Towards optimal route planning for solar-powered battery electric vehicles

R Luies

0000-0003-2246-313X

Thesis accepted in fulfilment of the requirements for the degree *Philosophiae Doctor* in Industrial Engineering at the Potchefstroom Campus of the North-West University

Supervisor Prof SE Terblanche

Graduation August 2021

Student number: 23511354

The art of programming is the art of organising complexity.
- Edsgar W. Dijsktra

ACKNOWLEDGEMENTS

Throughout the writing of this thesis, I have received a great deal of support and assistance. I would like to thank my supervisor, Prof Fanie Terblanche, for guidance and support, especially with mathematical modelling and writing quality C++ code. Your belief in my abilities was a significant driving factor in the completion of this thesis.

I would like to thank Danie Human and Piet van Huyssteen for providing insights into the problem and providing vehicle characteristic estimates for the North-West University solar car. This was very useful regarding the implementation of the simulation aspects.

I would like to thank my fiancée Anneke Lourens for the emotional support you provided the past years and for reading through the thesis countless times.

I would like to thank my parents, Susan Luies and Marthinus Luies, for contributing to my interests. My mother had a significant influence on my love for mathematics and my father had a substantial impact on my passion for engineering.

I would like to thank my brothers Juan Luies and Ian Luies for providing happy distractions with video games and my sister Chantel Luies for all the laughs we had about silly things. In addition, I would like to thank my to-be in-laws for their support.

ABSTRACT

Given the recent advances in battery technologies, Battery Electric Vehicles (BEVs) are more in demand since they are considered a better option than Internal Combustion Engine Vehicles (ICEVs). There are some drawbacks to using BEVs; for example, driving ranges are shorter compared to ICEVs, and limited charging station infrastructure may be available in certain parts of the world. Furthermore, batteries mounted in BEVs are the leading cause of high acquisition costs, and there are also some technical limitations since the maximum battery capacity degrades over time. These disadvantages negatively affect the adoption of BEVs.

There is expected to be an increase in BEV adoption around the world since they require less expensive and less frequent maintenance than ICEVs. A significant problem with BEVs is range anxiety, and route planning may help mitigate this. BEVs need frequent recharging during trips, which renders existing route planning methods used for ICEVs infeasible. Limited driving range, lack of charging stations and possible long charging times of BEVs affects the route choices significantly. BEV route planning may also lower BEVs' energy consumption and, consequently, the travel-cost.

In the thesis, Mixed Integer Linear Programming (MILP) models are proposed to address route planning for BEVs. Multiple factors such as wind speed and -direction, solar irradiation in the case of a solar panel mounted on the vehicle, vehicle acceleration and drive-train efficiency are incorporated to determine optimal routes.

As part of this thesis's case study, the models are adapted to race strategies for competing in the Sasol Solar Challenge. The Sasol Solar Challenge is a biennial competition where multiple teams worldwide design and build solar-powered vehicles to travel across South Africa over eight days. In recent years, the Sasol Solar Challenge has drawn considerable awareness to the development of solar-powered vehicles. In the past years, most of the effort focused on the cars' mechanical quality and efficiency, but teams gave little attention to the race strategy. An important task is to determine at which stages of the route an automobile should accelerate, decelerate, or maintain speed to use available energy efficiently. When considering a solar-powered car, accessible weather- and elevation data for the whole route is a requirement when determining a race strategy.

The solar-powered vehicle is simulated, using vehicle characteristics, weather- and elevation data, and an optimisation model to determine the best strategy for a given solar-powered automobile and route. These simulation- and optimisation models can also help with decisions regarding the design of solar-powered vehicles. A short-term advantage of this approach is the connection with current vehicle technologies regarding energy efficiency when travelling a known route, i.e. a strategy to assist a driver in obtaining better fuel economy.

Keywords: *mixed-integer linear programming, sasol solar challenge, battery electric vehicle, route planning*

CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF TABLES

ACRONYMS

AFS alternative fuel station. xiii, 2
AFV alternative fuel vehicle. xiii, 2

B&B branch and bound. xi, xiii, 9, 12, 14, 15
BEV Battery Electric Vehicle. v, vii, ix, xiii, 1–4, 22, 24–26, 28–30, 33, 40, 43, 51, 53, 54, 58–60, 62–65, 73, 76, 84, 87, 88

CRP Customisable Route Planning. xiii, 3
CSP Constrained Shortest Path. xiii, 3

DBCA Density-Based Clustering Algorithm. xiii, 2
DP dynamic programming. xiii, 71, 72

GA Genetic Algorithm. xiii, 2, 6
GIS Geographic Information System. vii, xiii, 26, 27, 51
GPOPS Gauss Pseudospectral Optimisation Software. xiii, 72

ICEV Internal Combustion Engine Vehicle. v, xiii, 1
ILP Integer Linear Programming. ix, xiii, 6, 12–16

LP Linear Programming. ix, xiii, 2, 5, 9, 12–15, 21, 55

MCWS Modified Clarke and Wright Saving. xiii, 2
MILP Mixed Integer Linear Programming. v, xiii, 2–4, 15, 17, 24, 25, 29, 38, 40, 47, 50, 51, 56, 72, 73, 75, 87

NWP numerical weather prediction. xiii, 36

PSO Particle Swarm Optimisation. xiii, 2

SoC State of Charge. xiii, 1, 2, 65, 79
SOS Special Ordered Set. xiii, 74
SOS2 Special Ordered Set of Type 2. xiii, 4, 40, 47, 50, 53, 54, 56–61, 63, 87
SOS3 Special Ordered Set of Type 3. xiii, 4, 40, 53–62, 75, 87

TSP Travelling Salesman Problem. xiii, 2

VRP Vehicle Routing Problem. xiii, 2

1

INTRODUCTION

1.1 CONTEXTUALISATION

The transportation sector accounts for around 20% of the total energy consumption in South Africa and 25% worldwide [1, 2]. Route planning may lower a vehicle's energy consumption and, consequently, the travel-cost and greenhouse gasses during operation. Given the recent advances in battery technologies, Battery Electric Vehicles (BEVs) are more in demand since they are considered a better option than vehicles with internal combustion engines. There are some drawbacks to using BEVs, such as shorter driving ranges than Internal Combustion Engine Vehicles (ICEVs) and limited charging station infrastructure in South Africa. Batteries mounted in BEVs are the leading cause of the high acquisition cost, and there are also some technical limitations since the maximum battery capacity degrades over time. These disadvantages negatively affect the adoption of BEVs.

According to Pelletier et al. [3], batteries should be replaced every 5 to 10 years, or after 1000 to 2000 cycles with large State of Charge (SoC) variations. Factors that influence battery degradation include over-charging, over-discharging, long storage with high SoC, the depth of discharge and temperatures outside the battery ratings.

In 2016 Grunditz and Thiringer [4] analysed over 40 BEVs which falls into small, medium-large, high-performing and sports car categories. All the BEVs have lithium-based batteries, with capacities varying between 12 and 90 kWh with a maximum travelling distance of between 85 and 528 km. The average medium-sized travelling distance was 250 km with a battery capacity of 30 kWh.

Most modern BEVs can recuperate energy during deceleration or when going downhill, as long as it does not exceed the capacity of the maximum energy capacity of the battery. BEVs need frequent recharging during trips, which renders existing route planning methods used for ICEVs infeasible. Limited driving range, lack of charging stations and possible long charging times of BEVs affects the route choices significantly.

There are financial incentives to use BEVs as a courier service; the authors of [5] noted that using a large number of BEVs can be more cost-effective than ICEVs. Several factors play a role: high daily distances, low speeds, frequent stops, traffic congestions, the reduced cost of BEVs by tax incentives, and long-term planning. Due to financial incentives, BEVs are more likely to be used for last-mile deliveries since routes are short, stops are frequent, driving speeds are slow, and production noise is low [6]. Schiffer et al. [7] concluded that for a five-year plan, the operational limitations for BEVs is a maximum delivery radius of 190 km from the depot due to a lack of close locations with charging stations.

1.1.1 *Problem statement*

When a solar panel is attached to a BEV the driving range can potentially extend up to 30%, resulting in more charging station options [8]. For hybrid vehicles, the fuel economy can improve up to 60% when a solar panel is attached to the hybrid vehicle, depending on the driving habits and weather conditions [9]. Manually planning a route for a solar-powered hybrid or BEV is not

easy since early decisions affect later decisions. A decision support system would be beneficial for route planning of solar-powered BEVs.

Most models and algorithms that exist for BEV route planning are heuristic and addresses the Vehicle Routing Problem (VRP), or only account for a few significant factors that influence the route choices and cannot easily be extended for solar-powered BEVs [10–17]. These models and algorithms assume that charging stations (and in one case regenerative braking) is the only way to charge batteries of BEVs. The main reason these models and algorithm cannot easily be extended to accommodate solar-power BEV is the lack of time-dependent variables such as solar irradiation, cloud coverage, wind speed and wind direction at a given time. The current state of literature is discussed in section 1.1.3 and section 1.1.2.

1.1.2 *Related work on the vehicle routing problem*

The VRP [16] is closely related to the Travelling Salesman Problem (TSP) [18], since it is a generalisation of the TSP. The VRP consists of finding a collection of routes from one or several depots to several geographically scattered cities or customers. The authors of [19] introduced an exact algorithm that solves the VRP, with capacity and distance constraints. The algorithm does not include extending the vehicle's distance limitation by refuelling on the route.

The authors of [17] introduced the green vehicle routing problem, formulated as a Mixed Integer Linear Programming (MILP) problem, which seeks tours for an alternative fuel vehicle (AFV), which visits a subset of vertices that may include alternative fuel stations (AFSs). The AFV starts and ends at the depot, with the objective to minimise the total travelling distance. Erdoğan and Miller-Hooks [17] used two construction heuristics, the Modified Clarke and Wright Saving (MCWS) heuristic and the Density-Based Clustering Algorithm (DBCA) with a customised improvement technique. The MCWS heuristic terminates with a set of tours that forms a feasible solution to the relaxed MILP problem. The DBCA exploits the problem's spatial properties, and builds clusters, the MCWS heuristic runs on each cluster. The authors used two sets of datasets, the first was 40 test problems with 20 customers, and the larger has 12 test problems, with customers varying between 111 and 500. The solutions' feasibility depends on locations of the AFSs and the subset of vertices (customers) that needs to be visited.

Aurélien et al. [20] proposed an improved MILP formulation that accounts for a more realistic, non-linear charging relationship between the time spent charging and the energy the battery gains during the time. The authors developed an arc-based tracking for SoC and time which outperforms the classic node-based tracking. They presented a heuristic and exact algorithm to find charging decisions given a route. Their computational experiments show that this alternative tracking strategy drastically improves the results because the arc-based formulation had a much tighter Linear Programming (LP) relaxation gap, compared to the node-based formulation.

1.1.3 *Related work on route planning for a single vehicle*

Umair and Sadiq [13] implemented a Particle Swarm Optimisation (PSO) based algorithm that can handle multiple constraints for the single-vehicle route planning problem. They compare the implementation to another heuristic and a Genetic Algorithm (GA) based implementation. Overall the PSO based algorithm obtained better objective values than the GA implementation and the heuristic (H_MCOP) proposed by Turgay Korkmaz [21] to solve the multi-constrained path problem.

Sweda et al. [14] modelled the problem of finding a minimum-cost path for a BEV when the vehicle must recharge along the way as a dynamic program. The authors prove that the optimal state space is discrete under certain assumptions that allow using a simple backwards recursion algorithm that guarantees an optimal solution. The dynamic program cannot account for recuperated energy through regenerative braking. The dynamic program is an elegant formulation, but the assumptions oversimplify the problem and modifications to the model are not trivial.

Artmeier et al. [10] formulated an energy-efficient routing algorithm, which is a particular case of the Constrained Shortest Path (CSP) problem on an energy graph representing the energy consumption. The implemented algorithm had a worst-case complexity of $\mathcal{O}(n^3)$. This approach does not allow variables to change over time, such as wind speed and -direction.

