_{(i,j)\in A} \left(E_{ij}(v_{ij}) \right) x_{ij} - \lambda E_0 \\
& + \sum_{(i,j)\in A} \left(\max\left\{ \sigma_{ij} - \theta_t, 0 \right\} \right) x_{ij} + \Gamma_t \theta_t \\
& + \lambda \sum_{(i,j)\in A} \left(\max\left\{ \tilde{\sigma}_{ij} - \theta_e, 0 \right\} \right) x_{ij} + \lambda \Gamma_e \theta_e \\
\text{s.t.} \quad & \underline{v}_{ij} \leq v_{ij} \leq \overline{v}_{ij}, \quad \forall (i,j) \in A \\
& \theta_t,\, \theta_e \geq 0 \\
& \boldsymbol{x} \in X = \text{set of paths.}
\end{aligned}
\tag{4.6}
$$

Proof of Theorem 4.2. The techniques for the proof are taken from Bertsimas and

Sim [7]. As mentioned in Section 3.2,

$$\max_{\mathbf{\Delta t} \in U_t} \left\{ \sum_{(i,j)\in A} \left(\Delta t_{ij}\right) x_{ij} \right\} = \max_{\{S_t | S_t \subset A, |S_t| \leq \Gamma_t\}} \left\{ \sum_{(i,j)\in S_t} \left(\sigma_{ij}\right) x_{ij} \right\},$$

which can be written as the linear program

$$\begin{aligned} \max_{z_{ij}} \quad & \sum_{(i,j)\in A} \sigma_{ij} x_{ij} z_{ij} \\ \text{s.t.} \quad & \sum_{(i,j)\in A} z_{ij} \leq \Gamma_t \\ & 0 \leq z_{ij} \leq 1 \qquad \forall (i,j) \in A \end{aligned}$$

and its dual is

$$\begin{aligned} \min_{p_{ij},\theta_t} \quad & \sum_{(i,j)\in A} p_{ij} + \Gamma_t \theta_t \\ \text{s.t.} \quad & p_{ij} + \theta_t \geq \sigma_{ij} x_{ij} \quad \forall (i,j) \in A \\ & p_{ij} \geq 0 \qquad\qquad\quad \forall (i,j) \in A \\ & \theta_t \geq 0. \end{aligned} \tag{4.7}$$

Because the primal is feasible and bounded, so is the dual, and they have the

same objective value. Therefore, problem (3.7) is equivalent to:

$$\begin{aligned}
\min_{v_{ij},x_{ij}} \quad & \sum_{(i,j)\in A}\left(\frac{\ell_{ij}}{v_{ij}}+\tau_{ij}\right)x_{ij}+\lambda\sum_{(i,j)\in A}\big(E_{ij}(v_{ij})\big)\,x_{ij}-\lambda E_0 \\
& + \min_{\substack{p_{ij},\theta_t \\ \text{s.t.} \quad p_{ij}+\theta_t\geq\sigma_{ij}x_{ij}, \quad \forall(i,j)\in A \\ p_{ij}\geq 0, \quad \forall(i,j)\in A \\ \theta_t\geq 0}} \left\{\sum_{(i,j)\in A} p_{ij}+\Gamma_t\theta_t\right\} \\
& + \lambda \max_{\{S_e|S_e\subset A,|S_e|\leq\Gamma_e\}}\left\{\sum_{(i,j)\in S_e}\big(\tilde{\sigma}_{ij}\big)\,x_{ij}\right\} \\
\text{s.t.} \quad & \underline{v}_{ij}\leq v_{ij}\leq\overline{v}_{ij}, \quad \forall(i,j)\in A \\
& \boldsymbol{x}\in X=\text{set of paths},
\end{aligned}$$

which is the same as

$$\begin{aligned}
\min_{v_{ij},x_{ij},p_{ij},\theta_t} \quad & \sum_{(i,j)\in A}\left(\frac{\ell_{ij}}{v_{ij}}+\tau_{ij}\right)x_{ij}+\lambda\sum_{(i,j)\in A}\big(E_{ij}(v_{ij})\big)\,x_{ij}-\lambda E_0 \\
& + \sum_{(i,j)\in A} p_{ij}+\Gamma_t\theta_t \\
& + \lambda \max_{\{S_e|S_e\subset A,|S_e|\leq\Gamma_e\}}\left\{\sum_{(i,j)\in S_e}\big(\tilde{\sigma}_{ij}\big)\,x_{ij}\right\} \\
\text{s.t.} \quad & p_{ij}+\theta_t\geq\sigma_{ij}x_{ij}, \quad \forall(i,j)\in A \\
& p_{ij}\geq 0, \quad \forall(i,j)\in A \\
& \theta_t\geq 0 \\
& \underline{v}_{ij}\leq v_{ij}\leq\overline{v}_{ij}, \quad \forall(i,j)\in A \\
& \boldsymbol{x}\in X=\text{set of paths}.
\end{aligned}$$

Applying the same technique to $\max_{\{S_e | S_e \subset A, |S_e| \leq \Gamma_e\}} \left\{ \sum_{(i,j) \in S_e} \left(\tilde{\sigma}_{ij}\right) x_{ij} \right\}$ yields the equivalent optimization problem

$$
\begin{aligned}
L_{\text{poly\_u}}(\lambda) = \min_{v_{ij}, x_{ij}, p_{ij}, \tilde{p}_{ij}, \theta_t, \theta_e} \quad & \sum_{(i,j) \in A} \left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij} \right) x_{ij} \\
& + \lambda \sum_{(i,j) \in A} \left(E_{ij}(v_{ij}) \right) x_{ij} - \lambda E_0 \\
& + \sum_{(i,j) \in A} p_{ij} + \Gamma_t \theta_t \\
& + \lambda \sum_{(i,j) \in A} \tilde{p}_{ij} + \lambda \Gamma_e \theta_e \\
\text{s.t.} \quad & p_{ij} + \theta_t \geq \sigma_{ij} x_{ij}, \quad \forall (i,j) \in A \\
& p_{ij} \geq 0, \quad \forall (i,j) \in A \\
& \theta_t \geq 0 \\
& \tilde{p}_{ij} + \theta_e \geq \tilde{\sigma}_{ij} x_{ij}, \quad \forall (i,j) \in A \\
& \tilde{p}_{ij} \geq 0, \quad \forall (i,j) \in A \\
& \theta_e \geq 0 \\
& \underline{v}_{ij} \leq v_{ij} \leq \overline{v}_{ij}, \quad \forall (i,j) \in A \\
& \boldsymbol{x} \in X = \text{set of paths},
\end{aligned}
\tag{4.8}
$$

Any optimal solution to problem (4.8) must satisfy:

$$
\begin{aligned}
p_{ij} &= \max\left(\sigma_{ij} x_{ij} - \theta_t, 0\right) \\
\tilde{p}_{ij} &= \max\left(\tilde{\sigma}_{ij} x_{ij} - \theta_e, 0\right).
\end{aligned}
$$

In addition, since $x_{ij} \in \{0, 1\}$, p_{ij} and $\tilde{p}_{ij}$ become

$$p_{ij} = \max\left(\sigma_{ij} - \theta_t, 0\right) x_{ij}$$
$$\tilde{p}_{ij} = \max\left(\tilde{\sigma}_{ij} - \theta_e, 0\right) x_{ij}.$$

Substituting this back in problem (4.8) yields the optimization problem (4.6).

□

Algorithm 2 Algorithm for solving problem (4.6).

1. Maximize the lower bound over λ using subgradient ascent. A natural subgradient at λ for optimal values v_{ij}^*, x_{ij}^*, and θ_e^* is:

$$\lambda \left(\sum_{(i,j)\in A} \left(E_{ij}(v_{ij}^*) \right) x_{ij}^* + \sum_{(i,j)\in A} \left(\max\left\{ \tilde{\sigma}_{ij} - \theta_e^*, 0 \right\} \right) x_{ij}^* - E_0 \right) + \lambda \Gamma_e \theta_e^*.$$

   Use the values $\hat{v}_{ij}, \hat{x}_{ij}, \hat{\theta}_e$ found from the steps below as inputs (for v_{ij}^*, x_{ij}^*, and θ_e^*) to find a good direction (even though they might not be optimal).

2. Find the optimal velocities in the same way as for the Lagrangian relaxation for the deterministic case (the uncertainty is not dependent on speed).

3. Use algorithm 1 to find $\hat{\theta}_t$, $\hat{\theta}_e$, and $\hat{\boldsymbol{x}}$.

4.3 Robust Optimization Model: Ellipsoidal Uncertainty

To solve robust optimization problems under ellipsoidal uncertainty, we will need to be able to incorporate it into the problem in way that is tractable (the main framework is discussed in Section 3.2). The problem formulation with Lagrangian relaxation is in Section 4.3.2, and the main idea is to simplify the problem by applying the techniques in Bertsimas and Sim [8] to

$$\max_{\boldsymbol{\Delta t}\in U_t}\left\{\sum_{(i,j)\in A}\left(\Delta t_{ij}\right)x_{ij}\right\} = \max_{\left\{\boldsymbol{\Delta t_{ij}}|\,||\Sigma_t^{-1/2}\boldsymbol{\Delta t_{ij}}||_2\leq\Omega_t\right\}}\left\{\sum_{(i,j)\in A}\left(\Delta t_{ij}\right)x_{ij}\right\}.$$

Using the Karush-Kuhn-Tucker conditions and solving the maximization problem, we find that

$$\max_{\left\{\boldsymbol{\Delta t_{ij}}|\,||\Sigma_t^{-1/2}\boldsymbol{\Delta t_{ij}}||_2\leq\Omega_t\right\}}\left\{\sum_{(i,j)\in A}\left(\Delta t_{ij}\right)x_{ij}\right\} = \Omega_t\sqrt{(\boldsymbol{d}^t)'\boldsymbol{x}},$$

where $\boldsymbol{d}^t$ is the vector whose entry for arc (i,j) is σ_{ij}.

Using this simplification, the basic problem we are solving is of the form

$$\begin{aligned}&\min \boldsymbol{c}'\boldsymbol{x} + \sum_{l=1}^{L} f_l\left((\boldsymbol{d}^l)'\boldsymbol{x}\right)'\boldsymbol{x}\\ &\text{subject to}\quad \boldsymbol{x}\in X\subseteq\{0,1\}^n\end{aligned} \tag{4.9}$$

where:

- $f_l(w_l)$ is concave, for all $l = 1,\ldots,L$.

- $\eta_l(w_l)$ is a subgradient of f_l at w_l, for all $l = 1, \ldots, L$.

Definition 4.1 (Subgradient ($\eta(w)$) of a Concave Function $f(w)$). A *subgradient* of a concave function $f(w)$ is a function $\eta(w)$ that satisfies

$$f(u) - f(w) \leq \eta(w)(u - w), \quad \forall u \in \mathbb{R}$$

■

A basic fact is that subgradient of a concave function is nonincreasing. Namely,

$$w_1 \leq w_2 \Rightarrow \eta(w_1) \geq \eta(w_2).$$

For $f_l\left((\boldsymbol{d}^l)'\boldsymbol{x}\right) = \Omega_l\sqrt{(\boldsymbol{d}^l)'\boldsymbol{x}}$, the subgradient $\eta_l\left(\boldsymbol{d}'\boldsymbol{x}\right)$ is

$$\eta_l\Big((\boldsymbol{d}^l)'\boldsymbol{x}\Big) = \begin{cases} \dfrac{\Omega_l}{2\sqrt{\left((\boldsymbol{d}^l)'\boldsymbol{x}\right)}}, & \text{if } (\boldsymbol{d}^l)'\boldsymbol{x} \neq 0 \\ \dfrac{\Omega_l}{\sqrt{d^l_{\min}}}, & \text{if } (\boldsymbol{d}^l)'\boldsymbol{x} = 0 \end{cases}$$

where $d^l_{\min}$ is the smallest positive element of the vector $\boldsymbol{d}^l$.

Proposition 4.1. The L dimensional vector $\eta(\boldsymbol{w}) = \eta(w_1, w_2, \ldots, w_L)$ is a subgradient of $f(\boldsymbol{w}) = \sum_{l=1}^{L} f_l(\boldsymbol{w})$ at $\boldsymbol{w}$, where the l^{th} component of $\eta(\boldsymbol{w})$ is $\eta_l(w_l)$.

Proof of Proposition 4.1. The proof follows from basic facts about subgradients.

Because $\eta_l(w_l)$ is a subgradient of $f_l(w_l)$ at w_l, we have that

$$f_l(r_l) - f(w_l) \leq \eta_l(w_l)(r_l - w_l), \quad \forall r_l \in \mathbb{L}, \quad \forall l = 1, \ldots, L$$

Adding all of the L inequalities yields

$$\begin{aligned} & \sum_{l=1}^{L} f_l(r_l) - \sum_{l=1}^{L} f_l(w_l) \leq \sum_{l=1}^{L} \eta_l(w_l)(r_l - w_l), \quad \forall \boldsymbol{r} \in \mathbb{R}^L, \\ \Rightarrow \qquad & f(\boldsymbol{r}) - f(\boldsymbol{w}) \leq \eta(\boldsymbol{w})'(\boldsymbol{r} - \boldsymbol{w}), \quad \forall \boldsymbol{r} \in \mathbb{R}^L \end{aligned}$$

□

We extend the Frank-Wolfe type algorithm in Section 5 of Bertsimas and Sim [8] to solve problem (4.9) of L variables. The algorithm is algorithm 3.

Proposition 4.2 (Algorithm 3 Improves the Objective Value At Each Iteration). For any $\theta_l \geq 0$, for all $l = 1, \ldots, L$, Let

$$\begin{aligned} \boldsymbol{x}_0 &= \underset{\boldsymbol{y} \in X}{\operatorname{argmin}} \left\{ \left(\boldsymbol{c} + \sum_{l=1}^{L} \theta_l (\boldsymbol{d}^l) \right)' \boldsymbol{y} \right\} \\ \boldsymbol{x}_1 &= \underset{\boldsymbol{y} \in X}{\operatorname{argmin}} \left\{ \left(\boldsymbol{c} + \sum_{l=1}^{L} \eta_l \left((\boldsymbol{d}^l)' \boldsymbol{x}_0 \right) (\boldsymbol{d}^l) \right)' \boldsymbol{y} \right\} \end{aligned}$$

Then

$$\boldsymbol{c}' \boldsymbol{x}_0 + \sum_{l=1}^{L} f_l \left((\boldsymbol{d}^l)' \boldsymbol{x}_0 \right) \geq \boldsymbol{c}' \boldsymbol{x}_1 + \sum_{l=1}^{L} f_l \left((\boldsymbol{d}^l)' \boldsymbol{x}_1 \right)$$

Algorithm 3 Algorithm for solving problem (4.9).

1. Initialization:

   - Select some $\theta_l \in \left[\eta\left((\boldsymbol{d}^l)'\boldsymbol{e}\right), \eta(0)\right]$, for all $l = 1, \ldots, L$.
   - Let $k = 0$,
   - $\boldsymbol{x}_0 = \underset{\boldsymbol{y}\in X}{\operatorname{argmin}} \left\{ \left(\boldsymbol{c} + \sum_{l=1}^{L} \theta_l (\boldsymbol{d}^l) \right)' \boldsymbol{y} \right\}$

2. Until $\boldsymbol{c}'\boldsymbol{x}_k + \sum_{l=1}^{L} f_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_k\right) = \boldsymbol{c}'\boldsymbol{x}_{k+1} + \sum_{l=1}^{L} f_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_{k+1}\right)$:

$$\boldsymbol{x}_{k+1} = \underset{\boldsymbol{y}\in X}{\operatorname{argmin}} \left\{ \left(\boldsymbol{c} + \sum_{l=1}^{L} \eta_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_k\right)(\boldsymbol{d}^l) \right)' \boldsymbol{y} \right\}$$

   If $L = 1$, terminate step 2 once $\left(\boldsymbol{d}^1\right)' \boldsymbol{x}_{k+1} = \left(\boldsymbol{d}^1\right)' \boldsymbol{x}_k$, instead of checking the condition $\boldsymbol{c}'\boldsymbol{x}_k + f_1\left((\boldsymbol{d}^1)'\boldsymbol{x}_k\right) = \boldsymbol{c}'\boldsymbol{x}_{k+1} + f_1\left((\boldsymbol{d}^1)'\boldsymbol{x}_{k+1}\right)$.

3. Output $\boldsymbol{x}_{k+1}$.

Proof of Proposition 4.2.

$$
\begin{aligned}
\boldsymbol{c}'\boldsymbol{x}_0 + \sum_{l=1}^{L} f_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_0\right) &= \boldsymbol{c}'\boldsymbol{x}_0 + \left(\sum_{l=1}^{L} \eta_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_0\right)(\boldsymbol{d}^l)\right)'\boldsymbol{x}_0 \\
&\quad + \sum_{l=1}^{L} f_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_0\right) - \left(\sum_{l=1}^{L} \eta_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_0\right)(\boldsymbol{d}^l)\right)'\boldsymbol{x}_0 \\
&\geq \boldsymbol{c}'\boldsymbol{x}_1 + \left(\sum_{l=1}^{L} \eta_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_0\right)(\boldsymbol{d}^l)\right)'\boldsymbol{x}_1 \\
&\quad + \sum_{l=1}^{L} f_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_0\right) - \left(\sum_{l=1}^{L} \eta_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_0\right)(\boldsymbol{d}^l)\right)'\boldsymbol{x}_0 \\
&= \boldsymbol{c}'\boldsymbol{x}_1 + \sum_{l=1}^{L} f_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_1\right) - \sum_{l=1}^{L} f_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_1\right) \\
&\quad + \left(\sum_{l=1}^{L} \eta_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_0\right)(\boldsymbol{d}^l)\right)'\boldsymbol{x}_1 \\
&\quad + \sum_{l=1}^{L} f_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_0\right) - \left(\sum_{l=1}^{L} \eta_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_0\right)(\boldsymbol{d}^l)\right)'\boldsymbol{x}_0 \\
&= \boldsymbol{c}'\boldsymbol{x}_1 + \sum_{l=1}^{L} f_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_1\right) \\
&\quad + \sum_{l=1}^{L} f_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_0\right) - \sum_{l=1}^{L} f_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_1\right) \\
&\quad + \left(\sum_{l=1}^{L} \eta_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_0\right)(\boldsymbol{d}^l)\right)'\boldsymbol{x}_1 \\
&\quad - \left(\sum_{l=1}^{L} \eta_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_0\right)(\boldsymbol{d}^l)\right)'\boldsymbol{x}_0 \\
&\geq \boldsymbol{c}'\boldsymbol{x}_1 + \sum_{l=1}^{L} f_l\left((\boldsymbol{d}^l)'\boldsymbol{x}_1\right) \qquad (*)
\end{aligned}
$$

The first inequality uses the optimality of x_1, and the second inequality $(*)$ uses the fact that $\eta\left((d^1)'x_0,\ldots,(d^L)'x_0\right)$ is a subgradient of $f\left((d^1)'x_0,\ldots,(d^L)'x_0\right)$ at $\left((d^1)'x_0,\ldots,(d^L)'x_0\right)'$.