Eisner et al. [11] consider the problem of electric vehicle route planning, which accounts for limited energy supply and the ability to recuperate energy. The authors apply the Bellman-Ford algorithm [22, 23] since the problem has negative weights in the input graphs. They also employ a generalisation of Johnson's potential shifting technique [24] to the negative edge cost functions to make Dijsktra's algorithm [25] applicable.

The authors of [15] studied the problem of electric vehicle route planning, where an important aspect is computing paths that minimise energy consumption. The authors introduce a practical approach to optimise for the energy consumption of a BEV between two locations. The method is an extension on Customisable Route Planning (CRP) of Delling et al. [26]. The implementation achieves query times below 5 ms on average and is fast enough for interactive applications. Although this approach is practical in terms of runtime, it oversimplifies the problem and lacks wind speed and -direction. It also does not have extensibility, such as accounting for solar-powered vehicles.

The authors of [27] proposed an integrated route and charging planning for BEVs, and take partial charging, non-linear charging functions as well as serve time into consideration. The experimental result shows that it leads to infeasible or excessively costly solutions when ignoring the partial and non-linear charging process features.

In a recent study, Chen et al. [28] investigated the tour planning problem for BEVs and proposed a MILP model with bi-objective functions. Initially a non-linear model was given and transformed into a MILP. The authors made some assumptions that simplifies the problem. A single BEV type is assumed with constant range, fixed travel time and energy consumption between nodes. The optimisation model used in [28] is provided in 2.5.1.

1.2 CONTRIBUTIONS

In this thesis, a path-based and flow conservation MILP model to solve the single-vehicle routing problem for solar powered BEVs is described and formulated. The advantages and disadvantages of the different modelling techniques are discussed. Empirical results regarding computation time, memory usage and solution quality are presented to determine which model is feasible to use as a decision support tool for route planning and BEV design. The models aim to incorporate as much detail as possible, with the advantage of extending the models or simulations with ease to solve closely related problems. The roads on a route are divided into smaller sections, of which time and velocity-dependent energy graphs are calculated. The base models are extended to account for energy used when a BEV accelerates. Later the models are also extended to allow charging

stations to be defined, along with any arbitrary charging function, to accommodate charging stations with different capacities. Two different methods are used to model energy graphs, and a comparison between the methods in terms of computation time and solution quality is provided. The path-based model is transformed into a heuristic to lower the total runtime at the cost of lower solution quality. The path-based model is changed to determine a race strategy for the Sasol Solar Challenge. The following is a list of the contributions developed in this thesis;

- a generalised path-based model for solar powered BEVs with Special Ordered Set of Type 3 (SOS3) energy graphs,
- a generalised flow conservation model for solar powered BEVs with SOS3 energy graphs,
- a generalised path-based model for solar powered BEVs with Special Ordered Set of Type 2 (SOS2) energy graphs,
- a generalised flow conservation model for solar powered BEVs with SOS2 energy graphs,
- a heuristic implementation of the path-based model using the k-shortest path algorithm,
- an extended base model to account for the acceleration,
- an extended base model to account for configurable charging stations,
- a modified path-based model to be used as a decision support tool for the Sasol Solar Challenge.

1.3 SUMMARY AND THESIS LAYOUT

This study focuses on route planning for a single BEV, which takes factors into account, such as the presence of a solar panel, regenerative braking, and acceleration. These extra factors, especially the solar panel presence, introduce more complexity since they are time-dependent and significantly influence the recuperation of energy. Most of the work done in the literature is either a simple model that is exact or more complex but heuristic. Within this thesis, MILP formulations are proposed that include the extra factors. For that, the necessary details are given on optimisation theory in Chapter 2 related to the formulation and solutions of MILP problems. Chapter 3 elaborates on the logic behind the formulation of BEV route planning with some improvements on the initial formulations. In Chapter 4, different optimisation formulations are compared by using simulated data; the practicality of these approaches are analysed. Chapter 5 studies the feasibility of the proposed models for the Sasol Solar Challenge, a solar car race where maximising the total distance over multiple days and geographic locations is the objective. Finally, Chapter 6 concludes the thesis and highlights future work, such as algorithmic improvements for scalability and usability of the models in other areas.

2

OPTIMISATION THEORY

2.1 INTRODUCTION AND CHAPTER OVERVIEW

This chapter gives a brief introduction to the mathematical and algorithmic concepts used in the thesis. Section 2.1.1 provides an introduction to an overview of exact and heuristic algorithms used in optimisation. Throughout the thesis, we focus more on models solved with exact algorithms, such as Linear Programming (LP) models solved with the simplex method. Specialised heuristic algorithms can be used to find feasible or partial solutions to $\mathcal{NP}$-complete problems. $\mathcal{NP}$-complete is a class which contains problems where there is no known way to find solutions efficiently, but the solutions can be verified quickly (polynomial time). We elaborate on different complexity classes in Section 2.2. Section 2.3 covers the basic concepts of linear programming, integer linear programming, and algorithms to solve linear programs and integer linear programs. The input data is modelled as directed graphs, which suffices an introduction to the basic concepts of graph theory. This chapter may be used as a reference for notation, especially Section 2.5. The reader may skip these concepts if familiar with the content.

2.1.1 *Optimisation*

The simplest case of optimisation consists of minimising or maximising a real function by systematically selecting from an allowed set of inputs $\mathcal{I}$ and choosing the best input. Given a function $f : \mathcal{I} \to \mathbb{R}$ we seek an element $\boldsymbol{x}^* \in \mathcal{I}$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{I}$ in minimisation problems, and $f(\boldsymbol{x}^*) \geq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{I}$ in maximisation problems. The corresponding minimisation mathematical program is:

$$\begin{aligned} \min_{x} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{I}. \end{aligned} \tag{2.1}$$

It is important to distinguish between local- and global optima since some algorithms cannot guarantee optimality. A point $\boldsymbol{x}^*$ is considered a local minimum if there exists some $\epsilon > 0$ such that, for all $\boldsymbol{x} \in \mathcal{I}$ within a distance ϵ of x^*, $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$. With the latter statement, when $f(\boldsymbol{x}^*) \geq f(\boldsymbol{x})$, the point $\boldsymbol{x}^*$ is considered a local maximum.

When complete search space is explored, the global optima can be determined. For a minimisation problem, a point $\boldsymbol{x}^*$ is considered the global minimum if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{I}$. For a maximisation problem, a point $\boldsymbol{x}^*$ is considered the global maximum if $f(\boldsymbol{x}^*) \geq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{I}$. Figure 1 is a graphical representation of these statements.

Heuristic Algorithms and Meta-Heuristics

When the complexity of a problem is too high, e.g., if it is not solvable to optimality given a certain amount of time, heuristic algorithms are preferred [29]. Some specialised heuristic algorithms can compute near-optimal and sometimes optimal solutions, but there is no solution quality guarantee. Even if a heuristic algorithm results in an optimal solution by accident, there is no way to prove optimality with a heuristic algorithm. A heuristic algorithm does not explore the complete search space that globally improves an objective function.

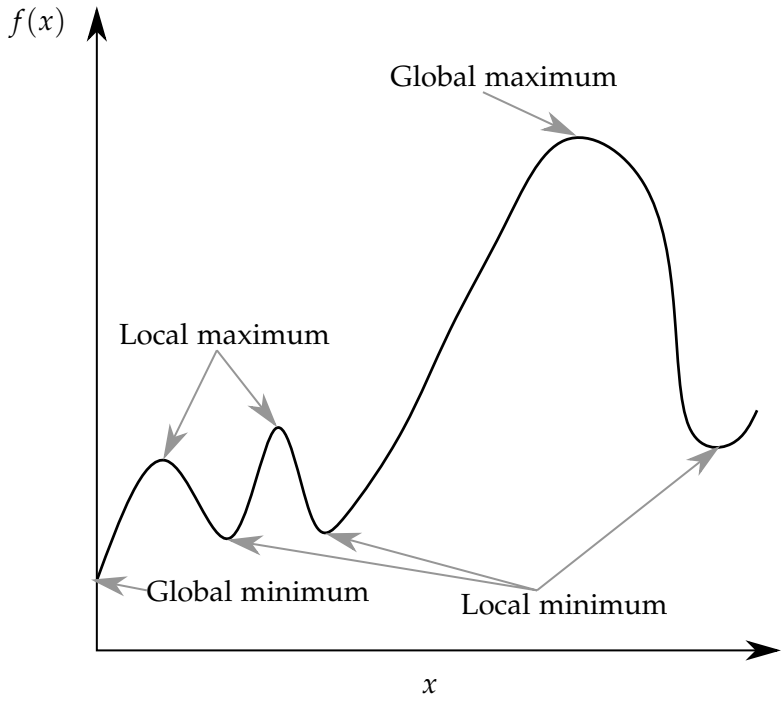


Figure 1: Local- and global optima for $f(x)$.

Heuristic algorithms are problem-specific algorithms, created to solve a particular problem, while meta-heuristics are general-purpose algorithms that can be applied to solve optimisation problems, e.g. evolutionary algorithms. Genetic Algorithms (GAs) are the most common evolutionary algorithms used, based on the principle of natural selection. These types of algorithms were first introduced by [30]. GAs offer significant benefits for optimisation problems since the transformation between Integer Linear Programming (ILP) model and a GA variant is trivial. GAs simulate survival of the fittest among individuals within a population. Initially, the algorithm generates a random population; then the algorithm uses three operators to improve the solution. The algorithm calculates the fitness value of each individual via a fitness function, which is normally the same as the objective function of a similar linear programming problem. The probability of survival is higher for individuals with a better fitness value. The goal of the selection operator is to reduce the population by removing individuals with a stochastic function. The cross-over operator chooses two solutions and mixes the genomes to produce an offspring solution. Random alterations of the genome are considered mutations. The algorithm alters new genomes generated by cross-over operations, usually with a low mutation rate. The mutation decreases the chance a new solution gets stuck in a local optimum. Other meta-heuristics include simulated annealing [31], iterated local search [32], ant colony optimisation [33] and particle swarm optimisation [34].

Exact Algorithms

Throughout this thesis, we are interested in models and methods that guarantee optimality, or atleast provide a quantification of solution quality. When heuristics do feature in this thesis, they are used in correspondence with an exact algorithm, e.g. using heuristic solutions to warm-start exact algorithms. Exact algorithms guarantee solution quality and terminate once an optimal solution is proven. Exact algorithms explore the entire search space that improves the current

objective function. For arbitrary non-linear problems, proving optimality is not easy, but for specific linear problems, it is possible. When the constraints of the optimisation problem form a convex polytope in $\mathbb{R}^n_+$, and the objective function is also linear; we can prove optimality once an objective value can no longer improve. More details are given in Section 2.3.

2.2 COMPUTATIONAL COMPLEXITY

The algorithms used to interpret and execute the implementation of a mathematical optimisation model, whether heuristic or exact, have some computational complexity associated with them [35]. Space- and time-limited computations, given an abstract model of computation, are the measure of these complexities.

Alan Turing introduced the concept of an abstract model of computation, which manipulates data elements on a vector of data according to a table of rules [36]. A Turing machine is considered deterministic when each state can only result in a single action, whereas with a non-deterministic Turing machine each state can result in multiple actions, which results in multiple states, this is illustrated with Figure 2.