Justifying $(*)$ in more detail:

$$
\begin{aligned}
&\left\{\begin{array}{l}\sum_{l=1}^{L} f_l\left((d^l)'x_0\right)-\sum_{l=1}^{L} f_l\left((d^l)'x_1\right)\\ \quad+\left(\sum_{l=1}^{L}\eta_l\left((d^l)'x_0\right)(d^l)\right)'x_1-\left(\sum_{l=1}^{L}\eta_l\left((d^l)'x_0\right)(d^l)\right)'x_0\end{array}\right\}\\
&=\left\{\begin{array}{l}f\left((d^1)'x_0,\ldots,(d^L)'x_0\right)-f\left((d^1)'x_1,\ldots,(d^L)'x_1\right)\\ \quad+\left(\sum_{l=1}^{L}\eta_l\left((d^l)'x_0\right)(d^l)\right)'(x_1-x_0)\end{array}\right\}\\
&=\left\{\begin{array}{l}f\left((d^1)'x_0,\ldots,(d^L)'x_0\right)-f\left((d^1)'x_1,\ldots,(d^L)'x_1\right)\\ \quad+\sum_{l=1}^{L}\left(\eta_l\left((d^l)'x_0\right)\left((d^l)'x_1-(d^l)'x_0\right)\right)\end{array}\right\}\\
&=\left\{\begin{array}{l}f\left((d^1)'x_0,\ldots,(d^L)'x_0\right)-f\left((d^1)'x_1,\ldots,(d^L)'x_1\right)\\ \quad+\begin{pmatrix}\eta_1\left((d^1)'x_0\right)\\ \vdots\\ \eta_L\left((d^L)'x_0\right)\end{pmatrix}'\left(\begin{pmatrix}(d^1)'x_1\\ \vdots\\ (d^L)'x_1\end{pmatrix}-\begin{pmatrix}((d^1)'x_0\\ \vdots\\ (d^L)'x_0\end{pmatrix}\right)\end{array}\right\}\\
&\geq 0
\end{aligned}
$$

Where the inequality is a direct application of the definition of a subgradient for a convex function. □

4.3.1 Candidate Solutions

We consider two additional candidate solutions:

Minimize Time—Robust Case with Ellipsoidal Uncertainty: We ignore the energy constraint, and minimize time, using the ellipsoidal uncertainty set, which becomes the optimization problem

$$\begin{aligned} \min_{v_{ij}, x_{ij}} \quad & \sum_{(i,j)\in A} \left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij} \right) x_{ij} + \Omega_1 \sqrt{\sum_{(i,j)\in A} \left(\sigma_{ij}\right) x_{ij}} \\ \text{s.t.} \quad & \underline{v}_{ij} \leq v_{ij} \leq \overline{v}_{ij}, \quad \forall (i,j) \in A \\ & \boldsymbol{x} \in X \subset \{0,\, 1\}^m, \end{aligned}$$

which is of the form (4.9) with $L = 1$. Like the deterministic case, the optimal choice of arc velocities is to set each arc velocity to its corresponding upper bound (go as fast as possible). We use algorithm 3 to heuristically find a good path.

In more detail, The subgradient $\eta\left(\boldsymbol{d}'\boldsymbol{x}\right)$ is

$$\eta\left(\boldsymbol{d}'\boldsymbol{x}\right) = \begin{cases} \dfrac{\Omega_1}{2\sqrt{\left(\boldsymbol{d}'\boldsymbol{x}\right)}}, & \text{if } \boldsymbol{d}'\boldsymbol{x} \neq 0 \\ \dfrac{\Omega_1}{\sqrt{d_{\min}}}, & \text{if } \boldsymbol{d}'\boldsymbol{x} = 0 \end{cases}$$

where $d_{\min}$ is the smallest positive element of the vector $\boldsymbol{d}$. Therefore, the shortest path problems are of the form $\min_{\boldsymbol{y}} \left\{ \left(\boldsymbol{c} + \eta\left(\boldsymbol{d}'\boldsymbol{x}_k\right)\boldsymbol{d}\right)' \boldsymbol{y} \right\}$ which have

arc costs

$$\left(\frac{\ell_{ij}}{v^*_{ij}} + \tau_{ij}\right) + \eta\left(\boldsymbol{d}'\boldsymbol{x}_k\right)\left(\sigma_{ij}\right).$$

When solving this shortest path problem, we can use the same A-star heuristic function as for the deterministic minimum-time problem (Section 4.4.1).

Minimize Energy—Robust Case with Ellipsoidal Uncertainty: We ignore the energy constraint, and minimize energy, using the polyhedral uncertainty set, which becomes an optimization problem

$$\begin{aligned}
\min_{v_{ij}, x_{ij}} \quad & \sum_{(i,j)\in A} \left(E_{ij}(v_{ij})\right) x_{ij} + \Omega_2 \sqrt{\sum_{(i,j)\in A} \left(\tilde{\sigma}_{ij}\right) x_{ij}} \\
\text{s.t.} \quad & \underline{v}_{ij} \le v_{ij} \le \overline{v}_{ij}, \quad \forall (i,j) \in A \\
& \boldsymbol{x} \in X \subset \{0,1\}^m,
\end{aligned}$$

which is of the form (4.9) with $L = 1$. Because the uncertainty does not depend on arc velocity, the optimal choice of arc velocities is the same as the deterministic case (solve $\min_{\underline{v}_{ij} \le v_{ij} \le \overline{v}_{ij}} \left\{E_{ij}(v_{ij})\right\}$). Then, use algorithm 3 to heuristically find a good path.

In more detail, the subgradient $\eta\left(\boldsymbol{d}'\boldsymbol{x}\right)$ is

$$\eta\left(\boldsymbol{d}'\boldsymbol{x}\right) = \begin{cases} \dfrac{\Omega_2}{2\sqrt{\left(\boldsymbol{d}'\boldsymbol{x}\right)}}, & \text{if } \boldsymbol{d}'\boldsymbol{x} \neq 0 \\ \dfrac{\Omega_2}{\sqrt{d_{\min}}}, & \text{if } \boldsymbol{d}'\boldsymbol{x} = 0 \end{cases}$$

where $d_{\min}$ is the smallest positive element of the vector $\boldsymbol{d}$.

Therefore, the shortest path problems are of the form $\min_{\boldsymbol{y}} \left\{ \left(\boldsymbol{c} + \eta \left(\boldsymbol{d}' \boldsymbol{x}_k \right) \boldsymbol{d} \right)' \boldsymbol{y} \right\}$ which have arc costs

$$E_{ij}(v_{ij}^*) + \eta \left((\boldsymbol{d}^2)' \boldsymbol{x}_k \right) \left(\tilde{\sigma}_{ij} \right).$$

When solving this shortest path problem, we can use the same A-star heuristic function as for the deterministic minimum-energy problem (Section 4.4.1), where we make sure to use the equivalent formulation of energy consumption and ignore the constant $\frac{mg}{\eta}(h_t - h_s)$.

As with the deterministic case, the optimal robust minimum-energy solution is the path with the least (planned or anticipated) energy consumption, which will always be feasible, if a feasible solution exists. Also, the anticipated energy of the optimal robust minimum-time solution is the most energy any optimal solution would anticipate consuming.

4.3.2 Solving the Lagrangian Relaxation

The full formulation (problem (3.7)) can be transformed into the following equivalent problem below (as described in Section 4.3):

$$\begin{aligned}
\min_{v_{ij},x_{ij}} \quad & \sum_{(i,j)\in A}\left(\frac{\ell_{ij}}{v_{ij}}+\tau_{ij}\right)x_{ij}+\Omega_t\sqrt{\sum_{(i,j)\in A}\left(\sigma_{ij}\right)x_{ij}} \\
& +\lambda\left(\sum_{(i,j)\in A}\left(E_{ij}(v_{ij})\right)x_{ij}+\Omega_e\sqrt{\sum_{(i,j)\in A}\left(\tilde{\sigma}_{ij}\right)x_{ij}}-E_0\right) \qquad (4.10)\\
\text{s.t.} \quad & \underline{v}_{ij}\le v_{ij}\le \overline{v}_{ij}, \quad \forall (i,j)\in A \\
& \boldsymbol{x}\in X\subset\{0,1\}^m.
\end{aligned}$$

It can be solved using algorithm 4.

4.4 Using A-Star

The A-star algorithm is a search algorithm that guarantees optimality, provided that an appropriate heuristic function $g(i)$ is chosen. An appropriate $g(i)$ is an underestimate of the shortest path distance from node i to node t satisfying the condition

$$g(i)\le g(j)+c_{ij},\ \forall (i,j)\in A.$$

- In the language of artificial intelligence: A-star uses a consistent and admissible heuristic $g(i)$, and the label of node i is the current shortest distance from s to node i plus $g(i)$.

Algorithm 4 Algorithm for solving problem (4.10).

- Maximize the lower bound over λ using subgradient ascent. A natural subgradient at λ for optimal values v_{ij}^* and x_{ij}^* is:

$$\sum_{(i,j)\in A} \left(E_{ij}(v_{ij}^*) \right) x_{ij}^* + \Omega_e \sqrt{\sum_{(i,j)\in A} \left(\tilde{\sigma}_{ij} \right) x_{ij}^*} - E_0.$$

  Use the values $\hat{v}_{ij}, \hat{x}_{ij}$ found from the algorithm below as inputs (for v_{ij}^* and x_{ij}^*) to find a good direction (even though they might not be optimal).

- We solve for the optimal velocities as in the deterministic case, as the uncertainty does not depend on arc velocity. Technically, we need to compute the velocities only once for each value of λ, as the optimal velocities remain the same throughout each iteration of algorithm 3.

- Apply algorithm 3 for $L = 2$ to obtain a good (but not necessarily optimal) path $\hat{\boldsymbol{x}}$.

- In the language of network optimization: A-star is Dijkstra's algorithm on a graph where $g(i)$ is a (well-chosen) dual feasible solution and the graph uses the reduced costs with dual variable $g(i)$ for each node i.

A-star stops once the true shortest path distance from s to t is known (that is, when node t is permanently labeled). See [38] for more information about the A-star algorithm, and see Sections 4.4.1 and 4.4.2 for our choice of the function $g(i)$.

Using these modifications substantially speeds up the shortest path computation. First, eliminating negative arc weights allows us to use algorithms like Dijkstra's algorithm (which requires non-negative arc weights), instead of needing to use an algorithm like the first-in-first-out label-correcting algorithm in Chapter 5 of Ahuja et al. [1] to account for negative arc weights. Second, because we only care about finding a path from the source to sink (instead of finding shortest paths from one source to all other nodes), we can stop the algorithm once an optimal path from the source to destination is found, even if the algorithm did not look at all of the arcs in the network. Using A-star helps guide the algorithm to check the arcs that are more likely to be on an optimal path from the source to the destination, which more quickly finds the optimal path and prevents the algorithm from checking some unnecessary arcs.

4.4.1 Deterministic Case

The first step is to reformulate the problem to eliminate negative arc weights (and keep all of the arcs in the network). While the time component is always

nonnegative, the energy consumption might be negative due to elevation (going downhill). Therefore, the total energy consumption on any path P from source s to sink t is

$$\begin{aligned}\sum_{(i,j)\in P} E_{ij}(v_{ij}) = & \sum_{(i,j)\in P}\left(\frac{\ell_{ij}}{\eta}\left(\frac{1}{2}\rho C_w A_f v_{ij}^2 + \mu mg\cos(\alpha_{ij}) + mg\sin(\alpha_{ij})\right)\right) \\ & + \sum_{(i,j)\in P} P_{\text{acc}}\left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij}\right) \\ = & \sum_{(i,j)\in P}\left(\frac{\ell_{ij}}{\eta}\left(\frac{1}{2}\rho C_w A_f v_{ij}^2 + \mu mg\cos(\alpha_{ij})\right)\right) \\ & + \sum_{(i,j)\in P}\frac{\ell_{ij}}{\eta}\left(mg\sin(\alpha_{ij})\right) + \sum_{(i,j)\in P} P_{\text{acc}}\left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij}\right).\end{aligned}$$

The first summation is always nonnegative, as $\cos(\alpha_{ij})$ is nonnegative for $-\pi/2 \leq \alpha_{ij} \leq \pi/2$. We simplify the second summation. The road grade α_{ij} can be computed as $\arcsin\left(\frac{h_j - h_i}{\ell_{ij}}\right)$, where h_i is the elevation of node i. The calculation $\arcsin\left(\frac{h_j - h_i}{\ell_{ij}}\right)$ is exact or a close approximation for small angles, depending on whether ℓ_{ij} is interpreted as the hypotenuse or base of a right triangle for the

trigonometric calculations. Using this computation,

$$\begin{aligned}\sum_{(i,j)\in P} \frac{\ell_{ij}}{\eta}\left(mg\sin\left(\alpha_{ij}\right)\right) &= \sum_{(i,j)\in P} \frac{\ell_{ij}}{\eta}\left(mg\sin\left(\arcsin\left(\frac{h_j-h_i}{\ell_{ij}}\right)\right)\right)\\ &= \sum_{(i,j)\in P} \frac{\ell_{ij}}{\eta}\left(mg\left(\frac{h_j-h_i}{\ell_{ij}}\right)\right)\\ &= \frac{mg}{\eta}\sum_{(i,j)\in P}\left(h_j-h_i\right)\\ &= \frac{mg}{\eta}\left(h_t-h_s\right).\end{aligned}$$

Hence,

$$\begin{aligned}\sum_{(i,j)\in P} E_{ij}(v_{ij}) = \sum_{(i,j)\in P}&\left(\frac{\ell_{ij}}{\eta}\left(\frac{1}{2}\rho C_w A_f v_{ij}^2+\mu mg\cos\left(\alpha_{ij}\right)\right)+P_{\text{acc}}\left(\frac{\ell_{ij}}{v_{ij}}+\tau_{ij}\right)\right)\\ &+\frac{mg}{\eta}\left(h_t-h_s\right).\end{aligned}$$

As a result, the Lagrangian relaxation model for finding a path from s to t with

nonnegative arc weights for the deterministic case is:

$$
\begin{aligned}
L(\lambda) = \min_{v_{ij}, x_{ij}} \quad & \left\{ \begin{array}{l} \displaystyle\sum_{(i,j)\in A} \left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij} \right) x_{ij} \\ \\ \displaystyle + \lambda \sum_{(i,j)\in A} \left(\begin{array}{l} \displaystyle\frac{\ell_{ij}}{\eta} \left(\frac{1}{2} \rho C_w A_f v_{ij}^2 + \mu m g \cos(\alpha_{ij}) \right) \\ \displaystyle + P_{\text{acc}} \left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij} \right) \end{array} \right) x_{ij} \\ \\ \displaystyle - \lambda E_0 - \lambda \frac{mg}{\eta} (h_t - h_s) \end{array} \right\} \\
& \underline{v}_{ij} \leq v_{ij} \leq \overline{v}_{ij}, \quad \forall (i,j) \in A \\
& \boldsymbol{x} \in X \subset \{0,1\}^m.
\end{aligned}
$$

To solve this problem, we apply the same method as in Section 4.1.2, and we find the arc velocities v_{ij}^* on an as-needed basis. Namely, we compute v_{ij}^* only when computing the cost of arc (i,j) (technically, we need to compute v_{ij}^* only once per arc per value of λ).

To compute v_{ij}, we use Theorem 4.1 as before, because $\dfrac{mg}{\eta}(h_t - h_s)$ does not depend on v_{ij}.

We now need to compute the heuristic function $g(i)$ for the objective function for the shortest path problem:

$$
\left\{ \begin{array}{l} \displaystyle\sum_{(i,j)\in A} \left(\frac{\ell_{ij}}{v_{ij}^*} + \tau_{ij} \right) x_{ij} \\ \\ \displaystyle + \lambda \sum_{(i,j)\in A} \left(\begin{array}{l} \displaystyle\frac{\ell_{ij}}{\eta} \left(\frac{1}{2} \rho C_w A_f \left(v_{ij}^* \right)^2 + \mu m g \cos(\alpha_{ij}) \right) \\ \displaystyle + P_{\text{acc}} \left(\frac{\ell_{ij}}{v_{ij}^*} + \tau_{ij} \right) \end{array} \right) x_{ij} \end{array} \right\}.
$$

To compute this heuristic function, we underestimate time and energy separately. For a constant speed, both time and energy increase as the distance traveled increases; therefore, we estimate $g(i)$ based on the minimum possible distance from node i to t. If we have x and y distances in meters (by using a map projection), the geometrical distance between node i and node t can be computed using the Pythagorean theorem. Otherwise, we can use the haversine formula to compute the distance between two points with given latitude and longitude coordinates (for example, Sinnott [41]). This geometrical distance, $\ell g(i)$, is an underestimate of the shortest path distance from node i to t.

Estimate for Time: To obtain a lower bound on time (as an estimate), we use the maximum possible velocity, v^g_{time}. We also underestimate the time due to stops and turns. We assume for this heuristic that no time is lost due to stopping or turns.

Estimate for Energy: As described when finding the deterministic minimum-energy candidate solution in Section 4.1.1, the (unconstrained) velocity at which energy is minimized is

$$v^g_{\text{en}} = \sqrt[3]{\frac{\eta P_{\text{acc}}}{\rho C_D A_f}}.$$

Let α^{hg} be the value of α with the smallest value of $\cos(\alpha)$ from all of the arcs in the data.

Therefore, the heuristic function for A-star that we suggest is

$$g(i) = \ell^g(i) \left(\frac{1}{v^g_{\text{time}}} + \lambda \left(\begin{array}{l} \frac{1}{\eta} \left(\frac{1}{2} \rho C_w A_f \left(v^g_{\text{en}} \right)^2 + \mu m g \cos\left(\alpha^g \right) \right) \\ + P_{\text{acc}} \left(\frac{1}{v^g_{\text{en}}} \right) \end{array} \right) \right). \quad (4.11)$$

We now prove that $g(i)$ in (4.11) is *consistent*: $g(i) \leq g(j) + c_{ij}$, $\forall (i,j) \in A$.

Proof. The proof follows from the triangle inequality of geometric ("straight line") distances and the fact that $g(j)$ is an underestimate for the true cost from going from node j to the end:

$$\begin{aligned}
g(i) &= \ell^g(i) \left(\frac{1}{v^g_{\text{time}}} + \lambda \left(\begin{array}{l} \frac{1}{\eta} \left(\frac{1}{2} \rho C_w A_f \left(v^g_{\text{en}} \right)^2 + \mu m g \cos\left(\alpha^g \right) \right) \\ + P_{\text{acc}} \left(\frac{1}{v^g_{\text{en}}} \right) \end{array} \right) \right) \\
&\leq \left(\ell^g(j) + \ell_{ij} \right) \left(\frac{1}{v^g_{\text{time}}} + \lambda \left(\begin{array}{l} \frac{1}{\eta} \left(\frac{1}{2} \rho C_w A_f \left(v^g_{\text{en}} \right)^2 + \mu m g \cos\left(\alpha^g \right) \right) \\ + P_{\text{acc}} \left(\frac{1}{v^g_{\text{en}}} \right) \end{array} \right) \right) \\
&= g(j) + \ell_{ij} \left(\frac{1}{v^g_{\text{time}}} + \lambda \left(\begin{array}{l} \frac{1}{\eta} \left(\frac{1}{2} \rho C_w A_f \left(v^g_{\text{en}} \right)^2 + \mu m g \cos\left(\alpha^g \right) \right) \\ + P_{\text{acc}} \left(\frac{1}{v^g_{\text{en}}} \right) \end{array} \right) \right) \\
&\leq g(j) + \ell_{ij} \left(\frac{1}{v^*_{ij}} + \lambda \left(\begin{array}{l} \frac{1}{\eta} \left(\frac{1}{2} \rho C_w A_f \left(v^*_{ij} \right)^2 + \mu m g \cos\left(\alpha_{ij} \right) \right) \\ + P_{\text{acc}} \left(\frac{1}{v^*_{ij}} \right) \end{array} \right) \right) \\
&= g(j) + c_{ij}.
\end{aligned}$$

In the second-to last step of the proof, we used the fact that v^g_{time} and v^g_{en} and

α^g underestimate the total deterministic time and energy, respectively, compared to using any other arc speed and arc angle. □

4.4.2 A-Star Heuristic Function for Robust Cases and Candidate Solutions

We can use the heuristic function in (4.11) for the robust optimization problems. In addition, we modify the heuristic slightly to find various candidate solutions:

Minimize Time: Use only the travel time part of $g(i)$.