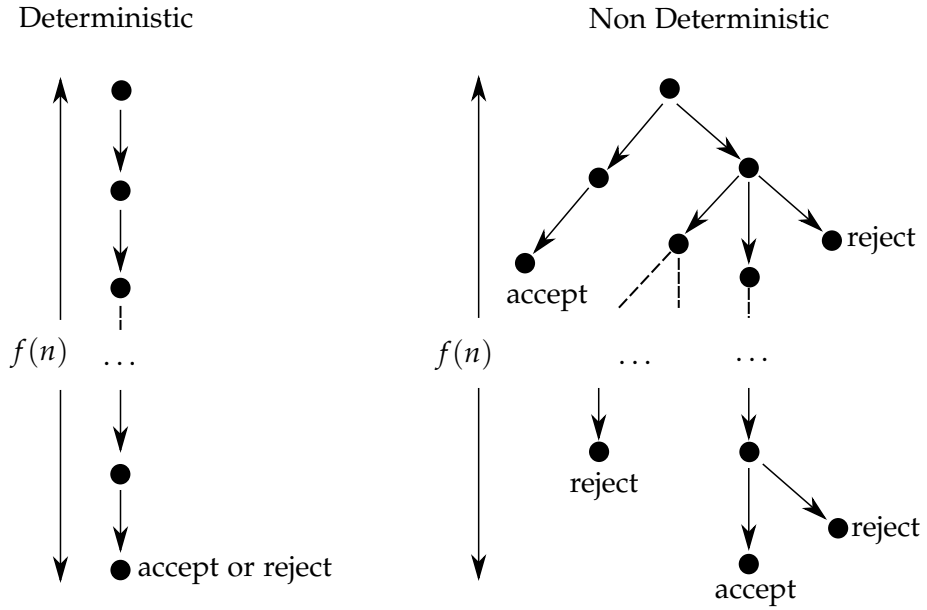


Figure 2: Visualisation of states and actions of a deterministic Turing machine (left) and a non-deterministic Turing machine (right) [37].

Many complexity classes are defined in terms of DTIME, which contains all the problems that can be solved in limited time with a deterministic Turing machine, given by the function $f(n)$. The complexity class in DTIME for function $f(n)$ is expressed as DTIME$(f(n))$.

DEFINITION 1: The complexity class $\mathcal{P}$ is a set of all problems that can be solved with a deterministic Turing machine in DTIME, and expressed as a polynomial function of DTIME,

$$\mathcal{P} = \bigcup_{k \in \mathbb{N}} \text{DTIME}(n^k).$$

Complexity classes for non-deterministic Turing machines that can be solved in limited time are described in NTIME.

DEFINITION 2: The complexity class $\mathcal{NP}$ is a set of all problems that can be solved with a non-deterministic Turing machine in NTIME, and can be expressed as a polynomial function of NTIME,

$$\mathcal{NP} = \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k).$$

An important property of problems in $\mathcal{P}$ and $\mathcal{NP}$, is the ability to verify the solution using a deterministic Turing machine in polynomial time. The problems in $\mathcal{NP}$ contain all the solvable problems in $\mathcal{P}$, but there exist problems harder than the problems in $\mathcal{NP}$, e.g. problems we cannot verify in polynomial time with a deterministic Turing machine, which we call $\mathcal{NP}$-hard problems. The class $\mathcal{NP}$-complete is defined as the hardest problems in $\mathcal{NP}$ since solutions to these problems can be verified in polynomial time with a deterministic Turing machine, but are only solvable in polynomial time with a non-deterministic Turing machine. The unsolved problem of $\mathcal{P}$ versus $\mathcal{NP}$, ask whether the solution to any problem that we can verify in polynomial time can also be solved in polynomial time. Figure 3 illustrates the effect of $\mathcal{P} = \mathcal{NP}$, and $\mathcal{P} \neq \mathcal{NP}$ would have on the complexity classes with an Euler diagram.

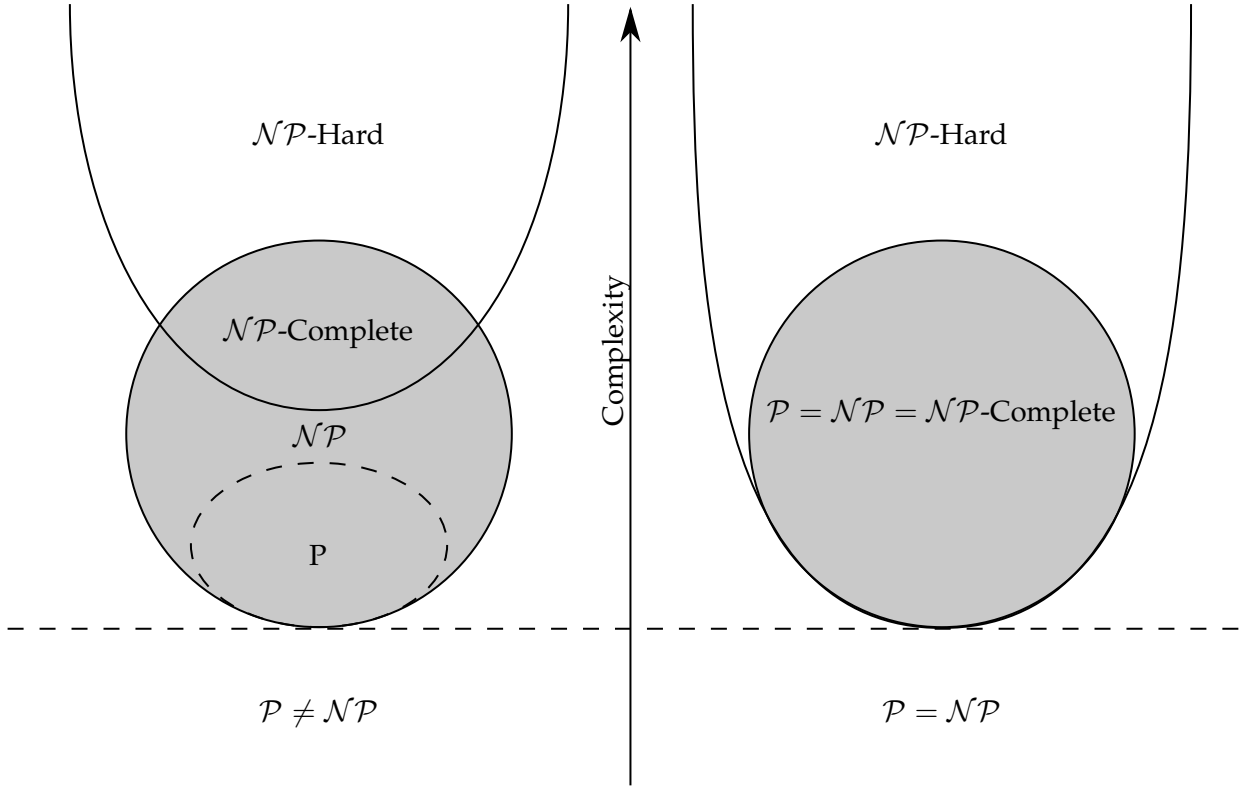


Figure 3: Euler diagram for complexity classes $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-complete, and $\mathcal{NP}$-hard, when $\mathcal{P} \neq \mathcal{NP}$ (left) and $\mathcal{P} = \mathcal{NP}$ (right).

The big $\mathcal{O}$ notation classifies the complexity of algorithms for a Turing machine, usually describing the dominant order at which the worst-case running time and space requirements grow as the input size n increase. Associated with the big $\mathcal{O}$ notation are several types of other bounds on asymptotic growth rates, such as best-case analysis [38]. The worst-case analysis has been criticised for being too pessimistic since it ignores the correlated effects of the operation on data structures. An alternative is an average-case analysis, but it requires some notion of devising a probability distribution over inputs, which entails the collection of input data. In such a case,

the average-case complexity may be more accurate than the worst-case analysis [39]. Tarjan [40] addressed the probabilistic methods of complexity analysis with a method to consider expensive and cheap operations over the whole series of operations of an algorithm; this method is known as amortised complexity analysis.

2.3 LINEAR PROGRAMMING

In this section, we discuss the underlying principles on which the mathematical models from Chapter 3 and Chapter 5 are based on. Firstly, we give a brief introduction to linear programming, which includes the theory of duality, the simplex method, and finally the branch and bound (B&B) and branch and cut algorithm for integer programs.

A linear program maximises or minimises a linear objective function with linear inequality or equality constraints. The general linear program can be described as,

$$p_{LP} = \max\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \in \mathbb{R}^n_+\} \tag{2.2}$$

where A is an $m \times n$ matrix, $\boldsymbol{c}^T$ is a row vector with n elements, $\boldsymbol{b}$ is a column vector with dimension m and $\boldsymbol{x}$ is a column vector with dimension n. It is important to mention that any linear equality can be replaced with two linear inequalities, e.g.

$$c_1x_1 + c_2x_2 = b_1 \tag{2.3}$$

can be replaced with

$$\begin{aligned} -c_1x_1 - c_2x_2 &\leq -b_1, \\ c_1x_1 + c_2x_2 &\leq b_1, \end{aligned} \tag{2.4}$$

$\forall\ x_1, x_2 \in \mathbb{R}_+$.

2.3.1 *Duality*

Duality theory applies to general linear programs and explores the relationship between the solutions of paired linear programs. One problem is called the primal, and the other the dual. It does not matter which problem is called the primal, since the dual of a dual is the primal, but for demonstration, we use the symmetrical linear program (2.2) as the primal problem. The dual is then defined as

$$d_{LP} = \min\{\boldsymbol{b}^T\boldsymbol{y} : A^T\boldsymbol{y} \geq c^T, \boldsymbol{y} \in \mathbb{R}^m_+\}, \tag{2.5}$$

where $\boldsymbol{y}$ is a column vector with dimension m. Feasible solutions to the dual problem provide an upper-bound to the primal problem p_{LP} and feasible solutions to the primal give lower-bounds to the dual d_{LP}. If the primal has an unbounded optimal value, the dual is infeasible. These properties are described by the following well-known theorem:

THEOREM 1 (DUALITY THEOREM): Duality theorem states that:

1. if the primal has an optimal solution, then so does the dual, and $p_{LP} = d_{LP}$;
2. if the primal is unbounded, $p_{LP} = \infty$, then the dual is infeasible;
3. if the primal is infeasible, then the dual is either infeasible or unbounded.

THEOREM 2 (COMPLEMENTARY SLACKNESS): Let $\boldsymbol{x}$ be feasible for the primal LP and $\boldsymbol{y}$ be feasible for the dual. Then we have that $\boldsymbol{x}$ and $\boldsymbol{y}$ are optimal solution of their respective LPs if and only if complementary slackness holds:

1. $y_i = 0$, or $b_i - \sum_{j=1}^{n} a_{ij} y_i = 0 \qquad \forall i = 1, 2, \ldots, m$, and

2. $x_j = 0$, or $\sum_{i=1}^{m} a_{ij} x_j - c_j = 0 \qquad \forall j = 1, 2, \ldots, n$,

where a_{ij} is the element in the i-th row and j-th column of the matrix A.

2.3.2 *Simplex method*

The idea of the simplex method is to move a basic feasible solution to an adjacent vertex to improve the objective value [41]. If the admissible set is a convex polytope, at least one of the vertices must be an optimal solution, given a linear objective function. A graphical representation of the simplex method is shown in Figure 4 to demonstrate the operations of the simplex method.