Minimize Energy: Use only the part of $g(i)$ that estimates energy.

Minimize Length: $g(i) = \ell^g(i)$.

In the last step of the proof of consistency for the deterministic case, $=$ is replaced by $\leq$ for the robust cases, as we add a nonnegative term to the second to last line to obtain the cost for each particular robust optimization formulation (0 is used as the underestimate for the robust part of the cost).

For the candidate solutions, a similar proof can be carried out, using the fact that we separately underestimate time, energy, and length (and use 0 for the parts that are not optimized for each candidate solution).

4.5 Pseudo-Polynomial Time Algorithms

In this section, we present two pseudo-polynomial time algorithms for certain variations of the electric vehicle routing problem. One is a dynamic programming algorithm and one uses an expanded network. While these algorithms might not

be as practical as the other algorithms in this chapter, they show that the electric vehicle routing problem is weakly NP-Hard, and that the decision version is weakly NP-Complete (the proof of NP-Hardness is in Section 3.1.2).

Both algorithms assume that $v_{ij} \in V_{ij}$ for all $(i,j) \in A$, where V_{ij} is a set of velocities for arc (i,j), and V is the set of velocities for all of the arcs. This means that the energy consumed when traveling over each arc is one of $|V|+1$ possibilities, where the plus 1 is for an energy consumption of ∞ (indicating that it is not feasible to take an arc). Therefore, without loss of generality, we will assume that the set of possible energy consumptions over arc (i,j) is $\tilde{E}_{ij}$ and that the set of all possible energy consumptions in total is $\tilde{E}$.

$\left|\tilde{E}\right|$ can be exponential in $|N|$, where N is the set of nodes, even for small values of $|V|$. For example, if $|V| = 2$, for a single path with $|N| = n$ nodes, then $\left|\tilde{E}\right|$ could be as large as 2^{n-1}.

If the traveller can use different velocities over each arc, say in the velocity set V_{ij}, then we take $V = \cup_{(i,j)\in A} V_{ij}$ and set the energy consumption of arc (i,j) to be $E_{ij}(v)$ if $v \in V_{ij}$ and ∞ otherwise.

4.5.1 Dynamic Programming Algorithm

We use a dynamic programming approach as in Section 2.1 of Bertsekas [6]:

State $(i, \tilde{e})$**:**

- i is the node we are currently at, $i = 1, \ldots, N$
- $\tilde{e}$ is the remaining energy available, $\tilde{e} \in \tilde{E}$.

Stage (k)**:** The maximum number of arcs remaining to get from node i to node $t = N$ (at most $N - k$ arcs were used so far). Note that we can stay at node i when going from stage k to stage $k + 1$.

Cost to Go Function $J_k(i, \tilde{e})$**:** $J_k(i, \tilde{e})$ is the minimum cost of a path from node i to node $t = N$ using at most $N - k$ arcs with $\tilde{e}$ energy remaining.

We can now write the cost to go function.

Initial Conditions:

$$J_N(i, \tilde{e}) = \infty, \quad i \neq t, \quad \tilde{e} \in \tilde{E}$$
$$J_k(i, \tilde{e}) = \infty, \quad \tilde{e} \notin \tilde{E}, \quad k = 0, 1, \ldots, N$$
$$J_N(t, \tilde{e}) = 0, \quad (i = t), \quad \tilde{e} \in \tilde{E}$$

Bellman Equation:

$$J_k(i, \tilde{e}) = \min \left(\min_{\substack{j \in \{j:(i,j) \in A\} \\ v_{ij} \in V_{ij} \\ \tilde{e} - E_{ij}(v_{ij}) \geq 0}} \left\{ \frac{\ell_{ij}}{v_{ij}} + \tau_{ij} + J_{k+1}\left(j, \tilde{e} - E_{ij}(v_{ij})\right) \right\}, J_{k+1}(i, \tilde{e}) \right)$$
$$i = 1, 2 \ldots, N, \quad k = 0, 1, 2, \ldots, N - 1, \quad \tilde{e} \in \tilde{E}$$

This is $J_k(i, \tilde{e}) =$ the minimum of using arc (i, j) to go to node j from node i (at velocity v_{ij}), or staying at node i (and allowing the path to use one less arc).

Here: $E_{ij}(v_{ij})$ is the energy function (3.1):

$$E_{ij}(v_{ij}) = \frac{\ell_{ij}}{\eta}\left(\frac{1}{2}\rho C_w A_f v_{ij}^2 + \mu mg\cos(\alpha_{ij}) + mg\sin(\alpha_{ij})\right) + P_{\text{acc}}\left(\frac{\ell_{ij}}{v_{ij}} + \tau_{ij}\right).$$

Optimal Cost: The optimal cost of a least-cost path from i to node $t = N$ using at most $\tilde{e}$ units of energy is

$$J^*(i,\tilde{e}) = J_0(i,\tilde{e})$$

Complexity: This dynamic program has $N\left|\tilde{E}\right|$ states, $N+1$ stages, and each minimization requires checking up to $N|V|$ terms (there are at most $|\{j : (i,j) \in A\}| + 1$ terms in the minimization for node i).

4.5.2 An Expanded Network Approach

We can expand the network using a dynamic flows approach as in Section 19.6 of Ahuja et al. [1].

Network: A description of the expanded network is below.

- $G = (N, A)$ is the original directed graph as before.
- $G' = (N', A')$ is the new expanded graph.
- For each node i in G, make $\left|\tilde{E}\right|$ copies of it, where $i_{\tilde{e}}$ is in G' if and only if $\tilde{e} \in \tilde{E}$. $i_{\tilde{e}}$ is the node representing that we are at node i in G and we have $\tilde{e}$ units of energy left to get to the final destination.

- Include an arc $(i_{\tilde{e}}, j_{\tilde{e}'})$ in G' if and only if there is an arc (i, j) in G and there is some velocity $v_{ij} \in V_{ij}$ such that $\tilde{e} - \tilde{e}'$ units of energy are consumed. Give this arc cost $c_{ij} = \dfrac{\ell_{ij}}{v_{ij}} + \tau_{ij}$.
- Include a source node s' and a sink node t' in G'.
- Include an arc $(s', 1_{E_0})$ in G' with cost 0.
- For each $\tilde{e} \in \tilde{E}$, make an arc $(N_{\tilde{e}}, t')$ in G' with cost 0.

Getting an Optimal Path: Solving a static shortest path problem over G' will give a minimum cost path. Then, an optimal path in G is to take the nodes in the path in G', omitting s', t', and ignoring the values of $\tilde{e}$ on the nodes. Because the only outgoing arc from s' is $(s', 1_{E_0})$, we will use at most E_0 units of energy.

Note that a path in the expanded network corresponds to a walk in the original network. However, a path in the expanded network will correspond to a cycle in the original network only if there is a negative-cost cycle in the original network.

Complexity: There are $(N\left|\tilde{E}\right| + 2)$ nodes and at most $|A|\,|V|\left|\tilde{E}\right|$ arcs in G'. If we use a heap implementation of Dijkstra's algorithm, the total complexity is $O\left(|A|\,|V|\left|\tilde{E}\right| + N\left|\tilde{E}\right| \log\left(N\left|\tilde{E}\right|\right)\right)$. However, because this graph is a directed acyclic graph, we can solve this problem using the algorithm for shortest path in acyclic graphs, which implies a complexity of $O\left(|A|\,|V|\left|\tilde{E}\right|\right)$.

Chapter 5

Data and Computations

In this chapter, we apply the algorithms in Chapter 4 to real-world examples. First, we describe the data we use to construct the problem instances (Section 5.1). Then, in Section 5.2, we describe the method for doing the computations and show the results based on example trips in Massachusetts (Section 5.2.1) and Michigan (Section 5.2.2).

5.1 Data

We use three types of data:

- Road network data that we extract from the Esri StreetMap North America data for ArcGIS Desktop 10.1 ([45]). It is from the detailed streets data file.
- Elevation data from the U.S Geological Survey ([48]), obtained from the seamless viewer (`http://nationalmap.gov/viewer.html`). It is the "National Elevation Dataset 1/3 Arc-Second (NED 1/3)" and has approximately a 10-

meter resolution (10 meters by 10 meters raster cells).

- Map data, which is used as a basemap for some of the pictures. It is from the Esri World Street Map basemap ([16]).

For most of the coefficients in the energy model, we use proprietary numerical values provided by Ford (reasonable, but not exact, values for most coefficients can also be found in the literature). Estimates were made for the other parameters in the problem when the appropriate data was not available.

We apply uncertainty to the arcs in the following way:

Time: $\sigma_{ij} = 0.2\left(\frac{\ell_{ij}}{\overline{v}_{ij}} + \tau_{ij}\right)$

Energy: $\tilde{\sigma}_{ij} = 0.2E_{ij}(\overline{v}_{ij})$

5.2 Computations

The computations were implemented in Java and used the Java Universal Network/Graph Framework (JUNG) (`http://jung.sourceforge.net/index.html`), and pictures of the routes were drawn using Unfolding for Eclipse (`http://unfoldingmaps.org`) with the OpenStreetMap map provider ([34]).

Some insights illustrated by the computations in Sections 5.2.1 and 5.2.2 are:

- We can compute a Pareto frontier, illustrating a time-energy tradeoff between various paths. Sometimes, changing routes save a significant amount of energy while requiring only a little more time.

- Applying the estimates from Lagrangian relaxation suggests that the solutions from our algorithms are near-optimal, if not optimal.

- The computations run quickly. This is primarily due to our application of Lagrangian relaxation, particular choice of algorithms to solve the robust optimization problems, and our use of A-star to solve the shortest path problems that arise during the computations.

- Using robust optimization protects the driver from uncertainty. In particular, there will be plenty of energy to make it to the destination, even under uncertainty, and the tradeoff will be only a modest amount of time.

To produce the computations, we first computed the minimum time and minimum energy solutions (incorporating the relevant uncertainty). The total energy consumed (anticipated energy consumed, when solving the robust optimization problems) of the minimum energy solution was set as the lower energy limit, and the energy consumed (anticipated energy consumed, when solving the robust optimization problems) by the minimum time solution was set as the upper energy limit. We then solved 60 instances (one for each particular energy limit), varying the energy limit in equal amounts from the lower limit to the upper limit.

5.2.1 A Trip from Belmont, MA to MIT

We apply the algorithms for polyhedral uncertainty to an example trip from Belmont, MA to MIT. The underlying network has 38,713 nodes and 80,711 arcs.

The computations illustrate that we can compute a Pareto frontier representing the time-energy tradeoff of various routes. The trip time and energy consumption shown in the Pareto frontier in Figure 5.1 are the trip times and energy consumption that the algorithm for robust optimization anticipates (assuming a

worst-case scenario within the restrictions from the uncertainty sets). An example of the time-energy tradeoff is illustrated by paths 5 and 6 in Figure 5.1. Path 6 (the robust minimum-time path) has an anticipated travel time of 20 minutes and 38 seconds, and would require (an anticipated) 1.76 kWh. If the driver would be willing to use path 5 and take an additional 21 seconds (1.7% more time), the (anticipated) energy consumption would be reduced by 0.30 kWh (16.8%). As shown in Figure 5.2, the routes on the Pareto frontier (Figure 5.1) are different, although some routes have substantial overlap. The multiple points on the Pareto Frontier that correspond to one path indicate that changing the vehicle speeds on the arcs also affects the total energy consumption, such as path numbers 1, 3, and 5 in Figure 5.1. Generally, going faster will save time and require more energy, and this change might be significant. For example, going faster on path 1 from energy-optimal velocities to maximum velocities leads to a 2 minute 40 second (anticipated) decrease in time (10.3%) at an additional (anticipated) energy consumption of 0.03 kWh (2.6%).

As shown in Figure 5.1, the Lagrangian relaxation does a good job of estimating the Pareto Frontier for the robust case, under polyhedral uncertainty. While the lower estimates are technically not lower bounds for the robust optimization cases, as we use heuristics to solve the (minimization) subproblem, they still suggest that the solutions returned by the algorithm are quite good. The estimates from Lagrangian relaxation form a curve that seems to very closely fit to the feasible solutions. In addition, the percentage gap between the estimate and the (anticipated) travel time of the best feasible solution found was often less than 5% and sometimes near 0%, although it was as high as 16% for one instance. As

shown in Figure 5.1, there is a significant difference in the (anticipated) travel time of the best feasible solution compared to the estimate from the Lagrangian relaxation when the energy limit is very low. For the deterministic case, we solve the (minimization) subproblem to optimality, and the Lagrangian relaxation provides a true lower bound on the objective value of the main optimization problem. The feasible solutions on the Pareto frontier and the close fit of the bounds from Lagrangian relaxation indicate that the solutions our algorithms find are near-optimal (see Figure 5.11). Like the robust case, the bounds from Lagrangian relaxation seem to almost fit a curve to the feasible solutions in the plot. The percentage gap between the lower bound from Lagrangian relaxation and the travel time of the best feasible solution found is often less than 5% and sometimes near 0%, although it was as high as 20% for one instance. As shown in Figure 5.11, there is a significant difference in the travel time of the best feasible solution compared to the estimate from the Lagrangian relaxation when the energy limit is very low.

The algorithm runs in less than 2 seconds of CPU time for each of the instances, as shown in Figure 5.3. The algorithms were run on a MacBook Pro with a 2.0 GHz Intel Core i7 processor (4 cores) and 8 GB 1333 MHz DDR3 RAM, where the maximum heap size was set to 3 GB.

We also conducted simulations that highlight the advantage of using a robust formulation. Under some randomness, the deterministic solution might require more than the available energy for the driver to reach the destination, although it is faster than the robust solution. For example, Figures 5.4 (energy) and 5.5 (time) show the results from a simulation for the case when the energy limit is set to be

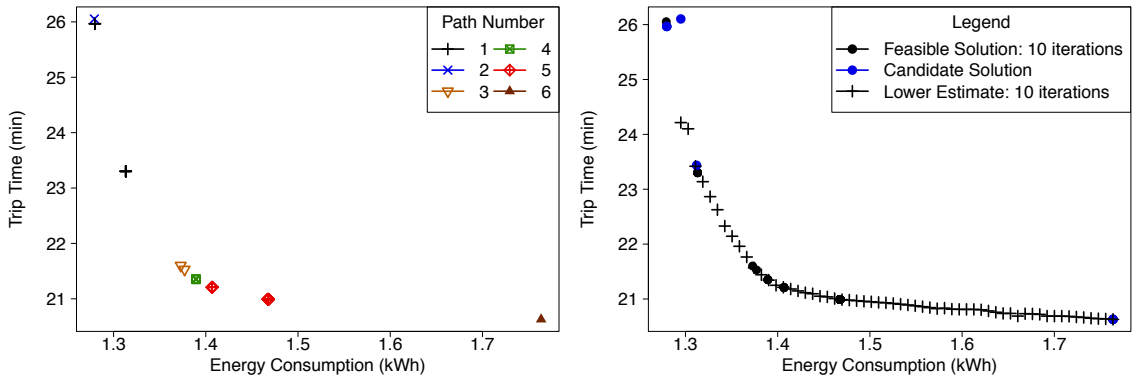

Figure 5.1: Left: The Pareto frontier, excluding candidate solutions, for the trip from Belmont, MA to MIT, using robust optimization under polyhedral uncertainty and allowing for 10 iterations of subgradient ascent. Right: The estimates of the robust Pareto frontiers suggest that our algorithm provides near-optimal solutions.

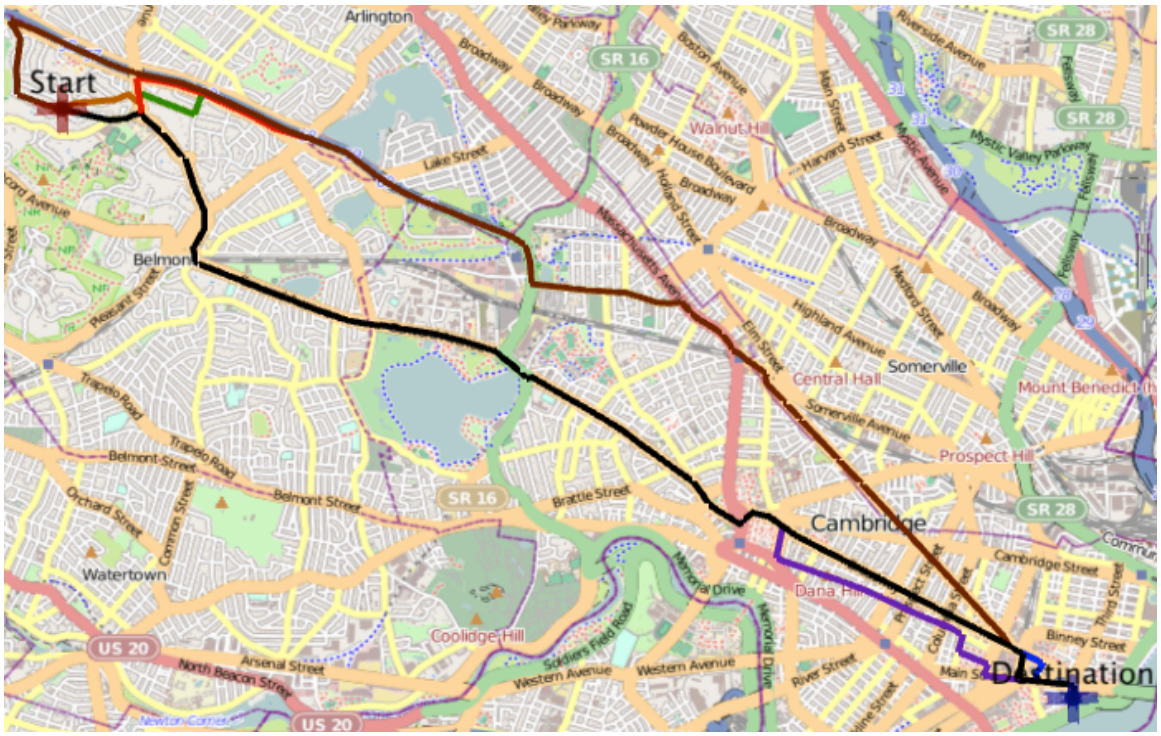

Figure 5.2: The paths produced for the trip from Belmont, MA to MIT using the robust optimization framework for polyhedral uncertainty, illustrated over OpenStreetMap data ([34]) using Unfolding for Eclipse.

1.53 kWh. While the path found by solving the deterministic case consumes too much energy in 38% of the trials, the robust route has enough energy for all of the trials. While the robust route (computed under polyhedral uncertainty) requires more time, the difference of the means of the simulated travel times is about 0.37 minutes (about 22 seconds).