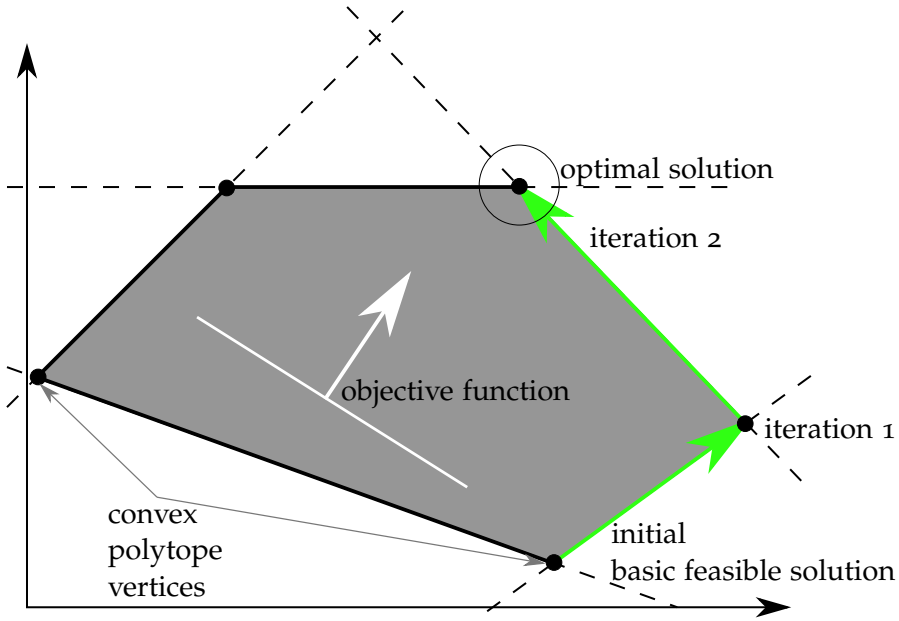


Figure 4: Graphical representation of the simplex method.

Let the linear program be in the standard form, e.g. $\min\{c\boldsymbol{x} : A\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \in \mathbb{R}^n_+\}$. Denote the set of indices for the basic variables as $\mathcal{B}$ and the set of indices for non-basic variables as $\mathcal{N}$. Add slack variables to each inequality to convert the inequality to an equality, e.g. replace (2.6) with (2.7)

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \leq b_1 \tag{2.6}$$

$$\begin{aligned} a_1 x_1 + a_2 x_2 + \cdots + a_n x_n + s &= b_1 \\ s &\geq 0 \end{aligned} \tag{2.7}$$

The linear program is now in the form

$$\min\{\boldsymbol{c}^T \boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \in \mathbb{R}^n_+\}. \tag{2.8}$$

Since the simplex method moves from one basic feasible solution to another, it requires increasing a non-basic variable. We need to track how such a change affects the basic variables. To understand this effect, rewrite $A\boldsymbol{x} = \boldsymbol{b}$ in basic and non-basic components, e.g.

$$\begin{aligned} & A\boldsymbol{x} = A_{\mathcal{B}} \boldsymbol{x}_{\mathcal{B}} + A_{\mathcal{N}} \boldsymbol{x}_{\mathcal{N}} = \boldsymbol{b} \\ \Leftrightarrow & \boldsymbol{x}_{\mathcal{B}} = A_{\mathcal{B}}^{-1} \boldsymbol{b} - A_{\mathcal{B}}^{-1} A_{\mathcal{N}} \boldsymbol{x}_{\mathcal{N}} \end{aligned} \tag{2.9}$$

Rewrite (2.9) as components of the non-basic variables,

$$\boldsymbol{x}_{\mathcal{B}} = A_{\mathcal{B}}^{-1}\boldsymbol{b} - \sum_{j\in\mathcal{N}} A_{\mathcal{B}}^{-1}A_j x_j. \tag{2.10}$$

To see the effect of a non-basic variable, we derive $\boldsymbol{x}_{\mathcal{B}}$ with respect to each non-basic variable,

$$\frac{\partial \boldsymbol{x}_{\mathcal{B}}}{\partial x_j} = -A_{\mathcal{B}}^{-1}A_j, \qquad \forall j \in \mathcal{N}, \tag{2.11}$$

it is clear that an increase of non-basic variable x_j will decrease the basic variables $\boldsymbol{x}_{\mathcal{B}}$ by the vector $A_{\mathcal{B}}^{-1}A_j$. To improve the objective value, the effect on the objective function needs to be determined, let the objective function from (2.8) be

$$z = \boldsymbol{c}^T\boldsymbol{x} = \boldsymbol{c}_{\mathcal{B}}^T\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{c}_{\mathcal{N}}^T\boldsymbol{x}_{\mathcal{N}}. \tag{2.12}$$

Rewrite (2.12) as a function of non-basic variables $\boldsymbol{x}_{\mathcal{N}}$,

$$\begin{aligned} z(\boldsymbol{x}_{\mathcal{N}}) &= \boldsymbol{c}_{\mathcal{B}}^T(A_{\mathcal{B}}^{-1}\boldsymbol{b} - \sum_{j\in\mathcal{N}} A_{\mathcal{B}}^{-1}A_j\boldsymbol{x}_j) + \boldsymbol{c}_{\mathcal{N}}^T\boldsymbol{x}_{\mathcal{N}} \\ \Leftrightarrow z(\boldsymbol{x}_{\mathcal{N}}) &= \boldsymbol{c}_{\mathcal{B}}^T A_{\mathcal{B}}^{-1}\boldsymbol{b} - \sum_{j\in\mathcal{N}} (c_j - \boldsymbol{c}_{\mathcal{B}}^T A_{\mathcal{B}}^{-1}A_j)x_j. \end{aligned} \tag{2.13}$$

To determine the rate at which the objective function will change for a given non-basic variable x_j, we derive the objective function z with respect to each non-basic variable x_j,

$$r_j = \frac{\partial z}{\partial x_j} = c_j - \boldsymbol{c}_{\mathcal{B}}^T A_{\mathcal{B}}^{-1}A_j, \qquad \forall j \in \mathcal{N}, \tag{2.14}$$

this is known as the reduced cost r_j. Since we are minimising, one of the non-basic variables with a negative reduced cost is increased, this variable enters the basis. When all the reduced cost associated with non-basic variables are greater or equal to zero, the current basic feasible solution is optimal since the objective value can no longer improve. The value of the chosen non-basic variable is increased as long as all basic variables stay non-negative. From equation (2.11) it is obvious that an increase in x_j, will decrease $\boldsymbol{x}_{\mathcal{B}}$ at a rate of

$$\boldsymbol{d}_{\boldsymbol{\mathcal{B}}}^j = A_{\mathcal{B}}^{-1}A_j. \tag{2.15}$$

When $d_i^j < 0$, the basic variable x_i will remain non-negative. From equation (2.10) we see if d_i^j is positive, x_i is only non-negative when

$$x_j \leq \frac{(A_{\mathcal{B}}^{-1}\boldsymbol{b})_i}{d_i^j}, \tag{2.16}$$

this suggests that the value of the non-basic variable entering the basis should be

$$x_j = \min\left\{\frac{x_i}{d_i^j} : i \in \mathcal{B},\ d_i^j > 0\right\}. \tag{2.17}$$

Pseudocode for the simplex method is given in algorithm 1, where the input is a standard linear program, and the output may be an optimal solution, an infeasible flag or an unbounded flag. When at least one variable in the basic feasible solution is zero, the solution is degenerate and the next iteration may not improve the objective value. When a basic feasible solution is not degenerate, the objective value strictly improves the next iteration. When the same basic feasible solution occurs more than once, the simplex method will cycle, and fail to terminate. Multiple methods exist to prevent cycling, such as Bland's rule [42] and the criss-cross method [43, 44]. Other methods avoid the appearance of degenerate solutions by adding small positive constant values to the right hand side, so basic feasible solutions are never zero [45].

Algorithm 1 Pseudocode for the Simplex method [41].

Input: A standard linear program.
Output: Maybe an optimal solution.

Initialise with a basic feasible solution $\boldsymbol{x}^0$ with $A_{\mathcal{B}}$ the associated basis.
if no basic feasible solution exists **then**
  return infeasible
end if
while basic feasible solution is not optimal **do**
  for all $j \in N$ **do**
    $r_j \leftarrow c_j - \boldsymbol{c}_{\mathcal{B}}^T A_{\mathcal{B}}^{-1} A_j$
  end for
  if $r_j \geq 0, \forall j \in \mathcal{N}$ **then**
    return current basic feasible solution (optimal).
  else
    for all $j \in \mathcal{N}$ **do**
      if $r_j < 0$ **then**
        $\boldsymbol{d}_{\mathcal{B}}^j \leftarrow \boldsymbol{A}_{\mathcal{B}}^{-1} A_j$
      end if
    end for
  end if
  if $\boldsymbol{d}_{\mathcal{B}}^j \leq \mathbf{0},\ \forall r_j < 0,\ j \in \mathcal{N}$ **then**
    return unbounded
  else
    for all $i \in B$ **do**
      $x_i \leftarrow (A_{\mathcal{B}}^{-1}\boldsymbol{b})_i$
    end for
    $x_j \leftarrow \min\{\frac{x_i}{d_i^j} : d_i^j > 0\}$
    let x_j enter the basis and the corresponding x_i exit the basis.
  end if
end while
return $\boldsymbol{x}$

2.3.3 *Integer linear program*

Integer linear programs are linear programs for which the decision variables are restricted to integer values [46]. A general integer linear program is in the form

$$\max\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \in \mathbb{Z}_+^n\}. \tag{2.18}$$

The search space for an ILP is different from the search space of an LP, and we cannot enjoy the luxury of only using the simplex method. A graphical representation of the difference in search spaces are illustrated with Figure 5. Figure 5a is the relaxed version of Figure 5b, e.g. the integer restrictions are dropped. The relaxed ILP contains the solutions of the original problem (Figure 5b) but all the solutions in the relaxed ILP are not included in the ILP. It is clear that not all the vertices of the convex polytope in Figure 5a are solutions in Figure 5b.

The branch and cut algorithm can be used to solve ILPs, which involves executing the branch and bound (B&B) algorithm and generating cutting planes to tighten the relaxed ILP. Figure 6 is a graphical representation of how cutting planes can help obtain a search space with all components of the extreme points integral. For any linear objective function, the integer linear program shown in Figure 5b can be solved to optimality with the simplex method, when cutting planes in Figure 6 are added.

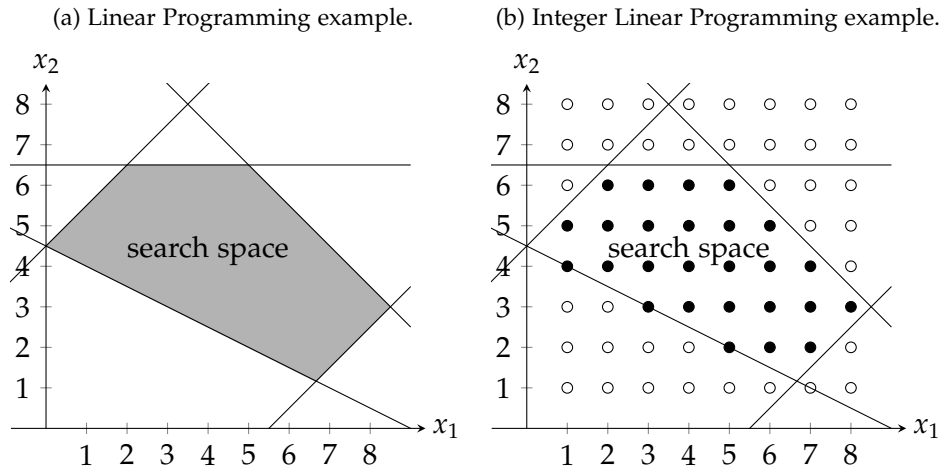


Figure 5: Difference between Linear Programming and Integer Linear Programming search space.

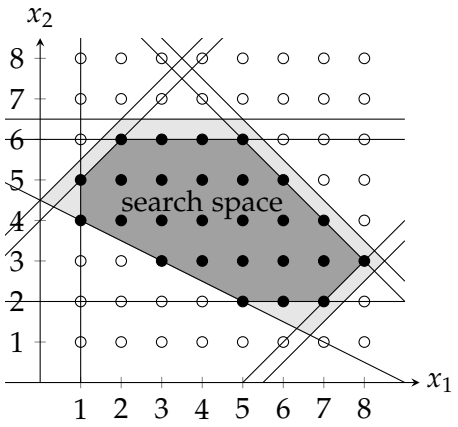


Figure 6: Search space of LP relaxation of Figure 5b with cuts to ensure integral extreme points.