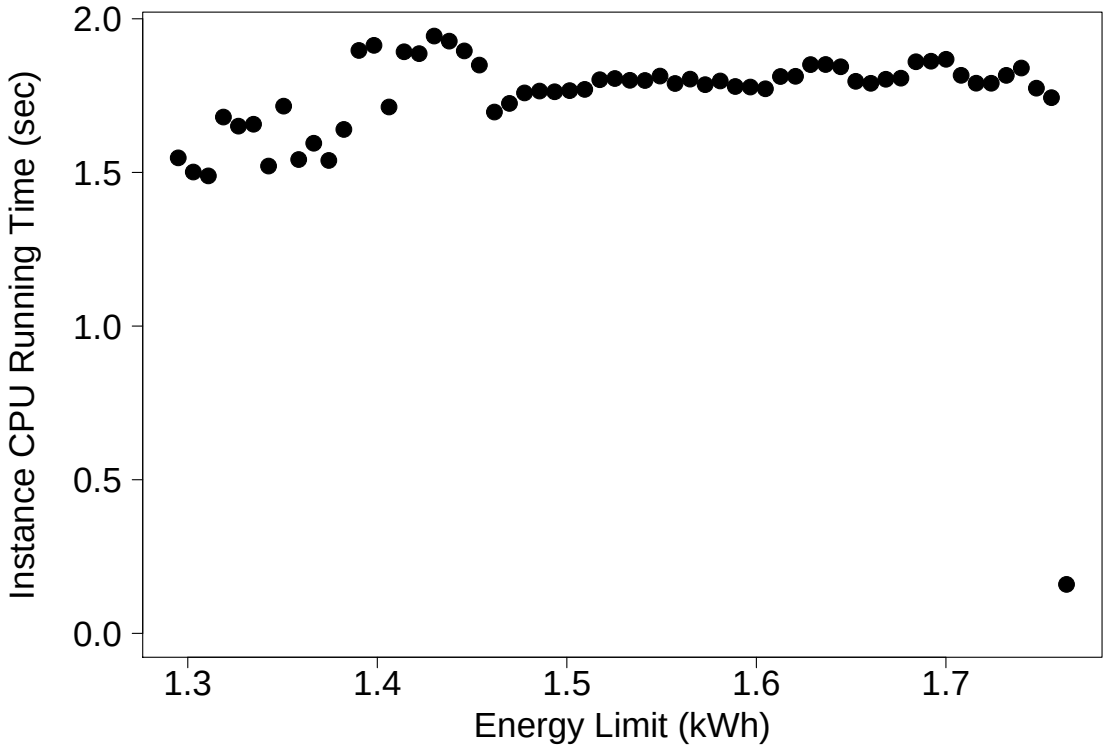

Figure 5.3: The CPU running times, in seconds, for the trip from Belmont, MA to MIT, for polyhedral uncertainty.

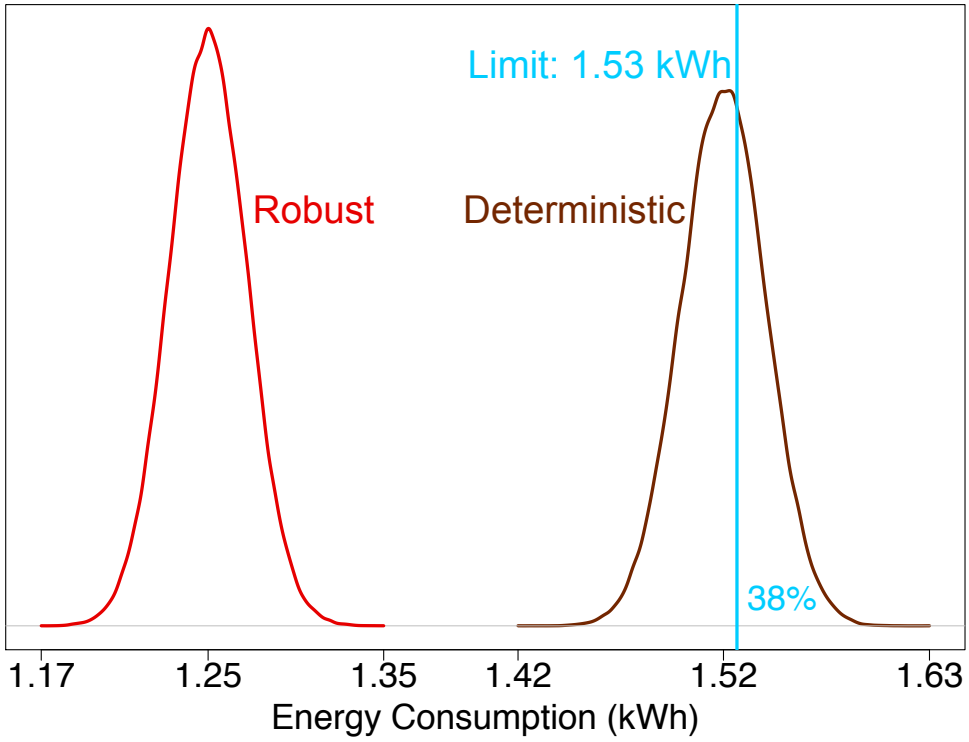

Figure 5.4: The Power of Robustness (1): The distribution of simulated trip energy consumption from Belmont, MA to MIT shows that incorporating robustness ensures that the vehicle can reach its destination when the energy limit is 1.53 kWh. In the deterministic case, the vehicle does not have enough energy to reach the destination 38% of the time.

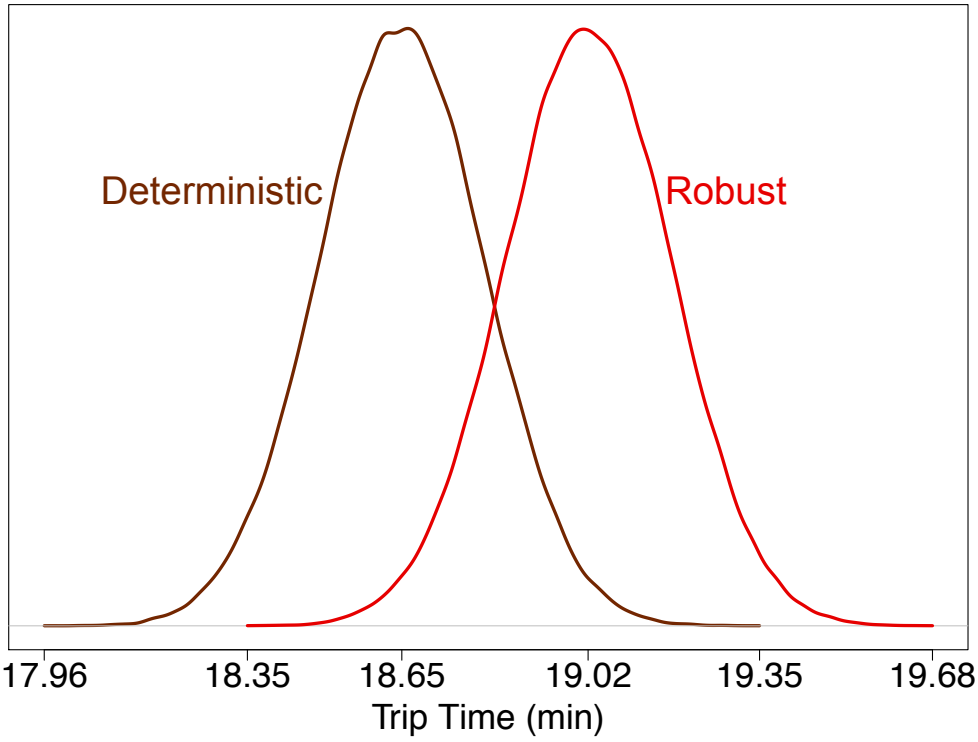

Figure 5.5: The Power of Robustness (2): According to the simulated travel time distributions for the trip from Belmont, MA to MIT, the time sacrificed is small (the difference of the means of the simulated travel times is about 0.37 minutes) compared with the increased confidence of reaching the destination using at most the allowed amount of energy.

We can also conduct similar computations for the deterministic case and for ellipsoidal uncertainty, which produce similar results.

The computations for ellipsoidal uncertainty are in Figures 5.6, 5.7, and 5.8. If we conduct simulations with underlying uncertainty, the robust formulation under ellipsoidal uncertainty also effectively protects against uncertainty, for a reasonable cost in time (the difference of the means of the simulated travel times is about 0.36 minutes), as shown in Figures 5.9 and 5.10. Figures 5.9 and 5.10 are very similar to Figures 5.4 and 5.5 because the algorithms for polyhedral and ellipsoidal uncertainty find the same route for the energy limit of 1.53 kWh.

The computations for the deterministic case are shown in Figures 5.11, 5.12, and 5.13.

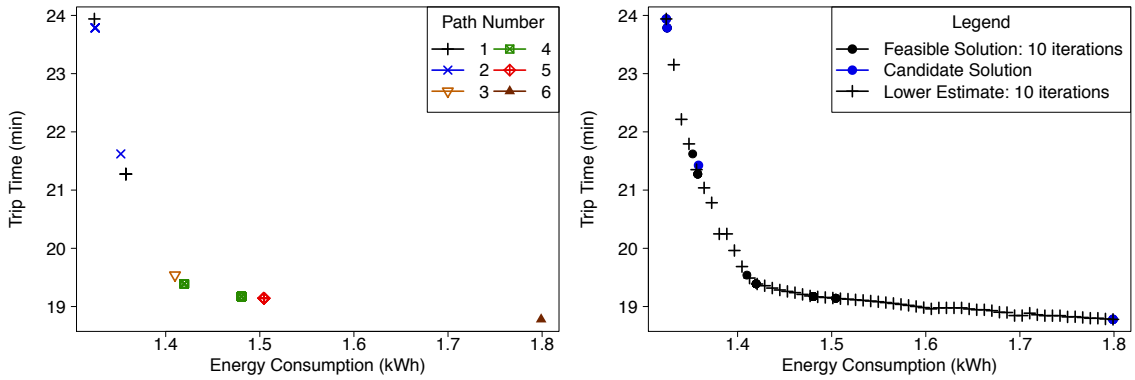

Figure 5.6: Left: The Pareto frontier, excluding candidate solutions, for the trip from Belmont, MA to MIT, using robust optimization under ellipsoidal uncertainty and allowing for 10 iterations of subgradient ascent. Right: The estimates of the robust Pareto frontiers suggest that our algorithm provides near-optimal solutions.

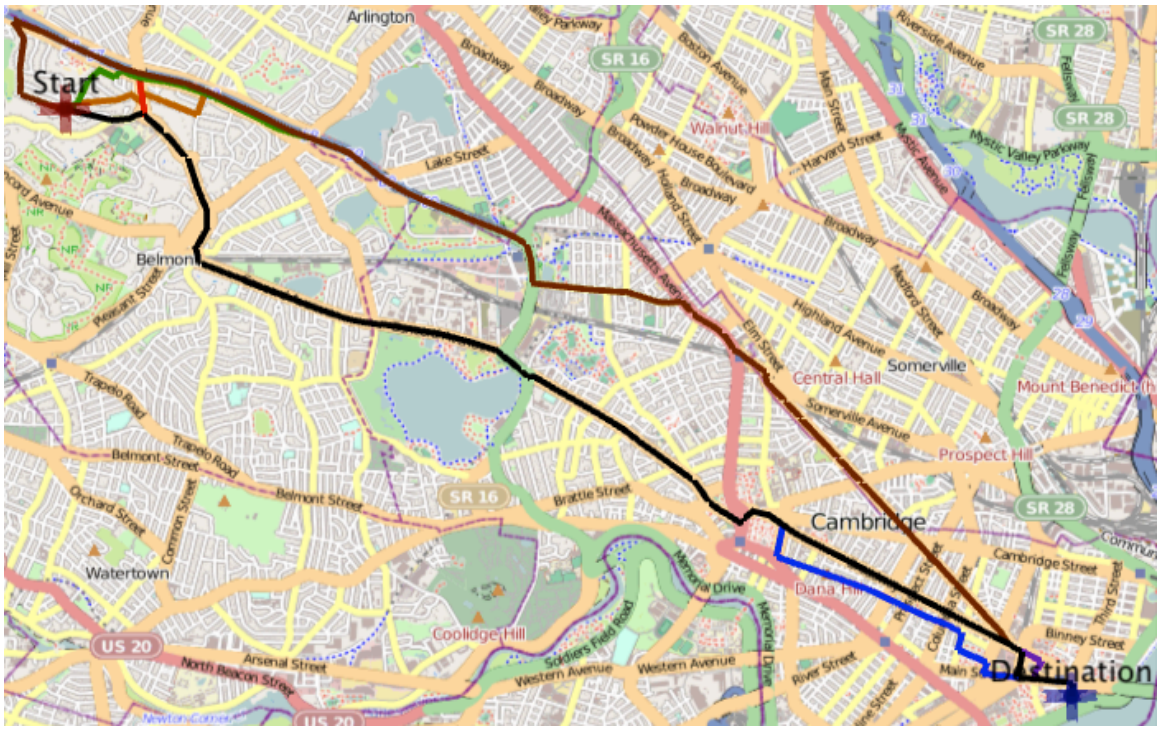

Figure 5.7: The paths produced for the trip from Belmont, MA to MIT using the robust optimization framework for ellipsoidal uncertainty, illustrated over OpenStreetMap data ([34]) using Unfolding for Eclipse.

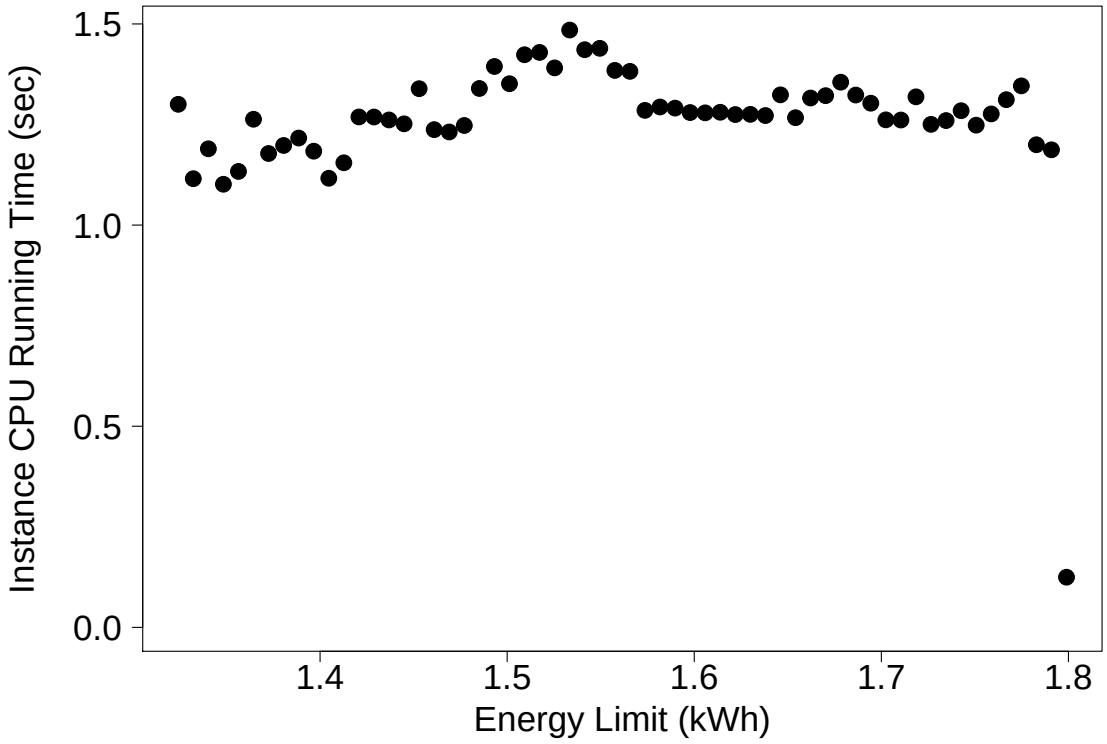

Figure 5.8: The CPU running times, in seconds, for the trip from Belmont, MA to MIT for ellipsoidal uncertainty.

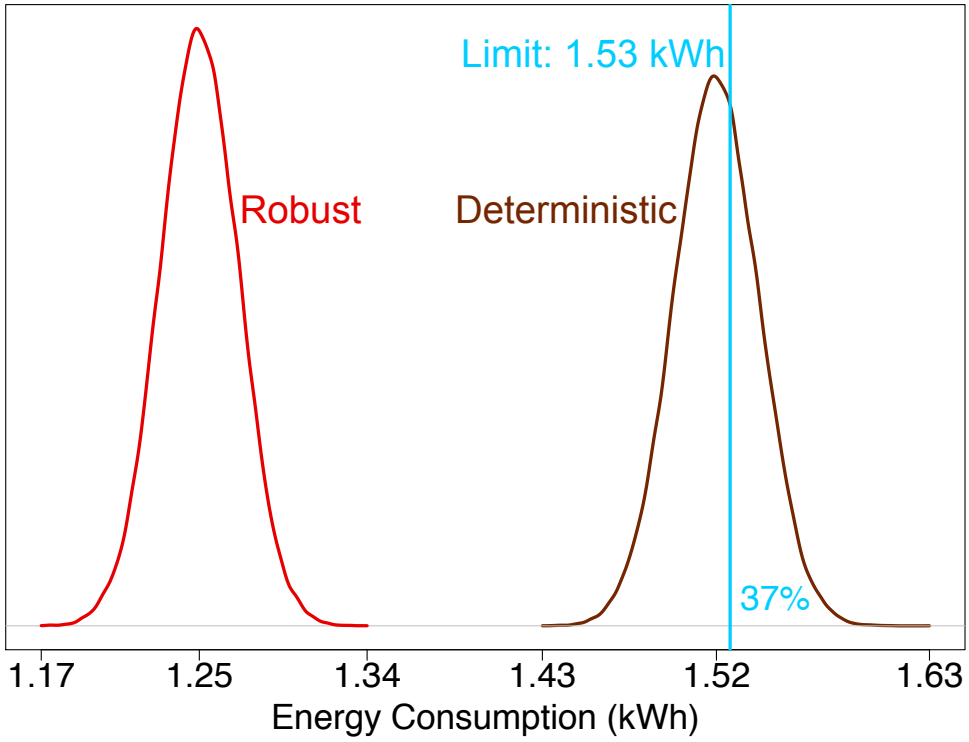

Figure 5.9: The Power of Robustness (1): The distribution of simulated trip energy consumption from Belmont, MA to MIT shows that incorporating robustness (under ellipsoidal uncertainty) ensures that the vehicle can reach its destination when the energy limit is 1.53 kWh. In the deterministic case, the vehicle does not have enough energy to reach the destination 37% of the time.

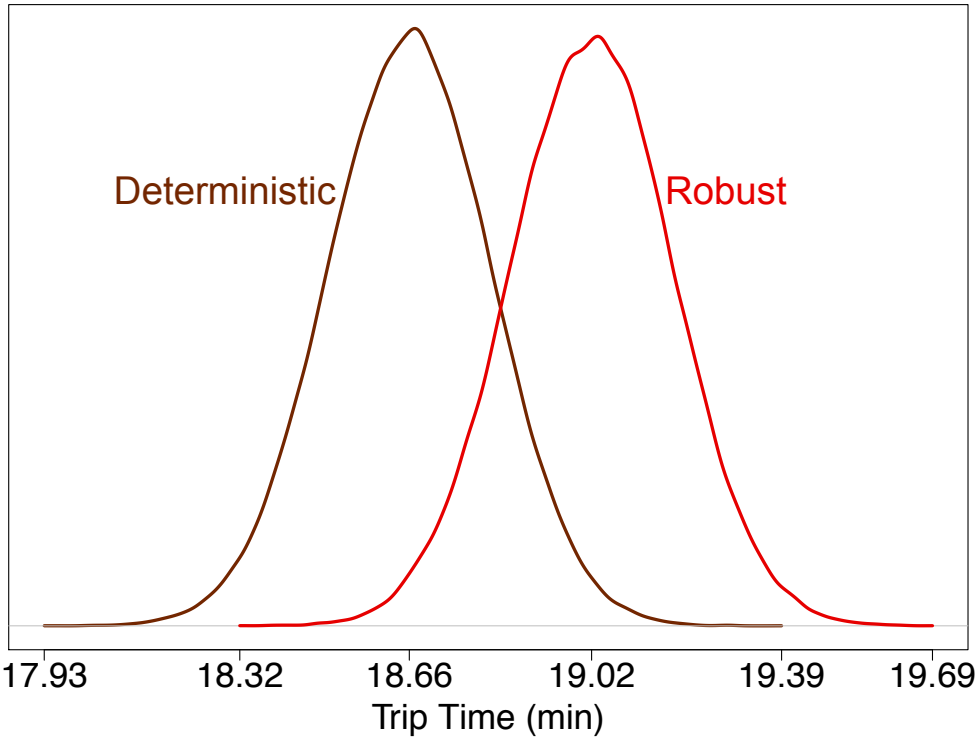

Figure 5.10: The Power of Robustness (2): According to the simulated travel time distributions for the trip from Belmont, MA to MIT, the time sacrificed is small (the difference of the means of the simulated travel times is about 0.36 minutes) compared with the increased confidence of reaching the destination using at most the allowed amount of energy.