2.3.4 *Cutting planes*

Cutting planes were introduced by Gomory [47] as a method to solve integer- and mixed-integer linear programs. Gomory cuts are very effective when used in combination with the B&B algorithm. Initially, the LP relaxation of the problem is used in conjunction with the simplex method to produce a basic feasible solution to the relaxed problem. When all the components of the basic feasible solution are not integral, the method finds a hyperplane with the basic feasible solution on one side and all the feasible integer solutions on the other side. This is added to a modified linear program, and the process is repeated until an integer solution is found. The simplex method produces a set of equations in the form,

$$x_i + \sum_{j \in \mathcal{N}} a_{ij} x_j = b_i, \forall i \in \mathcal{B}, \tag{2.19}$$

where $\mathcal{B}$, $\mathcal{N}$ are the sets containing the basic variables and non-basic variables respectively. Lets rewrite the equation of any basic variable, x_i from the set of equations (2.19), such that the fractional and integral parts are separate

$$\begin{aligned} & x_i + \sum_{j \in \mathcal{N}} \lfloor a_{ij} \rfloor x_j - \lfloor b_i \rfloor = (b_i - \lfloor b_i \rfloor) - \sum_{j \in \mathcal{N}} (a_{ij} - \lfloor a_{ij} \rfloor) x_j \\ \Leftrightarrow & x_i + \sum_{j \in \mathcal{N}} \lfloor a_{ij} \rfloor x_j - \lfloor b_i \rfloor = b'_i - \sum_{j \in \mathcal{N}} a'_{ij} x_j, \end{aligned} \tag{2.20}$$

where

$$0 \leq b'_i < 1, \tag{2.21}$$

and

$$0 \leq a'_{ij} < 1, \qquad \forall j \in \mathcal{N}. \tag{2.22}$$

To ensure the current non-integral basic feasible solution is excluded, but all the integer feasible solutions remain in the search space, the right-hand side of (2.20) should be less or equal to zero,

$$\begin{aligned} & b'_i - \sum_{j \in \mathcal{N}} a'_{ij} x_j \leq 0 \\ \Rightarrow & s_i - \sum_{j \in \mathcal{N}} a'_{ij} x_j = -b'_i, \forall s_i \in \mathbb{Z}_+, \end{aligned} \tag{2.23}$$

where s_i is the associated slack variable. Constraint (2.23) can be added to the modified linear program, this process can be repeated until an integer-feasible solution is found.

2.3.5 *Branch and bound*

The B&B algorithm systematically enumerates through the feasible search space to obtain the optimal solution(s). An initial LP relation solution forms the root node of a tree structure. The algorithm explores branches of the tree structure and updates the best feasible solution accordingly. Nodes that lead to a lower quality solution than the current best feasible solution are discarded.

The generic B&B algorithm is shown in Algorithm 2, where a minimisation problem is assumed. The input to the algorithm is an ILP problem. When there exists a heuristic solution as the input, the current best value B will get the objective value of the heuristic solution, and the current best solution s becomes the heuristic solution. Let Q be the queue that holds partial solutions to the problem. The function $p(L)$ generates partial solutions and are problem-specific; if no such function is defined, basic feasible solutions from the simplex method can be used. The algorithm keeps executing until the queue Q is empty. At each iteration, while the queue is not empty, an element in the queue, which is a node in the tree structure, should be removed. If the solution that

corresponds to the node n is a feasible solution to the ILP problem and the solution is of better quality, the current best bound B and best solution s are replaced. When the selected node does not result in a better feasible solution, branching is performed and the nodes with the potential to improve the solution quality are added to the queue. When s is unassigned; the algorithm terminates, no solution was found, otherwise the optimal solution is s, with objective value B.

Algorithm 2 Pseudocode for the B&B algorithm [48].

Input: A feasible heuristic solution x_h, Mixed Integer Linear Programming (MILP) problem L.
Output: An Optimal solution or no solution.

1: $s \leftarrow$ no solution
2: **if** heuristic solution x_h exists **then**
3: $B \leftarrow f(x_h)$
4: $s \leftarrow x_h$
5: **else**
6: $B \leftarrow \infty$
7: **end if**
8: $Q \leftarrow p(L)$
9: **while** $Q \neq \emptyset$ **do**
10: $Q \leftarrow Q - \{n\}, n \in Q$
11: **if** x_n is a feasible solution to ILP and $f(x_n) < B$ **then**
12: $B \leftarrow f(x_n)$
13: $s \leftarrow x_n$
14: **else**
15: $N \leftarrow \texttt{branch}(n)$
16: $Q \leftarrow Q \bigcup e, \qquad \forall\ \texttt{bound}(e) \leq B, e \in N$
17: **end if**
18: **end while**
19: **return** s

2.3.6 *Branch and cut*

Most modern MILP solvers use B&B algorithms but are still heavily dependent on problem-specific heuristics to improve computation times. The cutting-plane method on its own can be inefficient for large MILPs since recursively generating cuts results in extremely large LPs, which causes numerical difficulties for a LP solver and often converges slowly.

The B&B algorithm with cut generation, called branch and cut, may efficiently solve large MILPs. Algorithm 3 shows the generic branch and cut algorithm. Initially, the current solution is set to `null` and the corresponding objective value to ∞. A minimisation problem is assumed. The initial problem is added to the list of active LP problems $\mathcal{L}$. At this stage, each LP problem is considered in $\mathcal{L}$. As long as there are active problems, a problem p is selected and the relaxed version of the problem is solved. The function `relax()` returns a problem with integer and binary constraints removed from decision variables. The `solve()` function in line 9 can invoke any arbitrary algorithm that solve LP problems, but the simplex method is usually used. Whenever the LP problem p does not have a feasible solution, another problem is selected from the list of current problems. When the relaxed problem has a feasible solution, $\boldsymbol{x}$ is updated to the current relaxed solution, and v to the current relaxed objective value. If the current relaxed solution is greater than the current best integral feasible solution, there is no need to explore the solution further and a new problem is selected from the list of current problems. When the current relaxed

solution $\boldsymbol{x}$ is integral, current best integral solution $\boldsymbol{x}^*$ and the corresponding objective value v^* is updated, else search for cuts that are violated by $\boldsymbol{x}^*$ and result in an integral solution. The function `add_cut(`p_r`,c)` results in a new problem that adds the cut c to the relaxed problem p_r. The algorithm adds these new problems $p_c \in \mathcal{C}$ to the list of current problems $\mathcal{L}$. From here, the usual branching of the branch and bound algorithm applies. The function `branch()`, partitions the problem into new problems with restricted feasible regions. The algorithm adds partitioned problems to the list of ongoing problems.

Algorithm 3 Pseudocode for the branch and cut algorithm [49].

Input: The initial ILP problem p_0.
Output: Maybe an optimal solution $\boldsymbol{x}^*$ and objective value v^*.

Add initial problem to the list of active problems $\mathcal{L}$
$\boldsymbol{x}^* \leftarrow$ null
$v^* \leftarrow \infty$
$\mathcal{L} \leftarrow \mathcal{L} \bigcup \{p_0\}$
while $\mathcal{L} \neq \emptyset$ **do**
  $\mathcal{L} \leftarrow \mathcal{L} - \{p\}, n \in Q$
  $p_r \leftarrow$ `relax(`p`)`
  repeat
    `solve(`p_r`).`
    if `solution(`p_r`)` $\neq$ feasible **then**
      break repeat
    else
      $\boldsymbol{x} \leftarrow$ `solution(`p_r`)`
      $v \leftarrow$ `objective(`p_r`)`
    end if
    if $v \geq v^*$ **then**
      break repeat
    end if
    if $\boldsymbol{x} \in \mathbb{Z}^n$ **then**
      $\boldsymbol{x}^* \leftarrow \boldsymbol{x}$
      $v^* \leftarrow v$
      break repeat
    end if
    $\mathcal{C} \leftarrow$ `generate_cuts(`$\boldsymbol{x}^*$`)`
    if $\mathcal{C} \neq \emptyset$ **then**
      for all $c \in \mathcal{C}$ **do**
        $p_c \leftarrow$ `add_cut(`p_r`,` c`)`
        $\mathcal{L} \leftarrow \mathcal{L} \bigcup \{p_c\}$
      end for
      continue
    end if
    $\mathcal{B} \leftarrow$ `branch(`p_r`)`
    for all $b \in \mathcal{B}$ **do**
      $\mathcal{L} \leftarrow \mathcal{L} \bigcup \{b\}$
    end for
  until $\mathcal{C} = \emptyset$
end while
return $(\boldsymbol{x}^*, v^*)$

To limit the time a MILP solver spend on solving a problem is to set a optimality gap tolerance [50]. The optimality gap is the difference measure between the current best upper and lower bounds. There are two types of differences, the absolute gap given by,

$$\text{(absolute gap)} = \text{(upper bound)} - \text{(lower bound)} \tag{2.24}$$

and the relative gap given by,

$$\text{(relative gap)} = \frac{\text{(upper bound)} - \text{(lower bound)}}{\text{(upper bound)} + \epsilon} \tag{2.25}$$

where $\epsilon > 0$ and a minimisation problem is assumed.

2.3.7 *Piecewise linear functions*

Not all problems are explicitly linear, but they can still be formulated as a MILP model by approximating separable non-linear functions with piecewise linear functions [51]. Define a function $f(e)$ that approximates the non-linear function $g(x) \in [a, b]$ as

$$f(e) = \begin{cases} g(a) + \frac{e-a}{b-a}(g(b) - g(a)), \text{ if } e \in [a, b] \text{ and } a < b \\ f(a), \text{ if } e = a = b \end{cases} \tag{2.26}$$

which is the linear interpolation between the coordinates $(a, g(a))$ and $(b, g(b))$. The following constraints can be added to the MILP model to approximate $g(x) \in [a, b]$

$$x = \lambda a + \mu b \tag{2.27}$$

$$z = \lambda f(a) + \mu f(b) \tag{2.28}$$

$$\lambda + \mu = 1, \qquad \lambda, \mu \geq 0. \tag{2.29}$$

To get finer granularity of the function, we can divide the domain into disjoint intervals $[a_i, b_i]$. The MILP model should only choose one interval to work with. The binary variables δ_i for all $i \in I$ can be added, where I is an index set for each disjoint interval of $f(x)$. To ensure only one interval is selected,

$$\sum_{i \in I} \delta_i = 1, \tag{2.30}$$

is added with the modified constraints as

$$x = \lambda_i a_i + \mu_i b_i, \ \forall i \in I, \tag{2.31}$$

$$z = \lambda_i f_i(a_i) + \mu_i f_i(b_i), \ \forall i \in I, \tag{2.32}$$

$$\lambda_i + \mu_i = \delta_i, \ \forall i \in I, \tag{2.33}$$

$$\lambda_i, \mu_i \geq 0, \quad \delta_i \in \{0, 1\}, \qquad \forall i \in I. \tag{2.34}$$

2.3.8 *Big M*

Some of the models in this thesis have multiple Big M constraints, which suffice a brief overview of the method. The Big M method refers to MILPs formulations where violations of a constraint are associated with a large positive constant M. The Big M ensures equality of variables, and only

when the associated binary variable takes the value one, else it leaves the variables constrained between $-M$ and M. Consider the following example where ψ is the associated binary decision variable, and x, y are decision variables in $\mathbb{R}_+$,

$$\begin{aligned} x + y &\leq M\psi \\ x + y &\geq -M\psi, \end{aligned} \tag{2.35}$$

when $\psi = 0$, then $x = y$, otherwise,

$$-M \leq x + y \leq M. \tag{2.36}$$

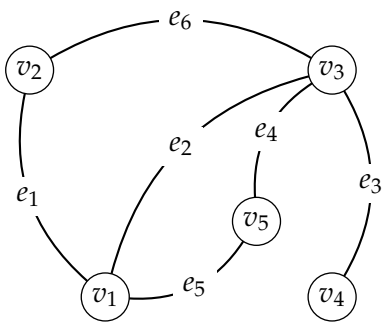
Known problems with the Big M problem in practice is numerical instability when M is too large or if M is too small, resulting in a constraint to become infeasible and making the entire model infeasible. Caution should be taken when choosing the Big M for a given dataset.

2.4 GRAPH THEORY

A transportation network can be modelled as a graph, this allows the use of graph algorithms such as single-source shortest path algorithms [52]. In this thesis some of the problems addressed are modelled as graphs; this suffices as an introduction to graph theory.

A graph includes points called *vertices* or *nodes* and the lines connecting them are called *edges*. A weight is usually associated with each edge and for the shortest path problem it represents the length of an edge. Figure 7a shows an example of a graph with 5 vertices and 6 edges, every edge in the graph connects two vertices for example, e_6 connects v_2 and v_3.

(a) Graph with 5 vertices and 6 edges.



(b) Digraph with 5 vertices and 7 arcs.

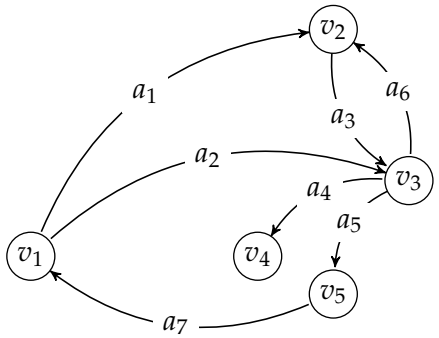


Figure 7: Graph and Digraph.

DEFINITION 3: A **graph** $\mathcal{G}$ is composed of two finite sets, a set of **vertices** $\mathcal{V}$, and a set of connecting lines $\mathcal{E}$ called **edges**, such that each edge connects two vertices. A graph is written as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

Digraphs (Directed graphs)

DEFINITION 4: A **digraph** $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ is a graph where each arc $a = (v, w) \in \mathcal{A}$ has a direction from vertex v to w with $v \in V$ and $w \in \mathcal{V}$. **Edges** are called **arcs** when using **digraphs**.

Figure 7b shows an example of a digraph with 5 vertices and 7 arcs, every arc in the graph connects two vertices in a directed fashion for example, a_7 connects v_5 and v_1.

Tree

DEFINITION 5: A **tree** is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where any two vertices are connected by exactly one path.

Figure 8 shows an example of a tree graph. There exists only one path between any two vertices, e.g. the path between v_1 and v_2 is the sequence of edges e_4, e_2, e_3.

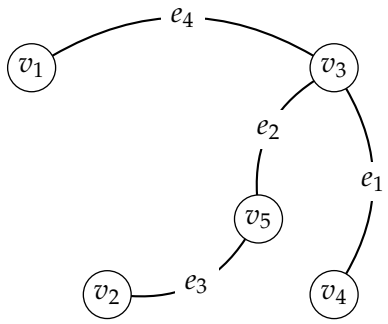


Figure 8: Example of a tree graph.

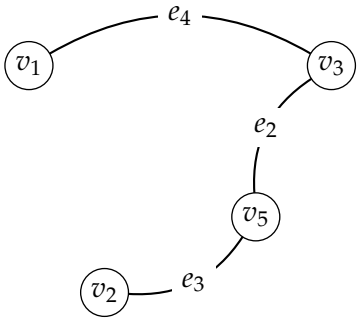
Path graphs

DEFINITION 6: The **degree** of a vertex in a graph is the number of edges that are incident to the vertex.

DEFINITION 7: The **path graph** $\mathcal{P}_n$ is a tree with two nodes of vertex degree one, and the other $n - 2$ nodes of vertex degree two.

Figure 9a shows an example of a path graph that corresponds with the path between v_1 and v_2 of Figure 8. A path graph is a graph that can be drawn so that all of its vertices and edges lie on a single straight line [53]. Figure 9b is an example where the path graph in Figure 9a is drawn in a straight line.

(a) Example of a path graph.



(b) Vertices rearranged in a line.

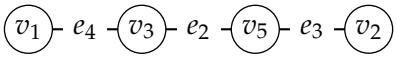


Figure 9: Path graph (left) rearranged into a horizontal line (right).

Directed path (dipath)

DEFINITION 8: A **directed** path is a finite sequence of edges directed in the same direction which joins a sequence of vertices, where all vertices and edges are distinct.

Disjoint graphs

DEFINITION 9: Two subgraphs are edge **disjoint** if they share no edges, and vertex disjointed if they share no vertices.

2.4.1 *Shortest path algorithms*

One of the most common shortest path algorithm used is Dijkstra's Algorithm [25]. Using abstract data types, such as minimum priority queues, can lead to improved computation time [54–56]. A minimum priority queue is an abstract data-type that consists of 3 basic operations: get minimum value from the queue, add to queue with priority, and decrease the priority of an element in the queue [57]. These operations are added to Dijkstra algorithm in Algorithm 4 with Q, the priority queue.

The input for Algorithm 4 is a finite weighted graph, a source node s and a target node t. The output is the minimum distance, and the predecessor node of each explored node. Once a node is considered by the algorithm as in line 11, the minimum distance in the set D is updated. The variable d_v presents the distance to the node v from the source node. The variable p_v represent the predecessor vertex of node v from the source node s.

Algorithm 4 Dijkstra's algorithm with min-priority queues.

Input: Finite weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, source node s, target node t
Output: D a set of minimum distances, P a set of predecessor nodes

1: $d_s \leftarrow 0$
2: $Q \leftarrow \emptyset$
3: **for each** $v \in V$ **do**
4: **if** $v \neq s$ **then**
5: $d_v \leftarrow \infty$
6: $p_v \leftarrow \varnothing$
7: **end if**
8: Add v to Q with priority d_v
9: **end for**
10: **while** Q is not empty **do**
11: $u \leftarrow$ extract minimum from Q
12: **if** $u = t$ **then**
13: **break**
14: **end if**
15: **for each** adjacent vertex $v \in Q$ of u **do**
16: $a \leftarrow d_u + w(e = (u, v))$
17: **if** $a < d_v$ **then**
18: $d_v \leftarrow a$
19: $p_v \leftarrow u$
20: Decrease priority of v in Q to a
21: **end if**
22: **end for**
23: **end while**
24: $D \leftarrow D \bigcup d_v \qquad \forall v \in V$
25: $P \leftarrow P \bigcup p_v \qquad \forall v \in V$

The source node s gets the distance to itself as zero, which line 1 describes. Initially, the queue of vertices the algorithm needs to explore is empty. The algorithm adds each vertex in V to the

queue Q. The priority of each vertex in the queue is initially ∞, except the source node s which is first in the queue. The initialisation of the queue is done in lines 3 - 9.

As long as the queue Q is not empty, the first element with minimum priority is removed and explored. The priority of the adjacent vertices of the selected element in the priority queue is updated, and the process repeats until the queue is empty or the algorithm reaches the target vertex.

2.5 MATHEMATICAL NOTATION

This section describes the notation used for the models in the thesis. The modelling depends on set theory for the indices of variables and set filtering functions to exclude unwanted indices. Summations over a set expand the columns, and $\forall$ groups some rows with the indices from the same set, thus still maintaining a matrix form. Consider the set $\mathcal{M} = \{1, 2, \ldots, M\}$, with variables $x_m \in \mathcal{M}$, and constants $c_m \in \mathcal{M}$, the objective function can be written as

$$\begin{aligned} z &= \sum_{m \in \mathcal{M}} x_m c_m \\ &\Leftrightarrow z = \boldsymbol{c}^T \boldsymbol{x}. \end{aligned} \tag{2.37}$$

An example of a constraint set can also be described as

$$\sum_{m \in \mathcal{M} \setminus g(p_1, p_2, \ldots, p_K)} x_m c_m = y_t + b_t, \forall t \in \mathcal{T} \setminus f(p_1, p_2, \ldots, p_K), \tag{2.38}$$

where g, f are set filtering functions that takes arbitrary parameters $p_1, \ldots, p_K$ and returns a subset of $\mathcal{M}$ and $\mathcal{T}$ respectively. $\mathcal{T} = \{1, 2, \ldots, T\}$ is an index set, and y_t are decision variables with indices in $\mathcal{T}$ and b_t constants with indices in $\mathcal{T}$. When f and g returns in empty sets, constraint set (2.38) generate equations

$$\begin{aligned} c_1 x_1 + c_2 x_2 + \cdots + c_M x_M &= y_1 + b_1 \\ c_1 x_1 + c_2 x_2 + \cdots + c_M x_M &= y_2 + b_2 \\ &\vdots \\ c_1 x_1 + c_2 x_2 + \cdots + c_M x_M &= y_N + b_N. \end{aligned} \tag{2.39}$$

Constraint set (2.39) can be written in the form

$$A\boldsymbol{u} = \boldsymbol{b}, \tag{2.40}$$

where

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_M \\ y_1 \\ \vdots \\ y_N \end{bmatrix}. \tag{2.41}$$

Throughout the thesis, slack and surplus variables are not shown in the models and the models do not need to be in standard LP form.

2.5.1 *Route Planning for a Battery Electric Vehicle (BEV)*

For completeness a recent optimisation model proposed by [28] is provided. Many fundamental modelling concepts discussed in this chapter was used by [28]. The authors made many assumptions to model the single BEV route planning problem, such as a single type of BEV with constant range and fixed travel and energy consumption between nodes. The charging time is proportionally linear to the desired quantity to recharge for an inverse recharging rate for each node.

The problem [28] formulated is given by
Maximise

$$\sum_{j \in \mathcal{N}_s} w_j y_j \tag{2.42}$$

Minimise

$$\sum_{(i,j) \in \mathcal{A}} k(Q - q_j - \frac{e_{i,j}}{2}) \tau_{i,j} x_{i,j} \tag{2.43}$$

subject to

$$\sum_{(i,j) \in \mathcal{A}} x_{i,j} \geq y_j \quad \forall j \in \mathcal{N} \tag{2.44}$$

$$\sum_{(i,j) \in \mathcal{A}} x_{i,j} \leq 1 \quad \forall j \in \mathcal{N} \setminus \{o, d\} \tag{2.45}$$

$$\sum_{(i,j) \in \mathcal{A}} x_{i,j} - \sum_{(j,i) \in \mathcal{A}} x_{i,j} = \begin{cases} -1 & \text{if} \quad j = o, \\ 1 & \text{if} \quad j = d, \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in \mathcal{N} \tag{2.46}$$

$$q_j \leq (\sum_{(i,j) \in \mathcal{A}} x_{i,j}) Q, \quad \forall j \in \mathcal{N} \tag{2.47}$$

$$t_j \leq (\sum_{(i,j) \in \mathcal{A}} x_{i,j}) T, \quad \forall j \in \mathcal{N} \tag{2.48}$$

$$v_j \leq h_j y_j, \quad \forall j \in \mathcal{N} \tag{2.49}$$

$$l_j \leq t_j \leq u_j, \quad \forall j \in \mathcal{N} \tag{2.50}$$

$$(q_i - e_{i,j} x_{i,j}) + r_i - (1 - x_{i,j}) Q \leq q_j \leq (q_i - e_{i,j} x_{i,j}) + r_i + (1 - x_{i,j}) Q, \quad \forall (i,j) \in \mathcal{A} \tag{2.51}$$

$$r_i \leq Q - q_i, \quad \forall i \in \mathcal{N}_c \tag{2.52}$$

$$r_i \leq Qv_i, \quad \forall i \in \mathcal{N}_c \tag{2.53}$$

$$(t_i + \tau_{i,j}x_{i,j} - (1 - x_{i,j})T \leq t_j \leq (t_i + \tau_{i,j}x_{i,j}) + (1 - x_{i,j})T, \quad \forall i \in \mathcal{N}_s, (i,j) \in \mathcal{A} \tag{2.54}$$

$$(t_i + \tau_{i,j}x_{i,j} - (1 - x_{i,j})T \leq t_j \leq (t_i + \tau_{i,j}x_{i,j}) + (1 - x_{i,j})T, \quad \forall i \in \mathcal{N}_c, (i,j) \in \mathcal{A} \tag{2.55}$$

$$c_i = g_i r_i, \quad \forall i \in \mathcal{N}_c \tag{2.56}$$

$$t_j, c_j, q_j, r_j \geq 0, \quad t_j, c_j, q_j, r_j \in \mathbb{R}, \quad \forall j \in \mathcal{N} \tag{2.57}$$

$$y_j, v_j, x_{i,j} \in \{0, 1\}, \quad \forall j \in \mathcal{N}, (i,j) \in \mathcal{A} \tag{2.58}$$