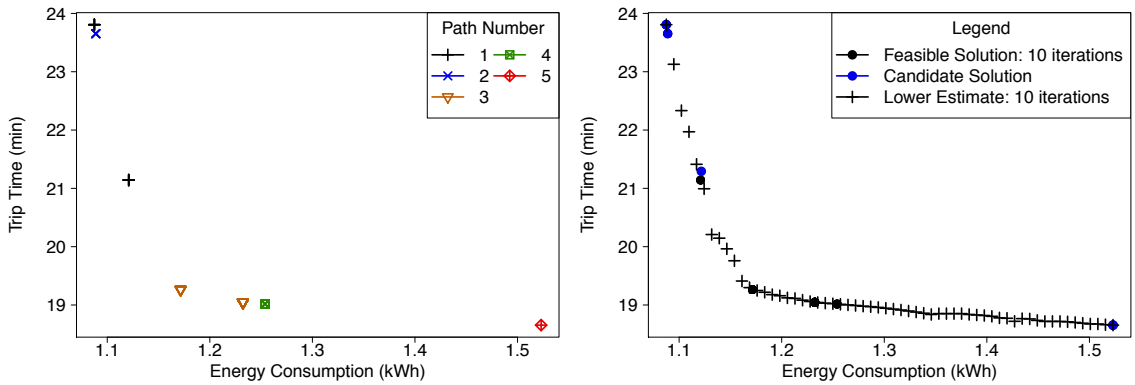

Figure 5.11: Left: The Pareto frontier, excluding candidate solutions, for the trip from Belmont, MA to MIT, for the deterministic case, and allowing for 10 iterations of subgradient ascent. Right: The estimates of the robust Pareto frontiers suggest that our algorithm provides near-optimal solutions.

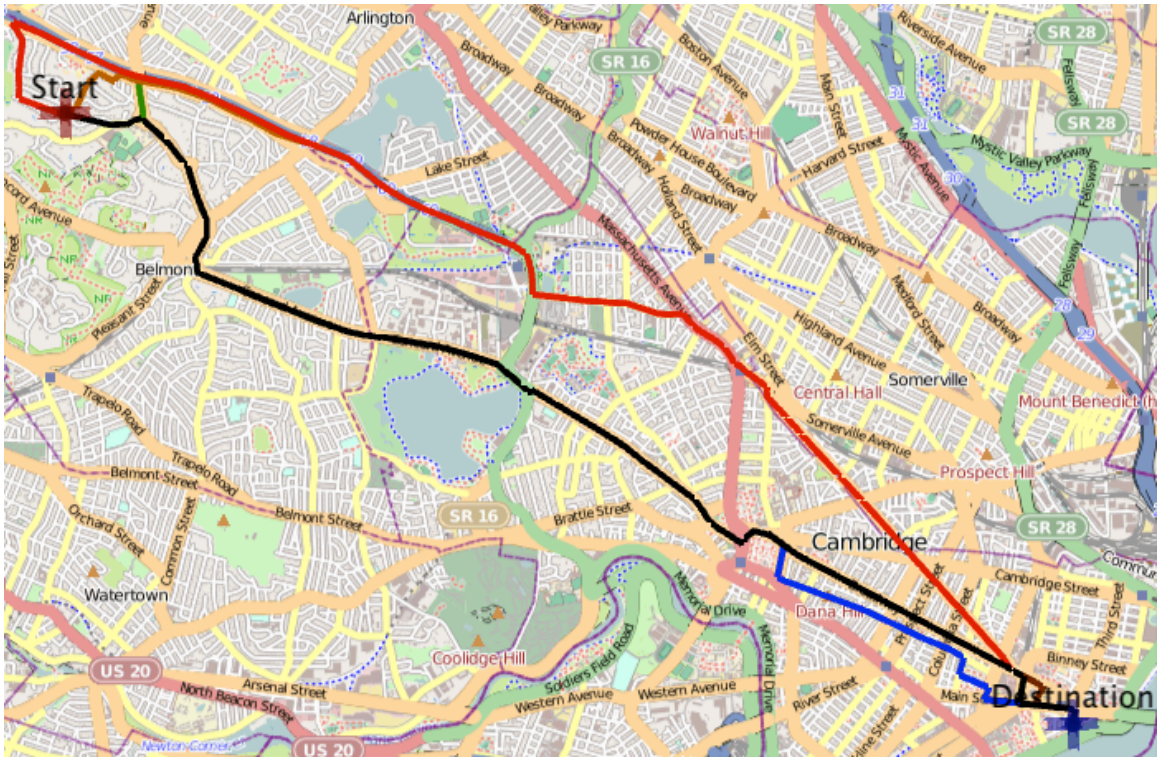

Figure 5.12: The paths produced for the trip from Belmont, MA to MIT using the deterministic framework, illustrated over OpenStreetMap data ([34]) using Unfolding for Eclipse.

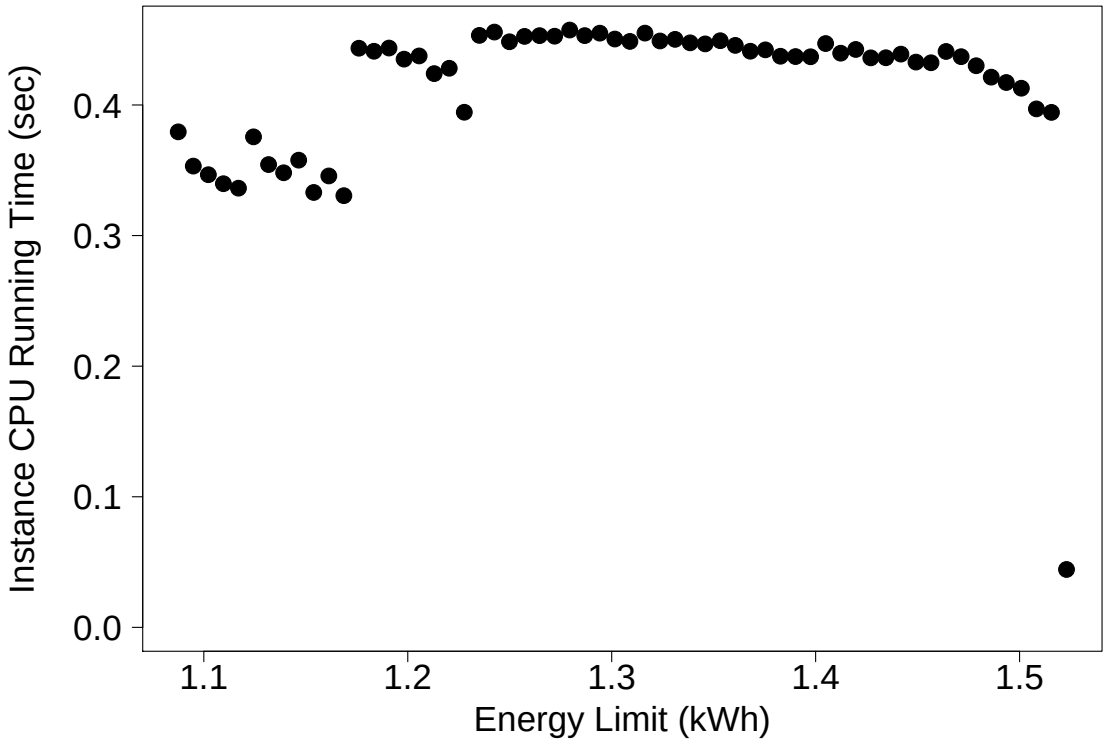

Figure 5.13: The CPU running times, in seconds, for the trip from Belmont, MA to MIT for the deterministic case.

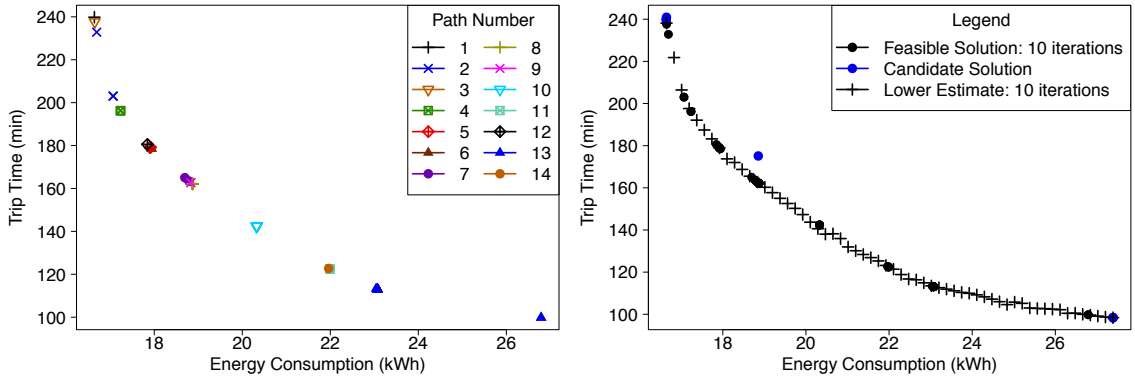

Figure 5.14: Left: The Pareto frontier, excluding candidate solutions, for the trip in Michigan, using robust optimization under polyhedral uncertainty and allowing for 10 iterations of subgradient ascent. Right: The estimates of the robust Pareto frontiers suggest that our algorithm provides near-optimal solutions.

5.2.2 A Longer Trip in Michigan, with a Larger Network

We apply the algorithms to an example trip in Michigan from Lansing, MI to Dearborn, MI for a larger network, which has 409,054 nodes and 975,666 arcs. In general, we obtain similar results as for the example trip from Belmont, MA to MIT, although running times are longer due to the greater distance traveled and larger network. Figures 5.14, 5.15, and 5.16 illustrate the results for polyhedral uncertainty.

We can also conduct similar computations for the deterministic case and for ellipsoidal uncertainty, which produce similar results. The computations for ellipsoidal uncertainty are in Figures 5.17, 5.18, and 5.19, and the computations for the deterministic case are in Figures 5.20, 5.21, and 5.22.

Figure 5.15: The paths produced for the trip in Michigan using the robust optimization framework for polyhedral uncertainty, illustrated over OpenStreetMap data ([34]) using Unfolding for Eclipse.

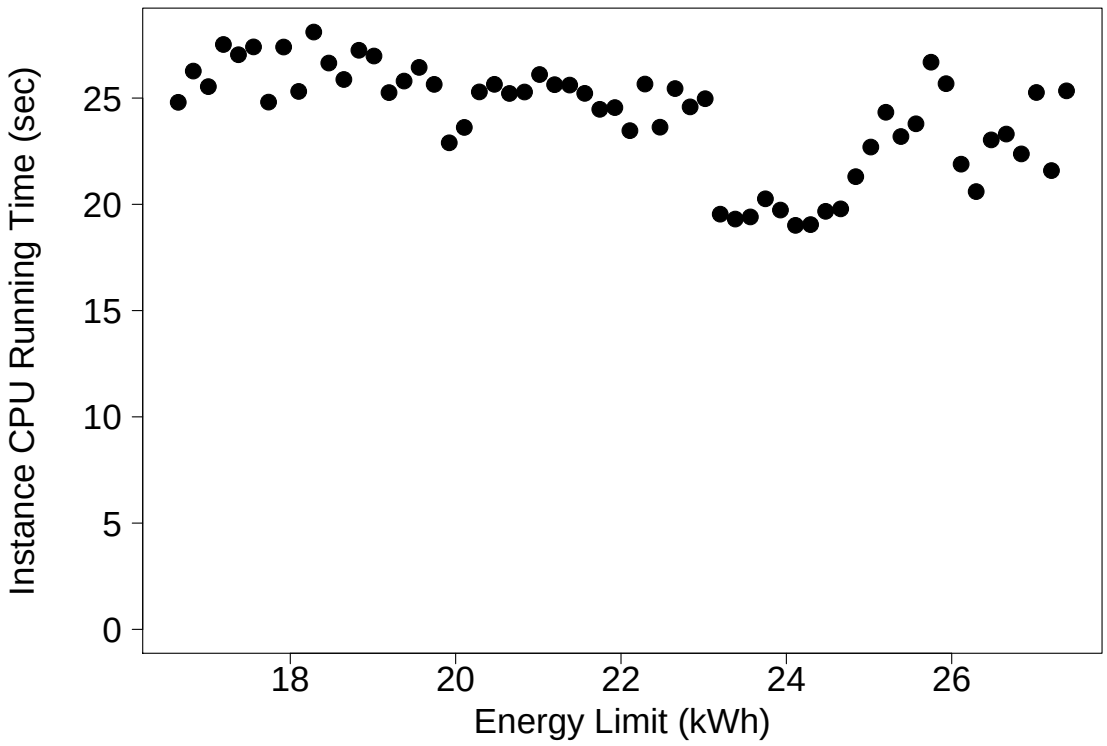

Figure 5.16: The CPU running times, in seconds, for the trip in Michigan, for polyhedral uncertainty.

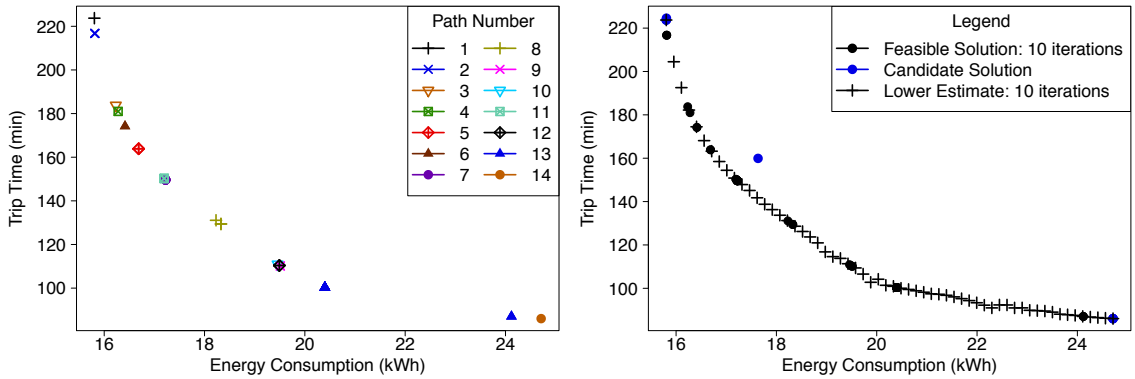

Figure 5.17: Left: The Pareto frontier, excluding candidate solutions, for the trip in Michigan, using robust optimization under ellipsoidal uncertainty and allowing for 10 iterations of subgradient ascent. Right: The estimates of the robust Pareto frontiers suggest that our algorithm provides near-optimal solutions.

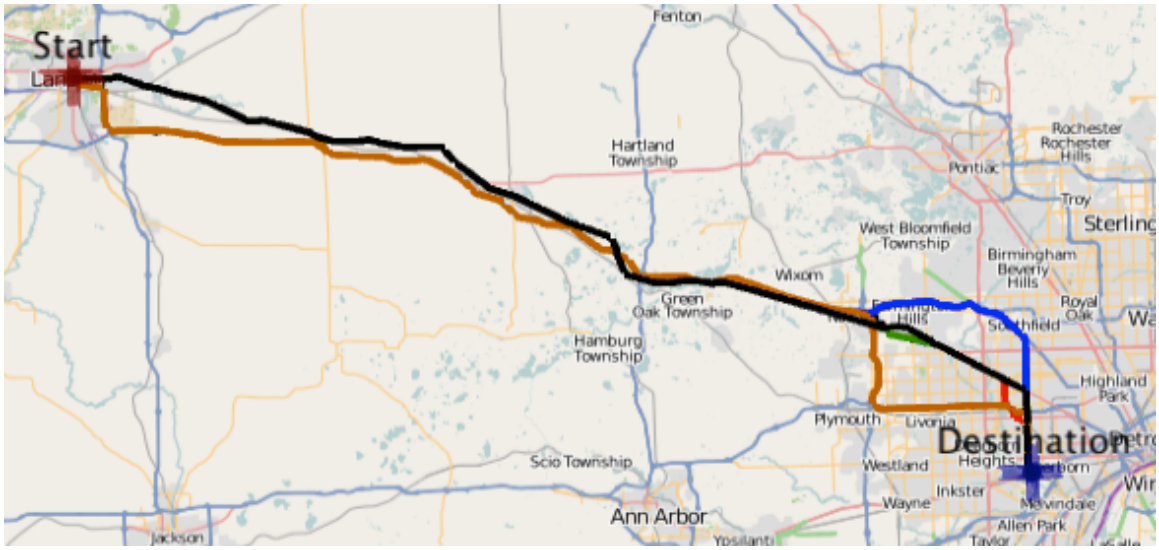

Figure 5.18: The paths produced for the trip in Michigan using the robust optimization framework for ellipsoidal uncertainty, illustrated over OpenStreetMap data ([34]) using Unfolding for Eclipse.

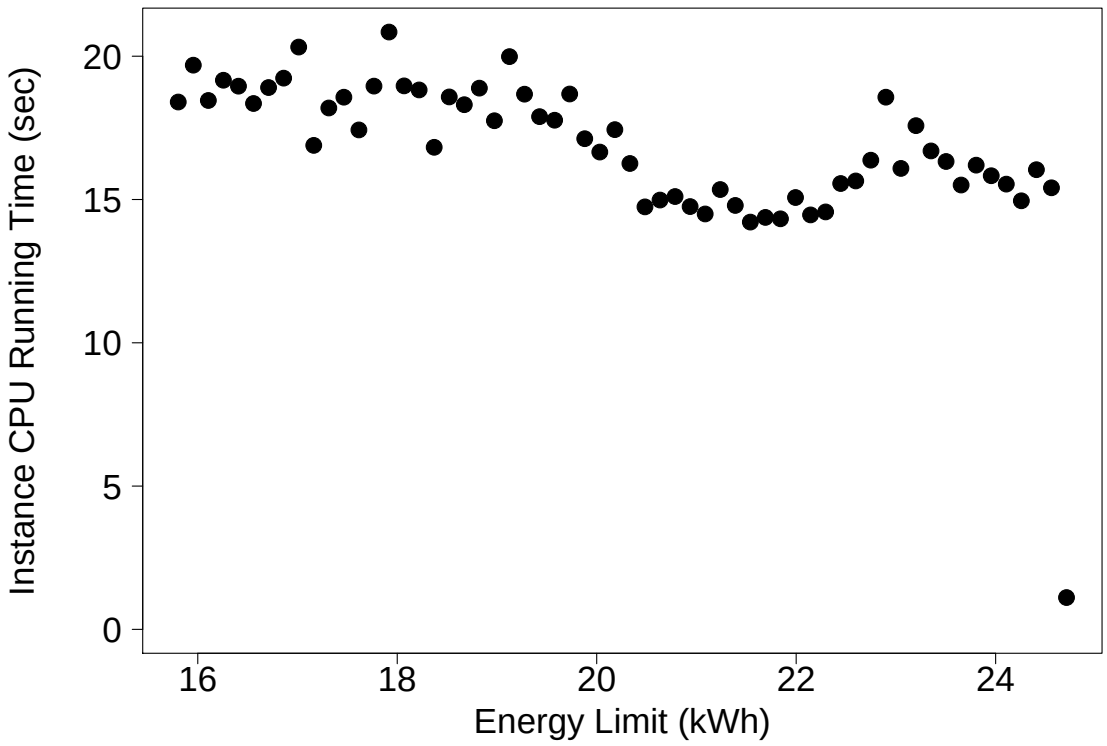

Figure 5.19: The CPU running times, in seconds, for the trip in Michigan for ellipsoidal uncertainty.

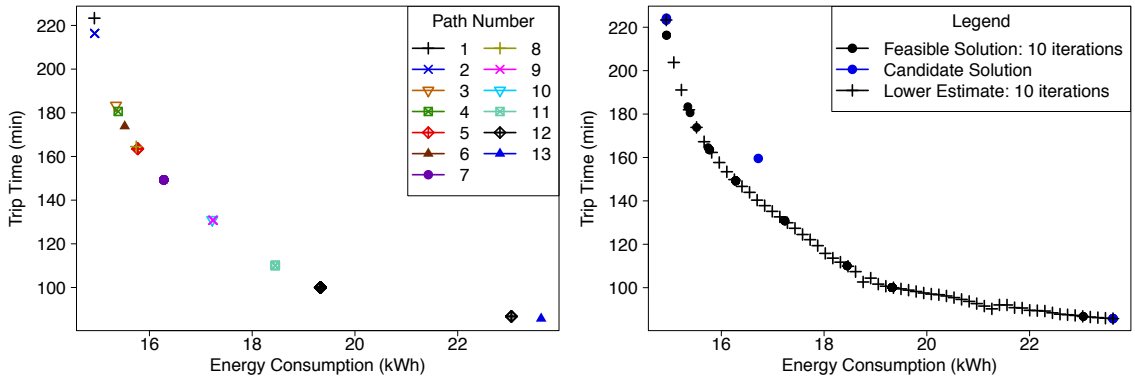

Figure 5.20: Left: The Pareto frontier, excluding candidate solutions, for the trip in Michigan, for the deterministic case, and allowing for 10 iterations of subgradient ascent. Right: The estimates of the robust Pareto frontiers suggest that our algorithm provides near-optimal solutions.

Figure 5.21: The paths produced for the trip in Michigan using the deterministic framework, illustrated over OpenStreetMap data ([34]) using Unfolding for Eclipse.

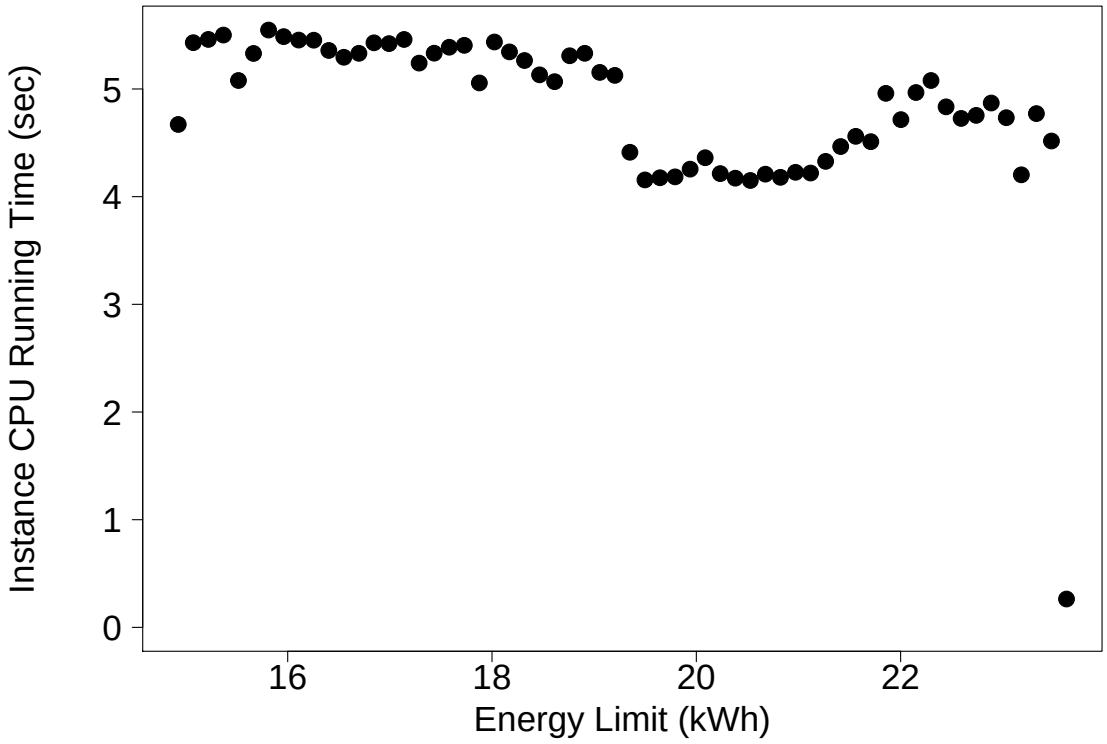

Figure 5.22: The CPU running times, in seconds, for the trip in Michigan for the deterministic case.

Chapter 6

Extensions, Future Work, and Conclusion

In this chapter, we propose extensions, describe additional future work, and then conclude the thesis. We first describe a way to incorporate traffic signals and traffic conditions into the optimization framework, and apply our algorithms to an example with traffic lights and some traffic data in Cambridge, MA (Section 6.1). Then, we propose an optimization-based formulation that explicitly incorporates acceleration (Section 6.2). After that, we describe additional future work (Section 6.3). And finally, we conclude this thesis (Section 6.4).

6.1 Extension: Traffic Signals and Traffic Conditions

We obtain data with traffic signals in Cambridge, MA ([11]), which is shown in Figure 6.1, and additional network data from the Massachusetts Department of Transportation (MassDOT) ([31]) to incorporate into our analysis. In particular,

we use the MassDOT data's average annual daily traffic count for each road or highway as an estimate for real-time traffic information that might be available for each arc in the graph. We break the traffic counts into 4 categories, as illustrated in Figure 6.2, and adjust the lower and upper velocity bounds for the arcs based on these counts:

- Minimal (0–5,765 ADT): 0% change in speed
- Low (5,766–25,786 ADT): 20% reduction in speed
- Medium (25,787–83,659 ADT): 40% reduction in speed
- High (83,660 and higher ADT): 60% reduction in speed.

In addition, we assume that each stoplight requires the driver to wait a random time that is uniformly distributed over 0 to 30 seconds. This translates to adding the mean of 15 seconds to τ_{ij} to the arcs with the stoplights, and adding another 15 seconds to σ_{ij} (the appropriate parameters for energy consumption are modified accordingly, based on this change).

The network has 9334 nodes and 17,575 arcs. The traffic signal data does not include stop signs. Incorporating this additional data changes the route taken, as shown in Figure 6.3. In particular, the top route in Figure 6.3 is on a street with traffic signals, and when these signals are incorporated into the data, the algorithm picks the bottom route in Figure 6.3 that avoids some of the traffic signals (stop signs were not included in the data). In both cases, only one route from MIT to Harvard Square is on the Pareto frontier. For example, when there are no traffic signals and minimal traffic conditions, the robust-minimum-time

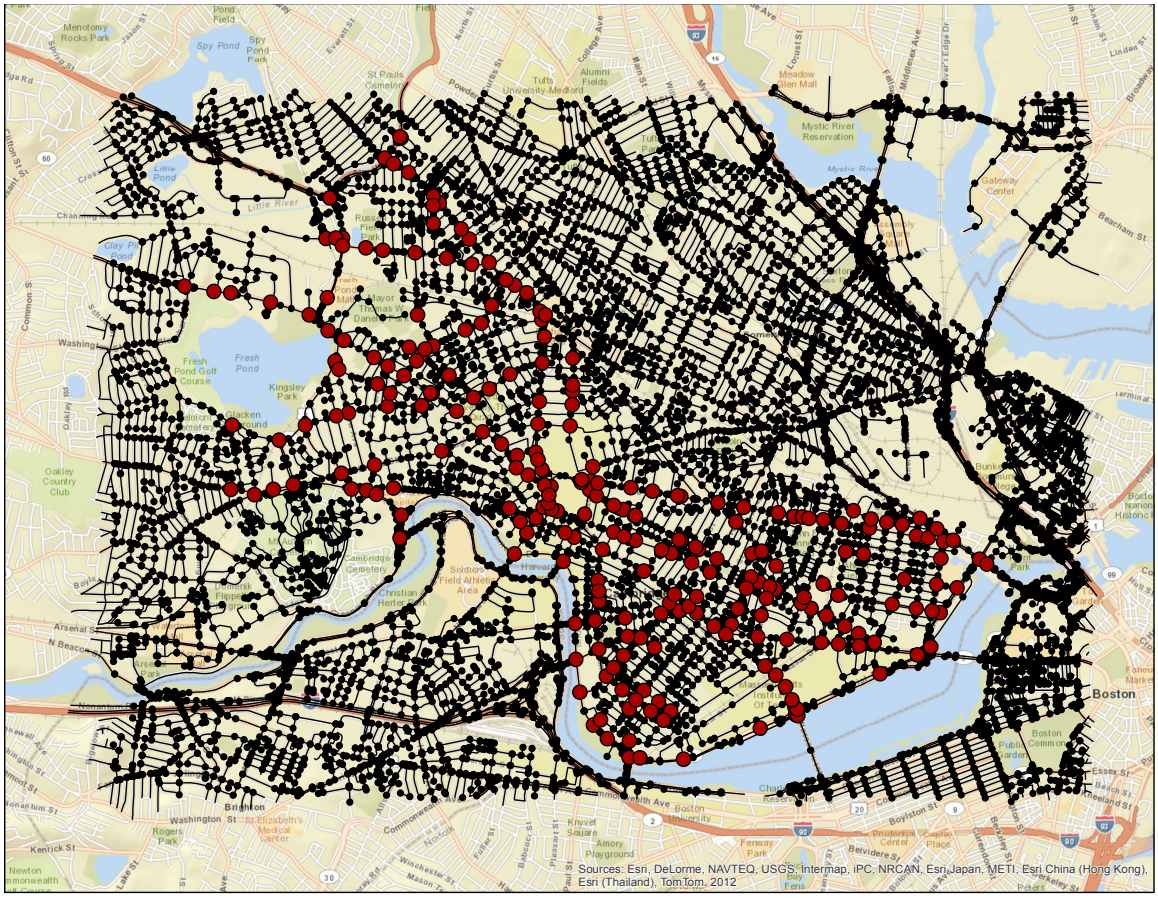

Figure 6.1: A smaller road network, combined with the locations of 250 traffic signals in Cambridge, MA from the Cambridge Department of Transportation ([11])

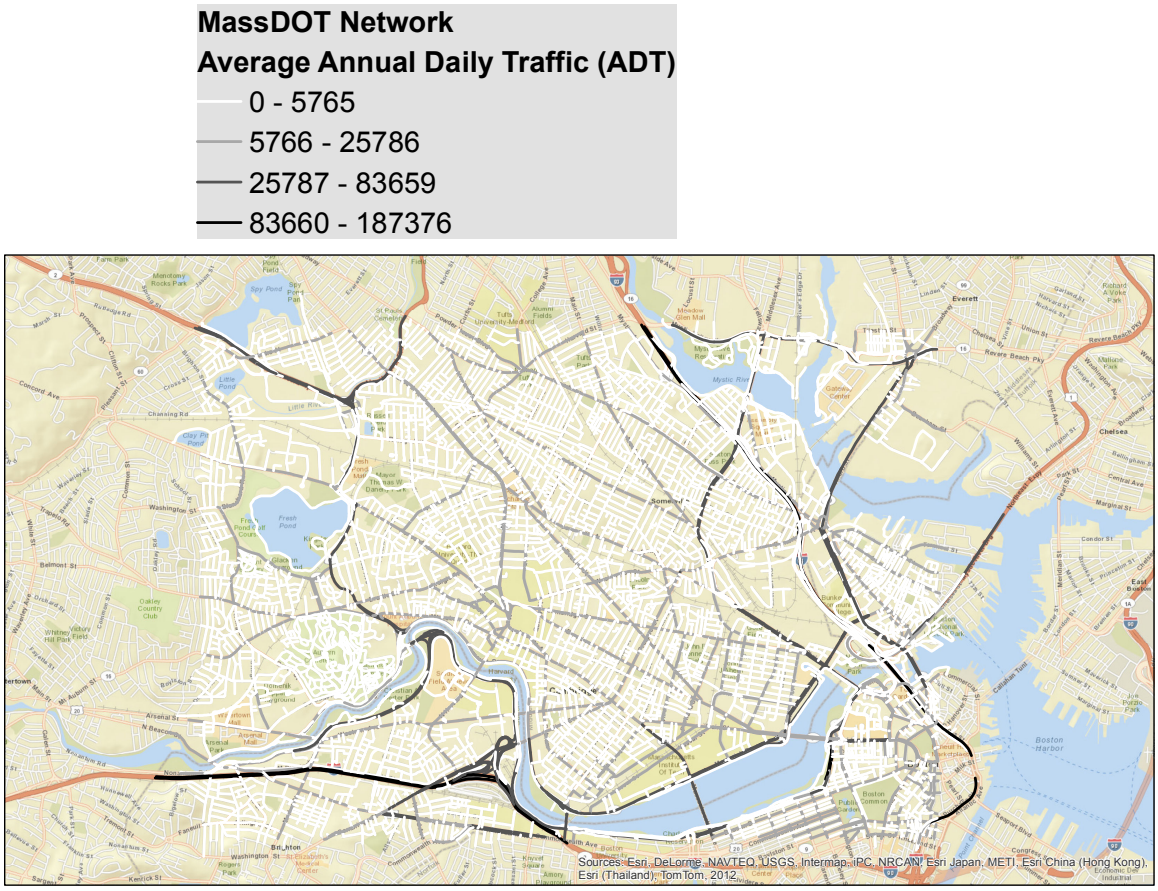

Figure 6.2: Average Annual Daily Traffic Count data from the Massachusetts Department of Transportation ([31])

path anticipates 6 minutes and 25 seconds and 0.4662 kWh under uncertainty, and under the deterministic setting, it would require 5 minutes and 50 seconds and 0.4199 kWh. However, with traffic signals and traffic conditions, the robust-minimum-time path anticipates 10 minutes and 4 seconds and 0.5706 kWh under uncertainty, and under the deterministic setting, it would require 8 minutes and 21 seconds and 0.4867 kWh. This is a substantial increase in time, and is due to the incorporation of traffic congestion and the additional time spent waiting at stoplights.

Figure 6.3: The path produced for a trip in Cambridge, MA changes due to traffic and traffic signals. In particular, the bottom path avoids some of the traffic signals. In particular, the path on the top is produced when traffic signals and traffic conditions are not considered, while the path on the bottom takes these things into account. Both paths were found using the robust optimization framework for polyhedral uncertainty and are illustrated over OpenStreetMap data ([34]) using Unfolding for Eclipse.

6.2 Extension: Incorporating Acceleration

6.2.1 Energy and Time Formulas

Accelerating or Decelerating Only

In terms of forces and integrals, the energy model (2.1) in Section 2.1 becomes:

$$F(t) = \frac{1}{2}\rho C_w A_f v^2(t) + \mu mg\cos(\alpha(t)) + mg\sin(\alpha(t)) + m\delta\frac{dv}{dt}$$
$$E = \int_0^{T_0} \frac{1}{\eta} F v(t) dt + P_{\text{acc}} T_0,$$

where T_0 is the total time.

Suppose that the vehicle accelerates from velocity v_0 to v_f at a rate $a > 0$ at a constant road grade α. Then results from using basic mechanics imply:

- The time it takes to accelerate is $t_{vf} = \dfrac{v_f - v_0}{a}$. Integrating with respect to time:

$$\frac{dv(t)}{dt} = a$$
$$\Rightarrow v(t) = a * t + v_0$$
$$\Rightarrow t_{vf} = \frac{v_f - v_0}{a}$$

- The vehicle will travel a distance of $\dfrac{v_f^2 - v_0^2}{2a}$ while accelerating. The time required to accelerate to velocity v_f is $t_{vf} = \dfrac{v_f - v_0}{a}$. During this time, a

distance of x_{vf} is traveled, which is

$$\begin{aligned} x_{vf} &= \frac{1}{2}at_{vf}^2 + v_0 t_{vf} \\ \Rightarrow x_{vf} &= \frac{\left(v_f - v_0\right)^2}{2a} + v_0\left(\frac{v_f - v_0}{a}\right) \\ \Rightarrow x_{vf} &= \frac{v_f^2 - v_0^2}{2a} \end{aligned}$$

- As we will show next, the vehicle will consume a total energy of

$$E = \left\{ \begin{array}{l} \frac{1}{\eta a}\left(\frac{1}{8}\rho C_w A_f v_f^4 + \frac{1}{2}\left(\mu mg\cos\alpha + mg\sin\alpha\right)v_f^2\right) + \frac{1}{2\eta}m\delta v_f^2 \\ -\frac{1}{\eta a}\left(\frac{1}{8}\rho C_w A_f v_0^4 + \frac{1}{2}\left(\mu mg\cos\alpha + mg\sin\alpha\right)v_0^2\right) - \frac{1}{2\eta}m\delta v_0^2 \\ + P_{\text{acc}}\left(\frac{v_f - v_0}{a}\right) \end{array} \right\}.$$

Note. If the vehicle is decelerating, namely, going from velocity v_0 to $v_f < v_0$ at a rate of $a < 0$, then all three prior formulas are the same (the integration for computing the energy is a little different, but the result is the same).

To obtain the energy, we integrate with respect to time from $t = 0$ to $T =$

$\frac{v_f - v_0}{a}$.

$$E = \left\{ \begin{array}{l} \int_0^{\frac{v_f - v_0}{a}} \frac{1}{\eta} \left(\begin{array}{l} \frac{1}{2}\rho C_w A_f \left(v_0 + at\right)^3 \\ + \left(\mu mg\cos\alpha + mg\sin\alpha + m\delta a\right)\left(v_0 + at\right) \end{array} \right) dt \\ + P_{\text{acc}} \left(\frac{v_f - v_0}{a} \right) \end{array} \right\}$$

$$E = \left\{ \begin{array}{l} \frac{1}{\eta a} \int_{v_0}^{v_f} \left(\frac{1}{2}\rho C_w A_f u^3 + \left(\mu mg\cos\alpha + mg\sin\alpha + m\delta a\right) u \right) du \\ + P_{\text{acc}} \left(\frac{v_f - v_0}{a} \right) \end{array} \right\}$$

$$[u = v_0 + at, du = adt]$$

$$E = \left(\begin{array}{l} \frac{1}{\eta a} \left(\frac{1}{8}\rho C_w A_f u^4 + \frac{1}{2}\left(\mu mg\cos\alpha + mg\sin\alpha + m\delta a\right) u^2 \right) \Big|_{v_0}^{v_f} \\ + P_{\text{acc}} \left(\frac{v_f - v_0}{a} \right) \end{array} \right)$$

$$E = \left\{ \begin{array}{l} \frac{1}{\eta a} \left(\frac{1}{8}\rho C_w A_f v_f^4 + \frac{1}{2}\left(\mu mg\cos\alpha + mg\sin\alpha + m\delta a\right) v_f^2 \right) \\ - \frac{1}{\eta a} \left(\frac{1}{8}\rho C_w A_f v_0^4 + \frac{1}{2}\left(\mu mg\cos\alpha + mg\sin\alpha + m\delta a\right) v_0^2 \right) \\ + P_{\text{acc}} \left(\frac{v_f - v_0}{a} \right) \end{array} \right\}$$

$$E = \left\{ \begin{array}{l} \frac{1}{\eta a} \left(\frac{1}{8}\rho C_w A_f v_f^4 + \frac{1}{2}\left(\mu mg\cos\alpha + mg\sin\alpha\right) v_f^2 \right) + \frac{1}{2\eta} m\delta v_f^2 \\ - \frac{1}{\eta a} \left(\frac{1}{8}\rho C_w A_f v_0^4 + \frac{1}{2}\left(\mu mg\cos\alpha + mg\sin\alpha\right) v_0^2 \right) - \frac{1}{2\eta} m\delta v_0^2 \\ + P_{\text{acc}} \left(\frac{v_f - v_0}{a} \right) \end{array} \right\}$$

Formulas when Traveling at a Constant Velocity after Accelerating

If the vehicle is traveling at a constant velocity v_c for a time T_0, at a constant road grade α, then we have the following results:

- The vehicle will travel a distance of $v_c T_0$ (distance equals speed times time).
- The vehicle will consume a total energy of

$$E = \frac{1}{\eta}\left(\frac{1}{2}\rho C_w A_f v_c^3 T_0 + \mu m g \cos(\alpha) v_c T_0 + m g \sin(\alpha) v_c T_0\right) + P_{\text{acc}} T_0.$$

  To obtain this result, apply the energy model (2.1) in Section 2.1, using $\Delta t = T_0$ and $\Delta v = 0$.