Where $\mathcal{N}_s$ is a set of non-charging station nodes and $\mathcal{N}_c$ is a set of charging nodes. $\mathcal{N}$ is a set of all nodes, which includes $\mathcal{N}_s, \mathcal{N}_c$ the origin o and the destination d. The set of arcs between nodes in $\mathcal{N}$ is denoted by $\mathcal{A}$. w_j is a weight score for visited node j and s_j is the fixed duration of stay at non-charging node j. For charging nodes, the amount of energy recharged on node j is given by r_j, and the rate of recharging is given by g_j. h_j is one if node j is a designated battery swap station, and zero otherwise. $e_{i,j}$ is the energy consumed travelling from node i to node j—the energy level when arriving at node j is q_j. The duration of stay at a charging station j is denoted c_j. The arrival time at node j is t_j with upper and lower bounds of u_j and l_j respectively. y_i is a binary decision variable to determine if node i was visited. v_i is a binary decision variable to determine if a vehicle recharges on node i and $x_{i,j}$ is a binary decision variable that indicated if the vehicle travels on an arc $(i,j) \in \mathcal{A}$.

Expressions (2.5.1) and (2.5.1) represents bi-objective functions the first to maximise the total sum of none charging stations and the second incorporates a range anxiety cost function. Constraints 2.5.1 and 2.5.1 handle the connectivity of nodes visited, whenever a node i is visited there must be atleast one incident arc $(i,j) \in \mathcal{A}$ enabled. For a path followed not to be disjoint the flow conservation constraints (2.5.1) are added. Constraint sets (2.5.1) and (2.5.1) guarantee that time and energy requirements are satisfied at each node. Constraint set (2.5.1) allows arrival time-windows to be specified per node. Constraints (2.5.1) and (2.5.1) ensures that the start time of next node accounts for the start time of the previous node and the duration spent travelling between the nodes. Constraint set (2.5.1) added to link the energy when travelling from one node to another; it also accounts for recharging related in constraints (2.5.1) and (2.5.1).

In the next chapter, many of the assumptions made by [28] are addressed; this includes variable velocity, duration and energy usage on arcs. The linear fashion in which charging time is handled

is also addressed with the ability for a BEV to include a solar panel where the energy obtained heavily depends on weather conditions.

2.5.2 *Summary*

This chapter gave an overview of optimisation algorithms and modelling methods that are applicable to the thesis. Most of the theory is on MILP since the models described in the thesis are MILP models. An introduction to graph theory was also provided since the input data is modelled as graphs.

In the next chapter, the general path-based and flow conservation models are described with variable durations and energy usages on arcs. Some preliminary results are also shown for these models.

3

BATTERY ELECTRIC VEHICLE ROUTE PLANNING APPROACHES

This chapter introduces multiple Mixed Integer Linear Programming (MILP) models to solve single-vehicle route planning for Battery Electric Vehicles (BEVs). The scope of this chapter is to introduce models that can provide an optimised path between two locations for a BEV with a solar panel, given any arbitrary linear objective function and vehicle parameters. Examples of objective functions include the shortest distance, the fastest travel time, most energy-efficient route or the route with the least cost.

Throughout the literature discussed in section 1.1.3, most approaches to this problem made assumptions that drastically simplifies the problem which may lead to some undesirable results [10,11,14,15,17,20,21,27,28]. Most of the existing models are time-independent where energy consumption relates only to vehicle and road characteristics [10,11,13–15,58]. Time-dependent models may incorporate features such as weather conditions. Figure 10 depicts a crude oversimplification of the dynamic nature of the problem, where the speed travelled on a segment depends on the weather. The vehicle starts on $t = 0$ at v_1 and ends at v_2 on different times depending on the velocity. In the top depiction, the vehicle has the slowest speed, and the bottom depiction has the fastest speed. The weather conditions are different at different locations and times. In some cases, it may be beneficial to drive at a greater speed to avoid getting caught under clouds; this is illustrated at the bottom of Figure 10. As the vehicle speed on a segment decreases, the duration the vehicle spends on the segment increases. The longer a vehicle spent on a segment, the more the sun angle changes.

Another assumption made with most existing algorithms is that the energy efficiency for the vehicle is constant [28]. Throughout the literature aspects of the problem are discretised and sequential approaches are followed which do not account for the entire search space. Generally, the models in the literature do not account for acceleration, which is an important aspect, especially when there are other vehicles on the road. Usually narrowly scoped optimised algorithms do not allow extensions on constraints or features trivially. At the time of writing, no existing single vehicle route planning approaches enabled the inclusion of a solar panel for a BEV to recuperate energy on-the-go [28].

A feature that allows charging on-the-go seems nonsensical, but if the cost to produce solar-panels decreases, such an implementation may become feasible by slightly extending driving ranges. Some models in the literature accounted for energy recuperated through regenerative braking. However, it is not trivial to expand these algorithms to account for solar-energy since solar-energy heavily depends on the time of day and cloud coverage. The problem becomes challenging when a holistic approach is taken since many factors need to be taken into account. Decisions made in the past significantly impact future decisions.

When considering solar energy, it is essential to use weather predictions where the BEV is supposed to travel at a specified time. Weather forecasts are not always accurate, mainly when predicting for the distant future. Thus the route planning is divided into, short-term, medium-term and long-term planning. Short-term planning takes fine granularity (five-minute intervals) weather data as an input, but only for a short period in the future, e.g. two hours; this gives more reliable results but fails to account for change over more extended periods. Medium-term planning

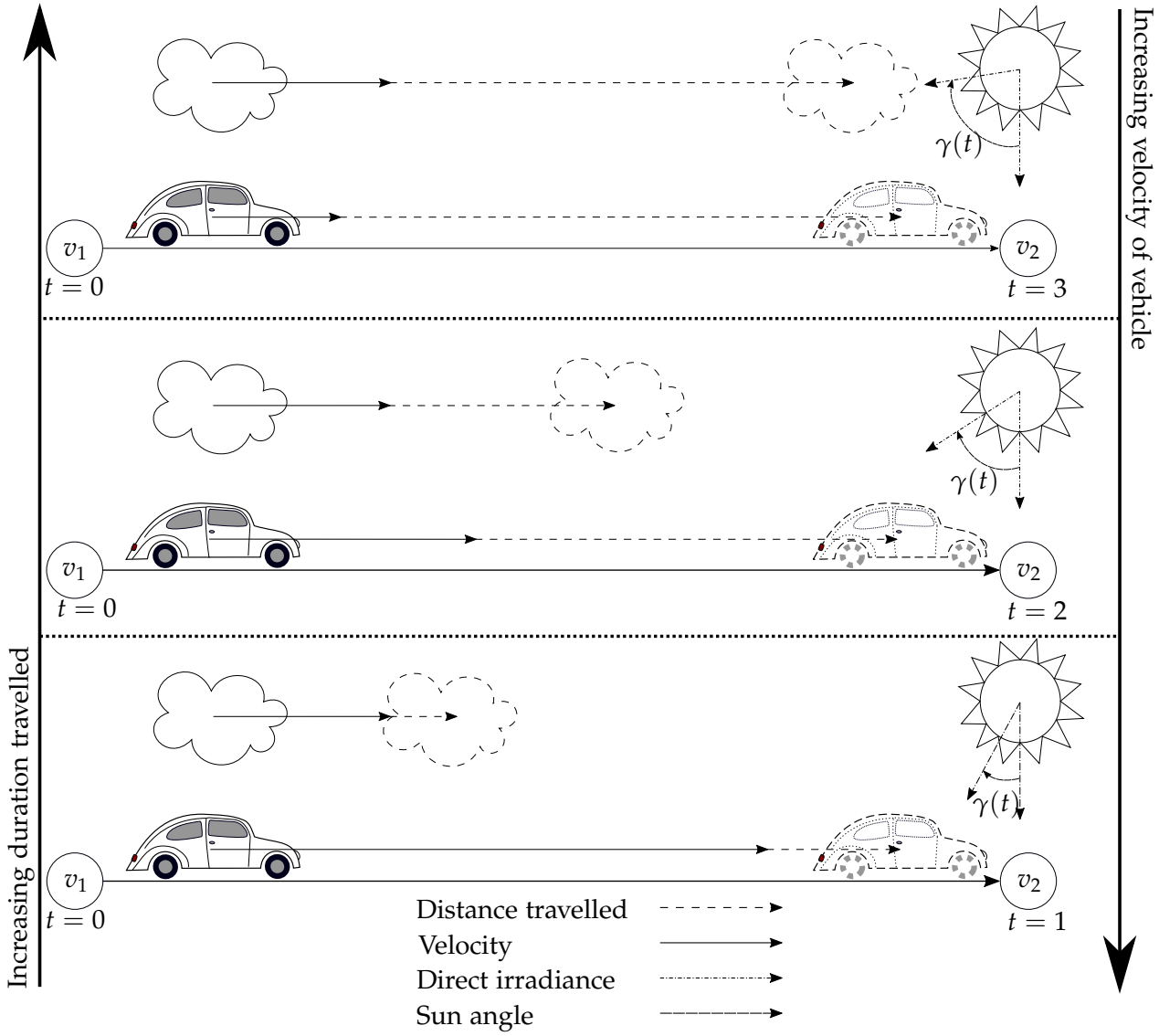


Figure 10: Representation of how different velocities can affect energy gained from the sun.

uses one-hour intervals over twenty-four-hours as an input. Long-term planning includes multiple days, with three-hour intervals.

In Section 3.2, the initial model is a shortest path formulation that does not account for energy usage, or the parameters of the BEVs. The model is then gradually extended to include all the required capabilities. Finally the models and objective functions are adapted to be generic for any vehicle by using linear piecewise approximations for the energy-usage and duration graphs.

3.1 DESCRIBING A ROUTE WITH GEOGRAPHIC INFORMATION SYSTEM (GIS) DATA

The problem needs to be broken down into the smallest possible parts to formulate a mathematical model. For this purpose a route is divided into multiple parts, called segments and each segment is approximated as a linear line, as shown in Figure 11. Associated with each segment are properties, such as the slope θ, length d and friction. Weather conditions, such as wind direction, -speed and cloud coverage which change over time are considered at a later stage.