Using these results and the ones in Section 6.2.1, suppose that the vehicle wishes to travel a distance of ℓ, starts at velocity v_0, accelerates at a rate $a > 0$ to velocity v_f, and then travels the rest of the distance at velocity v_f. Then:

- The total travel time is

$$\begin{aligned}\frac{v_f - v_0}{a} + \frac{\ell - \dfrac{v_f^2 - v_0^2}{2a}}{v_f} &= \frac{v_f - v_0}{a} + \frac{2a\ell + v_0^2 - v_f^2}{2a v_f} \\ &= \frac{\ell}{v_f} + \frac{1}{a}\left(\frac{v_f}{2} - v_0 + \frac{v_0^2}{2v_f}\right) \\ &= \frac{\ell}{v_f} + \frac{\left(v_f - v_0\right)^2}{2a v_f}.\end{aligned}$$

- The total energy consumed is

$$E = \left\{ \begin{array}{l} -\dfrac{1}{8a\eta}\rho C_w A_f \left(v_f^2 - v_0^2\right)^2 + \dfrac{1}{2\eta}\rho C_w A_f v_f^2 \ell \\ + \dfrac{\ell}{\eta}\left(\mu m g \cos(\alpha) + m g \sin(\alpha)\right) + \dfrac{1}{2\eta} m \delta \left(v_f^2 - v_0^2\right) \\ + P_{\text{acc}}\left(\dfrac{\ell}{v_f} + \dfrac{\left(v_f - v_0\right)^2}{2a v_f}\right) \end{array} \right\}.$$

If the vehicle is decelerating, namely, going from velocity v_0 to $v_f < v_0$ at a rate of $a < 0$, then the same formulas hold.

The algebra for the energy consumption is below:

$$E = \left\{ \begin{array}{l} \left\{ \begin{array}{l} \frac{1}{\eta a}\left(\frac{1}{8}\rho C_w A_f v_f^4 + \frac{1}{2}\left(\mu m g \cos\alpha + m g \sin\alpha\right) v_f^2\right) + \frac{1}{2\eta} m \delta v_f^2 \\ -\frac{1}{\eta a}\left(\frac{1}{8}\rho C_w A_f v_0^4 + \frac{1}{2}\left(\mu m g \cos\alpha + m g \sin\alpha\right) v_0^2\right) - \frac{1}{2\eta} m \delta v_0^2 \\ + P_{\text{acc}}\left(\frac{v_f - v_0}{a}\right) \end{array} \right\} \\ + \frac{1}{2\eta}\rho C_w A_f v_f^3 \left(\frac{2a\ell + v_0^2 - v_f^2}{2a v_f}\right) \\ + \frac{1}{\eta}\left(\mu m g \cos\left(\alpha\right) + m g \sin\left(\alpha\right)\right) v_f \left(\frac{2a\ell + v_0^2 - v_f^2}{2a v_f}\right) \\ + P_{\text{acc}}\left(\frac{2a\ell + v_0^2 - v_f^2}{2a v_f}\right) \end{array} \right\}.$$

This is the same as

$$E = \left\{ \begin{array}{l} \frac{1}{\eta a}\left(\frac{1}{8}\rho C_w A_f v_f^4\right) + \frac{1}{2\eta} m\delta v_f^2 \\ + \frac{1}{2\eta}\rho C_w A_f v_f^2 \left(\frac{2a\ell + v_0^2 - v_f^2}{2a}\right) \\ - \frac{1}{\eta a}\left(\frac{1}{8}\rho C_w A_f v_0^4\right) - \frac{1}{2\eta} m\delta v_0^2 \\ + \frac{1}{2a\eta}\left(\mu mg\cos\alpha + mg\sin\alpha\right) v_f^2 \\ - \frac{1}{2a\eta}\left(\mu mg\cos\alpha + mg\sin\alpha\right) v_0^2 \\ + \frac{1}{2a\eta}\left(\mu mg\cos\left(\alpha\right) + mg\sin\left(\alpha\right)\right)\left(2a\ell + v_0^2 - v_f^2\right) \\ + P_{\text{acc}}\left(\frac{\ell}{v_f} + \frac{\left(v_f - v_0\right)^2}{2av_f}\right) \end{array} \right\}.$$

Which is:

$$E = \left\{ \begin{array}{l} -\frac{1}{8a\eta}\rho C_w A_f \left(v_f^4 - 2v_0^2 v_f^2 + v_0^4\right) + \frac{1}{2\eta}\rho C_w A_f v_f^2 \ell \\ + \frac{\ell}{\eta}\left(\mu mg\cos\left(\alpha\right) + mg\sin\left(\alpha\right)\right) + \frac{1}{2\eta} m\delta\left(v_f^2 - v_0^2\right) \\ + P_{\text{acc}}\left(\frac{\ell}{v_f} + \frac{\left(v_f - v_0\right)^2}{2av_f}\right) \end{array} \right\}.$$

This is the same as

$$E = \left\{ \begin{array}{l} -\frac{1}{8a\eta}\rho C_w A_f \left(v_f^2 - v_0^2\right)^2 + \frac{1}{2\eta}\rho C_w A_f v_f^2 \ell \\ + \frac{\ell}{\eta}\left(\mu mg\cos\left(\alpha\right) + mg\sin\left(\alpha\right)\right) + \frac{1}{2\eta} m\delta\left(v_f^2 - v_0^2\right) \\ + P_{\text{acc}}\left(\frac{\ell}{v_f} + \frac{\left(v_f - v_0\right)^2}{2av_f}\right) \end{array} \right\}.$$

If $v_0 = v_f$ in the prior time and energy formulas, then a can be any value and we will obtain the time and energy consumption for the case of no acceleration.

6.2.2 Optimization-Based Formulations

It is possible to directly incorporate acceleration into a mathematical model. However, incorporating acceleration means that the travel time over a segment depends on the final velocity of the previous segment, which makes the optimization problem more difficult. Here we describe some possible optimization-based formulations that include acceleration.

In this section, we assume that the vehicle can accelerate or decelerate to v_f before traveling a distance of ℓ. Mathematically, we are assuming that

$$\ell \geq \frac{v_f^2 - v_0^2}{2a}.$$

This is with no loss of generality since we can add this as a constraint in the mathematical program and we can select v_f to be as close to or as far away from v_0 as we would like.

One Arc Formulation

We need to solve three problems and select the solution out of the three with the lowest objective value:

- Accelerate from v_0 to $v_f > v_0$ at rate $a > 0$.
- Decelerate from v_0 to $v_f < v_0$ at rate of acceleration $a < 0$.

- Choose a constant speed $v_0 = v_f$.

To illustrate the one arc formulation, consider the case when we are accelerating at rate $a > 0$.

$$\begin{aligned}
\min_{v_0, v_f, a} \quad & \frac{\ell}{v_f} + \frac{\left(v_f - v_0\right)^2}{2a v_f} \\
\text{s.t.} \quad & \left(\begin{array}{l} -\frac{1}{8a\eta}\rho C_w A_f \left(v_f^2 - v_0^2\right)^2 + \frac{1}{2\eta}\rho C_w A_f v_f^2 \ell \\ + \frac{\ell}{\eta}\left(\mu m g \cos(\alpha) + m g \sin(\alpha)\right) + \frac{1}{2\eta} m \delta \left(v_f^2 - v_0^2\right) \\ + P_{\text{acc}} \left(\frac{\ell}{v_f} + \frac{\left(v_f - v_0\right)^2}{2a v_f} \right) \end{array} \right) \leq E_0 \\
& \underline{v} \leq v_0 \leq v_f \leq \overline{v} \\
& \frac{v_f^2 - v_0^2}{2a} \leq \ell \\
& 0 < a \leq R_{\text{accel}}.
\end{aligned}$$

We can set $u = 1/a$ to obtain the problem (6.1).

$$\begin{aligned}
\min_{v_0, v_f, u} \quad & \frac{\ell}{v_f} + u\frac{\left(v_f - v_0\right)^2}{2v_f} \\
\text{s.t.} \quad & \left(\begin{array}{l} -\frac{u}{8\eta}\rho C_w A_f \left(v_f^2 - v_0^2\right)^2 + \frac{1}{2\eta}\rho C_w A_f v_f^2 \ell \\ +\frac{\ell}{\eta}\left(\mu m g \cos(\alpha) + mg\sin(\alpha)\right) + \frac{1}{2\eta} m\delta\left(v_f^2 - v_0^2\right) \\ + P_{\text{acc}}\left(\frac{\ell}{v_f} + u\frac{\left(v_f - v_0\right)^2}{2v_f}\right) \end{array} \right) \leq E_0 \\
& \underline{v} \leq v_0 \leq v_f \leq \overline{v} \\
& u\left(\frac{v_f^2 - v_0^2}{2}\right) \leq \ell \\
& u \geq 1/R_{\text{accel}}.
\end{aligned} \tag{6.1}$$

The corresponding problem for deceleration at a rate $a < 0$ is:

$$\begin{aligned}
\min_{v_0, v_f, a} \quad & \frac{\ell}{v_f} + \frac{\left(v_f - v_0\right)^2}{2av_f} \\
\text{s.t.} \quad & \left(\begin{array}{l} -\frac{1}{8a\eta}\rho C_w A_f \left(v_f^2 - v_0^2\right)^2 + \frac{1}{2\eta}\rho C_w A_f v_f^2 \ell \\ +\frac{\ell}{\eta}\left(\mu m g \cos(\alpha) + mg\sin(\alpha)\right) + \frac{1}{2\eta} m\delta\left(v_f^2 - v_0^2\right) \\ + P_{\text{acc}}\left(\frac{\ell}{v_f} + \frac{\left(v_f - v_0\right)^2}{2av_f}\right) \end{array} \right) \leq E_0 \\
& \underline{v} \leq v_f \leq v_0 \leq \overline{v} \\
& \frac{v_f^2 - v_0^2}{2a} \leq \ell \\
& R_{\text{decel}} \leq a < 0.
\end{aligned}$$

By substituting $\hat{u} = -1/a$, we obtain the problem

$$\begin{aligned}
\min_{v_0, v_f, \hat{u}} \quad & \frac{\ell}{v_f} - \hat{u}\frac{\left(v_f - v_0\right)^2}{2v_f} \\
\text{s.t.} \quad & \left(\begin{array}{l} \dfrac{\hat{u}}{8\eta}\rho C_w A_f \left(v_f^2 - v_0^2\right)^2 + \dfrac{1}{2\eta}\rho C_w A_f v_f^2 \ell \\ + \dfrac{\ell}{\eta}\left(\mu m g \cos(\alpha) + m g \sin(\alpha)\right) + \dfrac{1}{2\eta} m\delta\left(v_f^2 - v_0^2\right) \\ + P_{\text{acc}}\left(\dfrac{\ell}{v_f} + u\dfrac{\left(v_f - v_0\right)^2}{2v_f}\right) \end{array} \right) \leq E_0 \\
& \underline{v} \leq v_f \leq v_0 \leq \overline{v} \\
& -\hat{u}\left(\frac{v_f^2 - v_0^2}{2}\right) \leq \ell \\
& \hat{u} \geq -1/R_{\text{decel}}.
\end{aligned} \tag{6.2}$$

When v_0 and v_f are fixed, the formulations (6.1) and (6.2) are both linear optimization problems (select the formulation that is appropriate for the choice of v_i).

The problem with no acceleration (or deceleration) is a much simpler problem; it is a subproblem of formulations (6.1) and (6.2) for the case when $v_f = v_0$ (and

just ignore u):

$$\begin{aligned} \min_{v} \quad & \frac{\ell}{v} \\ \text{s.t.} \quad & \left(\begin{array}{l} +\dfrac{1}{2\eta}\rho C_w A_f v^2 \ell \\ +\dfrac{\ell}{\eta}\left(\mu m g \cos(\alpha) + m g \sin(\alpha)\right) \\ +P_{\text{acc}}\left(\dfrac{\ell}{v}\right) \end{array} \right) \leq E_0 \\ & \underline{v} \leq v \leq \overline{v}. \end{aligned} \tag{6.3}$$

Suppose that we require that $|a| \leq R$, instead of using the acceleration bounds for R_{accel} and R_{decel}. We can combine problems (6.1) and (6.2) in the following

way:

$$
\begin{aligned}
\min_{v_0, v_f, u, y} \quad & \frac{\ell}{v_f} + (2y-1)u\frac{\left(v_f - v_0\right)^2}{2v_f} \\
\text{s.t.} \quad & \left(\begin{array}{l} -\frac{u}{8\eta}\rho C_w A_f \left(v_f^2 - v_0^2\right)^2 + \frac{1}{2\eta}\rho C_w A_f v_f^2 \ell \\ +\frac{\ell}{\eta}\left(\mu m g \cos(\alpha) + m g \sin(\alpha)\right) + \frac{1}{2\eta} m\delta\left(v_f^2 - v_0^2\right) \\ + P_{\text{acc}}\left(\frac{\ell}{v_f} + (2y-1)u\frac{\left(v_f - v_0\right)^2}{2v_f}\right) \end{array} \right) \leq E_0 \\
& \underline{v} \leq v_0 \leq \overline{v} \\
& \underline{v} \leq v_f \leq \overline{v} \\
& v_f - v_0 \leq \overline{v} y \\
& v_0 - v_f \leq \overline{v}(1-y) \\
& (2y-1)u\left(\frac{v_f^2 - v_0^2}{2}\right) \leq \ell \\
& u \geq 1/R \\
& y \in \{0, 1\},
\end{aligned}
\tag{6.4}
$$

where $u = \dfrac{1}{|a|}$ and we use the binary variable

$$
y = \begin{cases} 1 & \text{if } v_f \geq v_0 \\ 0 & \text{if } v_f \leq v_0 \end{cases}
$$

We can determine the value of a from the values of u and y. When $v_f = v_0$, it does not matter which value y takes.

One Path Formulation

Suppose that the driver is about to traverse a given path P, where the initial speeds at each node i, v_i, are already decided. At what rate should the driver accelerate or decelerate from v_i to v_j (and then travel the remaining distance at speed v_j) on each arc $(i,j) \in P$ to minimize time, subject to a total budget for energy consumption? This question can be formulated as a linear program, based on the formulation (6.4):

$$\begin{aligned}
\min_{u_{ij}} \quad & \sum_{(i,j)\in P} \left(\frac{\ell_{ij}}{v_j} + (2y_{ij}-1)u_{ij}\frac{\left(v_j - v_i\right)^2}{2v_j} \right) \\
\text{s.t.} \quad & \sum_{(i,j)\in P} \left(\begin{array}{l} -\dfrac{(2y_{ij}-1)u_{ij}}{8\eta}\rho C_w A_f \left(v_j^2 - v_i^2\right)^2 + \dfrac{1}{2\eta}\rho C_w A_f v_j^2 \ell_{ij} \\ + \dfrac{\ell_{ij}}{\eta}\left(\mu m g \cos(\alpha) + mg\sin(\alpha)\right) + \dfrac{1}{2\eta} m\delta\left(v_j^2 - v_i^2\right) \\ + P_{\text{acc}}\left(\dfrac{\ell_{ij}}{v_j} + (2y_{ij}-1)u_{ij}\dfrac{\left(v_j - v_i\right)^2}{2v_j} \right) \end{array} \right) \leq E_0 \\
& (2y_{ij}-1)u_{ij}\left(\frac{v_j^2 - v_i^2}{2}\right) \leq \ell_{ij}, \quad \forall (i,j) \in P \\
& u_{ij} \geq 1/R_{ij}, \quad \forall (i,j) \in P.
\end{aligned} \tag{6.5}$$

The notation for (6.5) is:

- $u_{ij} = \dfrac{1}{|a_{ij}|}$, where a_{ij} is the rate of acceleration or deceleration for arc (i,j).
- $y_{ij} = \begin{cases} 1 & \text{if } v_j \geq v_i \\ 0 & \text{if } v_j < v_i \end{cases}$
  is a data point that indicates whether the car needs to accelerate (1) or de-

celerate (0) from v_i to v_j. Therefore, $a_{ij} = \dfrac{(2y_{ij} - 1)}{u_{ij}}$.

Things to note:

- $(2y_{ij} - 1)u_{ij}\left(\dfrac{v_j^2 - v_i^2}{2}\right) \leq \ell_{ij}$ is the requirement that the driver cannot travel more than length ℓ_{ij} when accelerating or decelerating on arc (i, j).
- We assume that there are no stops or turns along the way.
- The formulation for deceleration is similar.

General Path Formulation

We can extend the formulation (6.4) to the case when we optimize over a path as well.

The initial velocity for arc (i, j) is the final velocity of the previous arc, which is

$$v_{0,i} = \begin{cases} \displaystyle\sum_{k:(k,i)\in A} x_{ki} v_{ki} & i \neq s \\ 0 & i = s \end{cases}$$

where the x_{ki} ensure that we use the velocity of the arc that was actually traveled. We start off with an initial velocity of 0 at the source node, s.

We assume that we choose the rate of acceleration on each arc to be a_{ij}.

Let $u_{ij} = \dfrac{1}{|a_{ij}|}$

Let

$$y_{ij} = \begin{cases} 1 & \text{if } v_{ij} \geq v_{0,i} \\ 0 & \text{if } v_{ij} \leq v_{0,i} \end{cases}$$

Let $M = \max_{(i,j)\in A} \{\overline{v}_{ij}\}$.

The formulation is:

$$
\begin{aligned}
\min_{v_{ij},v_{0,i},u_{ij},y_{ij},x_{ij}} \quad & \sum_{(i,j)\in A}\left(\frac{\ell_{ij}}{v_{ij}}+(2y_{ij}-1)u_{ij}\frac{\left(v_{ij}-v_{0,i}\right)^2}{2v_{ij}}\right)x_{ij} \\
\text{s.t.} \quad & \sum_{(i,j)\in A}\left(\begin{array}{l}
-\dfrac{(2y_{ij}-1)u_{ij}}{8\eta}\rho C_w A_f\left(v_{ij}^2-v_{0,i}^2\right)^2 \\
+\dfrac{1}{2\eta}\rho C_w A_f v_{ij}^2\ell+\dfrac{1}{2\eta}m\delta\left(v_{ij}^2-v_{0,i}^2\right) \\
+\dfrac{\ell}{\eta}\left(\mu mg\cos(\alpha)+mg\sin(\alpha)\right) \\
+P_{\text{acc}}\left(\dfrac{\ell}{v_{ij}}+(2y_{ij}-1)u_j\dfrac{\left(v_{ij}-v_{0,i}\right)^2}{2v_{ij}}\right)
\end{array}\right)x_{ij}\le E_0 \\
& v_{0,i}=\sum_{k:(k,i)\in A}x_{ki}v_{ki}, \quad \forall i\in V, i\neq s \\
& v_{0,s}=0 \\
& \underline{v}_{ij}\le v_{ij}\le \overline{v}_{ij}, \quad \forall (i,j)\in A \\
& v_{ij}-v_{0,i}\le My_{ij}, \quad \forall (i,j)\in A \\
& v_{0,i}-v_{ij}\le M(1-y_{ij}), \quad \forall (i,j)\in A \\
& (2y_{ij}-1)u_{ij}\left(\frac{v_{ij}^2-v_{0,i}^2}{2}\right)\le \ell_{ij}, \quad \forall (i,j)\in A \\
& u_{ij}\ge 1/R_{ij}, \quad \forall (i,j)\in A \\
& y_{ij}\in\{0,1\}, \quad \forall (i,j)\in A \\
& \boldsymbol{x}\in X\subset\{0,1\}^{|A|}.
\end{aligned}
\tag{6.6}
$$

One could probably simplify the formulation (6.6) somewhat by substituting $v_{0,i}$ into the expressions in the formulation and use the fact that $x_{ij}\in\{0,1\}$ implies $x_{ij}=x_{ij}^2=x_{ij}^3=x_{ij}^4=x_{ij}^5$.

If we would like to require that we stop at the final destination, we would add the constraints $v_{it} = 0$ for all $(i, t) \in A$, where t is the sink node. In this case we would be forcing the vehicle to decelerate to 0 and decelerate for the whole traversal of that arc.

6.3 Additional Future Work

Besides the extensions in Sections 6.1 and 6.2, future work could continue to improve the modeling accuracy and include algorithmic enhancements to further improve running time, especially for larger networks. Possible ways to improve modeling accuracy are to incorporate driver behavior and to do additional calibration of the data and parameters used (based on real-world drives, for example). Additional algorithmic enhancements might include parallelization, and further improvements to solving the shortest path problem.

Incorporating driver behavior in to the optimization framework might improve modeling accuracy, as we expect an aggressive driver to consume more energy, but require less drive time. Currently, our model assumes that the driver will drive at the specified speeds v_{ij}, and at most the upper bounds $\overline{v}_{ij}$ (which can be seen as a speed limit). However, given a particular speed limit and route, different people will drive a specified route in different ways. A conservative driver will tend to accelerate and decelerate slowly, smoothly, and as little as possible. This driving behavior requires less energy, but more time than for an aggressive driver. However, some drivers wish to reach their destination faster, and drive more aggressively. We expect a more aggressive driver to:

- Go faster (on average) than the speed limit or a speed that an optimization algorithm suggests.
- Accelerate and decelerate more sharply and more often.

We would like to account for various kinds of aggressive driving, although we neither support any driving behavior that violates state or federal laws nor driving behavior that is unsafe. A possible way to account for driver behavior is to assume that more aggressive drivers will drive faster (for example, increase the velocity bounds by a factor) and consume more energy than the conservative driver. Because we discourage unsafe driving, it might be prudent not to decrease the driving time due to more aggressive driver behavior and to set energy consumption on each arc to be the maximum of the energy consumption due to conservative and due to aggressive driving behavior.

In addition to considering driver behavior, we would like to consider more realistic driving assumptions than having vehicles travel at a constant speed v_{ij} along each arc (i, j) (besides extra time for stops and turns). This leads to considering more realistic drive cycles, where a drive cycle is the velocity of a vehicle as a function of time during the entire drive. Various industry-standard drive cycles can be found in United States Environmental Protection Agency [47]. Velocity can change quickly in a realistic drive cycle, as illustrated by the plot of the industry-standard FTP-75 drive cycle (Figure 6.4). The FTP-75 drive cycle is commonly used to test mileage on gas vehicles (DieselNet [13], which references Appendix I, pages 543–610, of [33]). Incorporating more realistic drive cycles such as the FTP-75 would improve the accuracy of energy consumption.

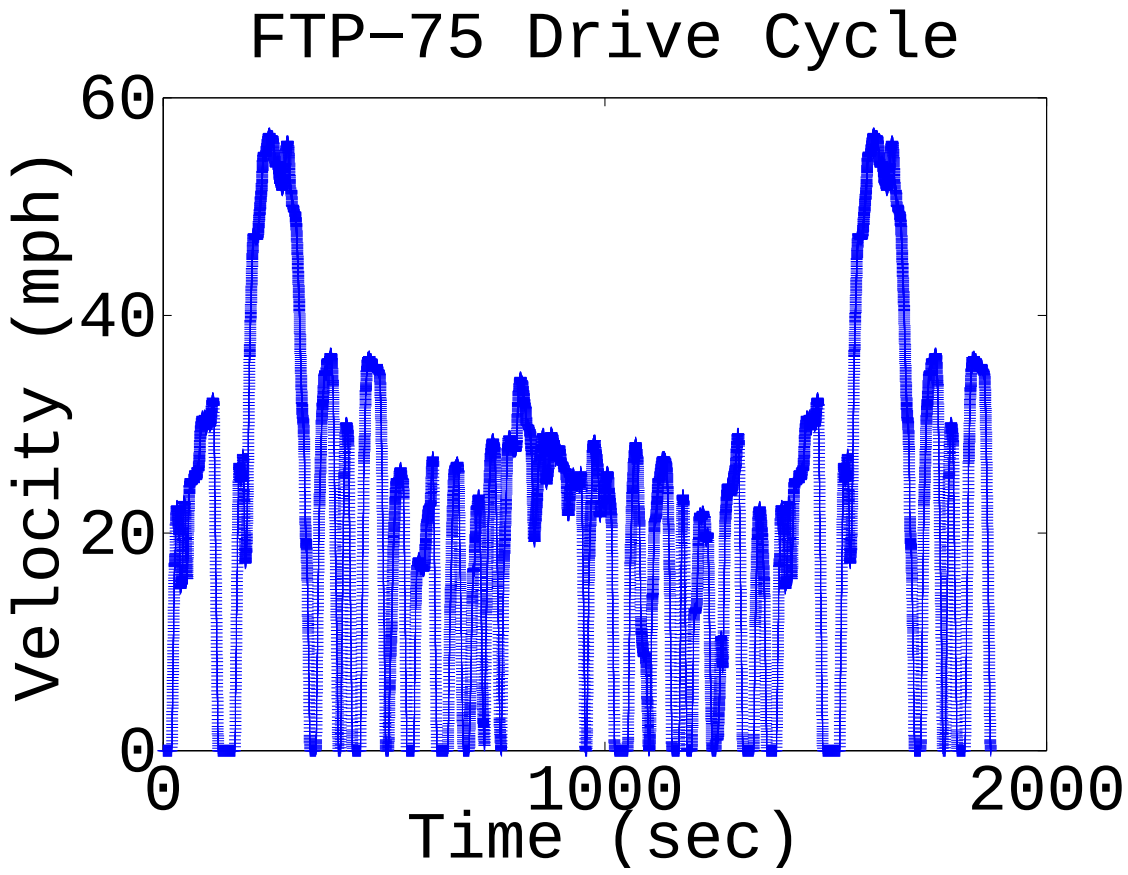

Figure 6.4: Driver velocity can change quickly, as illustrated by this plot of the FTP-75 drive cycle.

Although we feel our algorithms run quickly, more improvement might be possible. First, most of the algorithms do not incorporate parallelization. It might be worth considering how to take advantage of multiple processors in all steps in the algorithm. Second, it might be possible to speed-up the shortest path calculation, perhaps by using a more refined A-star heuristic function or considering other variations of Dijkstra's algorithm.

6.4 Conclusion

We provided an optimization framework for quickly finding routes for battery electric vehicles, and applied it to problems based on road network data. The main routing problem we solved was to minimize the travel time, given a requirement on the total energy consumption, and uncertainty in both time and energy. The optimization framework included a model for energy consumption of a battery electric vehicle (Chapter 2), an optimization-based formulation of the problem (Chapter 3), and algorithms to solve it (Chapter 4). We then used those algorithms on road network data of Massachusetts and Michigan to show that our algorithms could find good routes quickly (Chapter 5). After demonstrating the potential effectiveness of our methodology, we proposed some extensions (Sections 6.1 and 6.2) and additional ideas for future work (Section 6.3).

A key to the success of the optimization framework was incorporating a robust optimization methodology. Incorporating uncertainty can usually make an optimization problem much more difficult to solve, as in the case of incorporating randomness into a shortest path problem using a stochastic shortest path framework (Section 1.3). Stochastic shortest path problems are hard to solve (NP-Hard or #P-Hard) because of the use of random variables. Instead of using random variables, robust optimization uses a minimax approach with uncertainty sets. By selecting the right uncertainty sets, such as polyhedral and ellipsoidal uncertainty sets, the optimization problem incorporating uncertainty is still very manageable.

To be able to solve the main optimization problem efficiently, we leveraged the particular structure of the problem, and we were willing to use heuristics.

One use of the structure was taking advantage of the particular uncertainty sets for polyhedral and ellipsoidal uncertainty, which was a benefit of using robust optimization. We also used the fact that our problem was a path problem. We refined our algorithms to specifically solve a path problem on a network, defined by roads and intersections on a map.

Besides solving an optimization problem quickly, a contribution of this thesis was gathering many sources of data together and applying developed optimization algorithms to real-world data. We used vehicle data from Ford Motor Company to validate a proposed energy model, which was based on principles and formulas from physics and engineering. We obtained road network data from Esri (using ArcGIS), and combined it with elevation data from the US Geological Survey. In addition, we obtained data with traffic signals in Cambridge, MA, and network data from the Massachusetts Department of Transportation, which included average annual daily traffic counts to show how one might be able to further extend the work. Assembling this data together provided the examples to demonstrate the applicability of the work.

Bibliography

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, NJ, 1993.

[2] Andreas Artmeier, Julian Haselmayr, Martin Leucker, and Martin Sachenbacher. The shortest path problem revisited: Optimal routing for electric vehicles. In Rüdiger Dillmann, Jürgen Beyerer, Uwe Hanebeck, and Tanja Schultz, editors, *KI 2010: Advances in Artificial Intelligence*, volume 6359 of *Lecture Notes in Computer Science*, pages 309–316. Springer Berlin / Heidelberg, 2010.

[3] Michael Page Bailey. Constant Access Systems: A General Framework for Greedy Optimization on Stochastic Networks. *Operations Research*, 40(3-Supplement-2):S195–209, May-June 1992. doi: 10.1287/opre.40.3.S195.

[4] M. O. Ball, C. Colburn, and J.S. Provan. Network reliability. In *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, chapter 11, pages 673–762. Elsevier, Amsterdam, 1995.

[5] Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88(3):411–424, 2000. ISSN 0025-5610. doi: 10.1007/PL00011380. URL `http://dx.doi.org/10.1007/PL00011380`.

[6] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, Belmont, Massachusetts, third edition, 2005.

[7] Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98:49–71, 2003. ISSN 0025-5610.

[8] Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization under ellipsoidal uncertainty sets. Available at http://www.mit.edu/~dbertsim/papers/Robust%20Optimization/Robust%20Discrete%20Optimization%20under%20Ellipsoidal%20Uncertainty%20Sets.pdf, March 2004.

[9] Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations Research*, 52(1):35–53, January–February 2004.

[10] Dimitris Bertsimas, David B. Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011.

[11] City of Cambridge GIS. Cambridge, MA (Traffic Signals 2008). Obtained using Massachusetts Institute of Technology GeoWEB, 2008. URL http://library.mit.edu/item/001539399.

[12] CNBC.com. As electric car sales struggle, Obama calls for more funding. Retrieved June 11, 2013, April 2013. URL http://www.cnbc.com/id/100633877.

[13] DieselNet. Emission test cycles: FTP-75. Retrieved July 18, 2013. URL http://www.dieselnet.com/standards/cycles/ftp75.php.

[14] Mehrdad Ehsani, Yimin Gao, and Ali Emadi. *Modern Electric, Hybrid Electric, and Fuel Cell Vehicles: Fundamentals, Theory, and Design*. Power Electronics and Applications Series. CRC Press, second edition, 2010.

[15] Electric Drive Transportation Association. Electric drive sales. Retrieved June 11, 2013, June 2013. URL http://www.electricdrive.org/index.php?ht=d/sp/i/20952/pid/20952.

[16] Esri. World street map basemap. Retrieved through Esri ArcGIS for Desktop 10.1, 2013. URL http://goto.arcgisonline.com/maps/World_Street_Map.

[17] Ford Motor Company. Vehicle drive data, 2011.

[18] Ford Motor Company. Ford's electric vehicle technology: How EVs work. Retrieved May 30, 2013, 2013. URL http://www.ford.com/technology/electric/howevswork/.

[19] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[20] Jeffrey D. Gonder. Route-based control of hybrid electric vehicles. SAE Technical Paper 2008-01-1315, April 2008.

[21] Q. Gong, P. Tulpule, V. Marano, S. Midlam-Mohler, and G. Rizzoni. The role of ITS in PHEV performance improvement. In *American Control Conference (ACC)*, pages 2119–2124, June 2011.

[22] A. Haddoun, M. E. H. Benbouzid, D. Diallo, R. Abdessemed, J. Ghouili, and K. Srairi. A loss-minimization DTC scheme for EV induction motors. *Vehicular Technology, IEEE Transactions on*, 56(1):81–88, Jan. 2007.

[23] Refael Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, 1992. ISSN 0364765X. URL `http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=4475397&site=ehost-live`.

[24] Iqbal Husain. *Electric and Hybrid Vehicles: Design Fundamentals*. CRC Press, second edition, 2011.

[25] Svante Janson. One, two and three times log n/n for paths in a complete graph with random weights. *Combinatorics, Probability and Computing*, 8(4):347–361, 1999. ISSN 0963-5483. doi: http://dx.doi.org/10.1017/S0963548399003892.

[26] V. G. Kulkarni. Shortest paths in networks with exponentially distributed arc lengths. *Networks*, 16(3):255–274, 1986.

[27] V. G. Kulkarni. Minimal spanning trees in undirected networks with exponentially distributed arc weights. *Networks*, 18(2):111–124, 1988.

[28] V. G. Kulkarni and V. G. Adlakha. Maximum flow in networks with exponentially distributed arc capacities. *Communications in Statistics – Stochastic Models*, 1(3):263–289, 1985.

[29] V. G. Kulkarni and V. G. Adlakha. Markov and Markov-Regenerative PERT networks. *Operations Research*, 34(5):769–781, 1986. ISSN 0030364X. URL `http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=4480794&site=ehost-live`.

[30] Perry MacNeille, Oleg Gusikhin, Mark Jennings, Ciro Soto, and Sujith Rapolu. Integration of traffic simulation and propulsion modeling to estimate energy consumption for battery electric vehicles. In Nuno Pina, Janusz

Kacprzyk, and Joaquim Filipe, editors, *Simulation and Modeling Methodologies, Technologies and Applications*, volume 197 of *Advances in Intelligent Systems and Computing*, pages 3–19. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-34335-3.

[31] Massachusetts Department of Transportation. Road inventory file, 2012. URL `http://www.massdot.state.ma.us/planning/Main/MapsDataandReports/Data/GISData/RoadInventory.aspx`.

[32] S. Mirzaei and K Krishnan. A MINLP formulation of energy efficient location routing problem. In T. Doolen and E. Van Aken, editors, *Proceedings of the 2011 Industrial Engineering Research Conference*, 2011. [CD-Rom].

[33] Office of the Federal Register National Archives and Records Administration. Code of Federal Regulations, Title 40, Part 86. Revised as of July 1, 2012, July 2012.

[34] OpenStreetMap contributors. OpenStreetMap data, 2013. Data is available under the Open Database Licence. See `www.openstreetmap.org/copyright`.

[35] George H. Polychronopoulos and John N. Tsitsiklis. Stochastic shortest path problems with recourse. *Networks*, 27(2):133–143, March 1996.

[36] J. Scott Provan. A polynomial-time algorithm to find shortest paths with recourse. *Networks*, 41(2):115–125, 2003. URL `http://dx.doi.org/10.1002/net.10063`.

[37] Christopher P. Quigley. The use of vehicle navigation information and prediction of journey characteristics for the optimal control of hybrid and electric vehicles. SAE Technical Paper 2011-01-1025, April 2011.

[38] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, third edition, 2010.

[39] Martin Sachenbacher, Martin Leucker, Andreas Artmeier, and Julian Haselmayr. Efficient energy-optimal routing for electric vehicles. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[40] Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle routing problem with time windows and recharging stations. Technical report, University of Kaiserslautern, February 2012.

[41] R.W. Sinnott. Virtues of the haversine. *Sky and Telescope*, 68(2):159, 1984.

[42] Timothy M. Sweda and Diego Klabjan. Finding minimum-cost paths for electric vehicles. In *Electric Vehicle Conference (IEVC), 2012 IEEE International*, pages 1–4, March 2012.

[43] Fazal Syed, Swathi Nallapa, Allen Dobryden, Carrie Grand, Ryan McGee, and Dimitar Filev. Design and analysis of an adaptive real-time advisory system for improving real world fuel economy in a hybrid electric vehicle. SAE Technical Paper 2010-01-0835, April 2010.

[44] Edward Dean Tate, Jessy W. Grizzle, and Huei Peng. Shortest path stochastic control for hybrid electric vehicles. *International Journal of Robust and Nonlinear Control*, 18(14):1409–1429, 2008. ISSN 1099-1239.

[45] TomTom. StreetMap North America: U.S. and Canada detailed streets. Published by Esri for ArcGIS 10.1, 2012.

[46] Nora Touati-Moungla and Vincent Jost. Combinatorial optimization for electric vehicles management. In *International Conference on Renewable Energies and Power Quality (ICREPQ11)*, pages 504–506, 2011.

[47] United States Environmental Protection Agency. Dynamometer drive schedules. Retrieved July 18, 2013. URL `http://www.epa.gov/nvfel/testing/dynamometer.htm`.

[48] U.S. Geological Survey. National elevation dataset, 2011. URL `http://nationalmap.gov`.

[49] Michael J. Veenstra and Chris Gearhart. A pareto frontier analysis of renewable-energy consumption, range, and cost for hydrogen fuel cell vs. battery electric vehicles. SAE Technical Paper 2012-01-1224, April 2012.

[50] White House Office of the Press Secretary. Remarks by the president in state of union address. Retrieved June 10, 2013, January 2011. URL `http://www.whitehouse.gov/the-press-office/2011/01/25/remarks-president-state-union-address`.

[51] Owen Worley, Diego Klabjan, and Timothy M. Sweda. Simultaneous vehicle routing and charging station siting for commercial electric vehicles. In *Electric Vehicle Conference (IEVC), 2012 IEEE International*, pages 1–3, March 2012.

[52] Hai Yu, Finn Tseng, and Ryan McGee. Driving pattern identification for EV range estimation. In *Electric Vehicle Conference (IEVC), 2012 IEEE International*, pages 1–7, March 2012.

[53] Chen Zhang and A. Vahidi. Route preview in energy management of plug-in hybrid vehicles. *Control Systems Technology, IEEE Transactions on*, 20(2): 546–553, March 2012. ISSN 1063-6536.