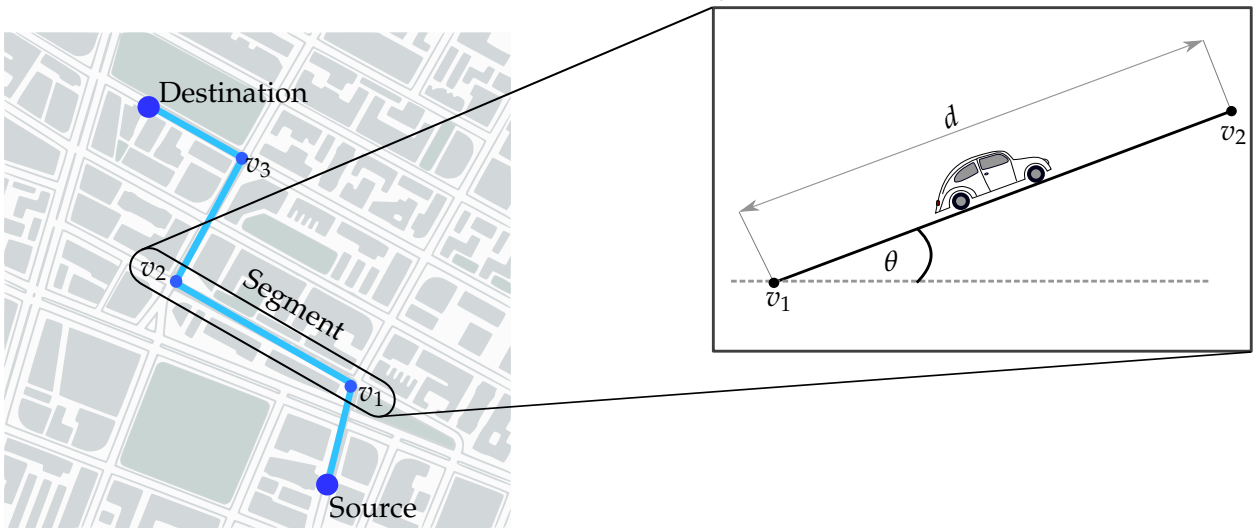
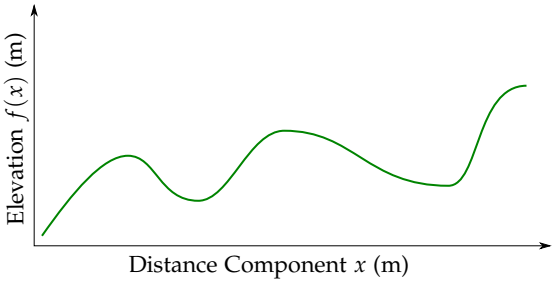


Figure 11: Segment between v_1 and v_2, with slope θ and length d.

The elevation of the route is a three-dimensional surface constrained to the drivable areas [59,60]. Solving for such a representation is not trivial, so the problem needs to be modelled as a graph and use the distance components along the route, Figure 11 illustrates such an idea. Figure 12a shows a non-linear function $f(x)$ representing the elevation with the distance component as a parameter which demonstrate the weights of an infinite path graph, where each arc corresponds to a slope $d\theta$. Since it is not possible to work with an infinite path graph, the problem needs to be discretised. A simple way to approximate a continuous path is to use the extreme points of the elevation data as vertices and estimate the slope between extreme points as segments. An example of such an approach is shown in Figure 13a, where $\theta_1, \ldots, \theta_5$ are the slopes that corresponds with the segments $s_1, \ldots, s_5$. The distance component in Figure 13a is not the distance of the segment, only a component of the segment length. Figure 13b shows a path with the lowest possible granularity, e.g. a single segment and slope. The granularity can be increased to improve the overall accuracy of the data Figure 12b shows a linear piecewise function that approximates the non-linear function more accurately at the expense of more data-points. Given the non-linearity of actual road-data, the vertex positions and segment lengths need to be chosen with care.

(a) Non-linear function $f(x)$ representing the elevation with the distance component as a parameter.



(b) Linearised version of the function $f(x)$ in Figure 12a, but with finer granularity than shown in Figure 13a.

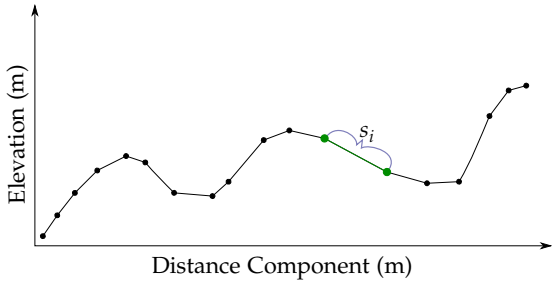
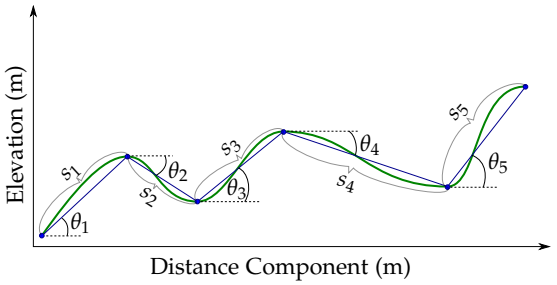


Figure 12: Difference between non-linear and linear piecewise functions.

(a) Linear piecewise function that uses the extreme points of elevation function for the path a BEV can follow.



(b) Linearised version of the function $f(x)$ in Figure 12a, but with coarser granularity than shown in Figure 13a.

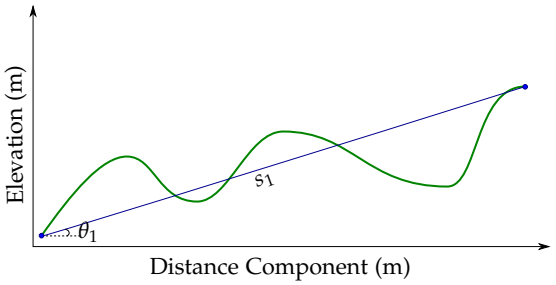


Figure 13: Side-by-side comparison showing the problem of not using enough vertices to linearise a non-linear function (Figure 13b).

Let each segment s be an ordered pair of vertices $(\boldsymbol{a}, \boldsymbol{b})$ and the path $\mathcal{P}$ a set of segments; we present this as a path graph in Euclidean space as,

$$\mathcal{P} = (\mathcal{V}, \mathcal{S}), \tag{3.1}$$

where $\mathcal{V}$ is the set that contains all the vertices in the graph, and S the set with all arcs (linear segments).

COROLLARY 1: Each path graph between vertices $\boldsymbol{a}$ and $\boldsymbol{b}$ must be a directed acyclic path graph, with no disjointed sub-graphs.

THEOREM 3: At most, one path graph in Euclidean space can contain a single linear segment, which starts at $\boldsymbol{a}$ and ends at $\boldsymbol{b}$ with the segment weight as the distance between $\boldsymbol{a}$ and $\boldsymbol{b}$.

Proof. Any line between vertices $\boldsymbol{a}$ and $\boldsymbol{b}$ always results in the same Euclidean distance,

$$\mathrm{d}(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}, \tag{3.2}$$

with $\boldsymbol{a} = (a_1, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, \ldots, b_n)$ in Euclidean n-space. Thus, any path containing only $\boldsymbol{a}$ and $\boldsymbol{b}$ are the same paths and indicates that there cannot be more than one arc directly connecting $\boldsymbol{a}$ and $\boldsymbol{b}$. When considering different routes with the same starting node and destination node, a collection of directed path graphs can be used.

Theorem 3 can be generalised to any directed path graph in Euclidean n-space.

THEOREM 4: Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ with vertices in Euclidean n-space, each arc $a \in \mathcal{A}$ containing the same source node $\boldsymbol{s}$ and target node $\boldsymbol{t}$, such that $a = (\boldsymbol{s}, \boldsymbol{t})$, is the same arc in Euclidean n-space, where the weight of the arc is the distance between $\boldsymbol{s}$ and $\boldsymbol{t}$. For any two directed path graphs, $\mathcal{P}_i = (\mathcal{V}_i, \mathcal{A}_i)$ and $\mathcal{P}_j = (\mathcal{V}_j, \mathcal{A}_j)$ that are sub-graphs of $\mathcal{G}$ and between $\boldsymbol{s}$ and $\boldsymbol{t}$, the following properties apply:

$$2|\mathcal{A}_i \bigcup \mathcal{A}_j| = |\mathcal{A}_j| + |\mathcal{A}_i|, \qquad \text{if} \quad \mathcal{P}_i = \mathcal{P}_j, \text{ otherwise,} \tag{3.3}$$

$$|\mathcal{A}_i \bigcup \mathcal{A}_j| < |\mathcal{A}_j| + |\mathcal{A}_i|, \qquad \text{if} \quad |\mathcal{A}_i \bigcap \mathcal{A}_j| > 0, \tag{3.4}$$

$$|\mathcal{A}_i \bigcup \mathcal{A}_j| = |\mathcal{A}_j| + |\mathcal{A}_i|, \qquad \text{if} \quad |\mathcal{A}_i \bigcap \mathcal{A}_j| = 0. \tag{3.5}$$

Proof. Given that arcs are distinct for an ordered vertex pair for directed path graphs $\mathcal{P}_i$ and $\mathcal{P}_j$ between any vertex $\boldsymbol{s}$ and $\boldsymbol{t}$; there can exist only one combination of valid directed path graphs for

any unordered set of arcs of any path graph. Consider the case where one path graph contains all arcs of another path graph with the same start and target node. To include different arcs in $\mathcal{P}_j$ than that of the path graph $\mathcal{P}_i$ causes cycles and not containing any more arcs yields the same path graph. The union of the same set still produces the original set, and thus, for any arbitrary path graphs,

$$2|\mathcal{A}_i \bigcup \mathcal{A}_j| = |\mathcal{A}_j| + |\mathcal{A}_i|.$$

Consider the case where $\mathcal{P}_i$ and $\mathcal{P}_j$ have no arcs in common; this is valid when there exists more than one path graph between $\boldsymbol{s}$ and $\boldsymbol{t}$. The property of the disjointed sets is that,

$$|\mathcal{A}_i \bigcup \mathcal{A}_j| = |\mathcal{A}_j| + |\mathcal{A}_i|.$$

Finally, in the case where some of the arcs are common between $\mathcal{P}_i$ and $\mathcal{P}_j$, and they are not disjointed, nor does one directed path graph contain the other completely. Since we cannot completely contain a directed path graph in another, we need to exclude at least one arc from $\mathcal{P}_i$. When an arc is excluded from a directed path graph between $\boldsymbol{s}$ and $\boldsymbol{t}$, the directed-path is invalidated and changes to the source or target node were made. Thus, we need to add another arc that cannot include the same nodes from the removed arc because it will result in the same directed-path graph. The only way to have a different path graph is to include another vertex from $\mathcal{V}$, assuming there exists one, and add two arcs to form a valid directed-path graph. The difference between the sets of arcs will always be larger than zero for this case, from set theory, we know that for such a case,

$$|\mathcal{A}_i \bigcup \mathcal{A}_j| < |\mathcal{A}_j| + |\mathcal{A}_i|.$$

It may seem that Theorem 3 and Theorem 4 are trivial, but it plays an important role in validating generated path graphs in the preprocessing step. In the next section, the conceptual models are formulated.

3.2 CONCEPTUAL MODELS

For this thesis, models need to accommodate multiple objective functions [1] when performing route planning for a BEV. In this section, basic models are formulated and extended by considering various objective functions and factors that may influence decisions.

The shortest path formulations have been studied extensively and will serve as a starting point for the formulations in this thesis. There are two well know MILP formulations for the shortest path problem, the path-based and the flow conservation model. The path-based formulation selects a single path ζ_ρ, given a set of finite paths $\mathcal{P}$, where the objective is to
minimise

$$\sum_{\rho \in \mathcal{P}} \zeta_\rho d_\rho, \tag{3.6}$$

subject to

$$\sum_{\rho \in \mathcal{P}} \zeta_\rho = 1, \tag{3.7}$$

$$\zeta_\rho \in \{0,1\},\ \forall \rho \in \mathcal{P}, \tag{3.8}$$

1 Only one objective function at a time, not multi-objective optimisation.

where d_ρ is the total distance of the path, which is the sum of all arc weights included in the path ρ. With a complete graph $G = (\mathcal{V}, \mathcal{A})$, the number of paths between a starting vertex s and a target vertex t is given by

$$\sum_{m=0}^{|\mathcal{V}|-2} \binom{|\mathcal{V}|-2}{m} m! \tag{3.9}$$

where m is a subset of the remaining $|\mathcal{V}|-2$ vertices. The obvious disadvantage of this model is the number of path variables. Another disadvantage is that a separate algorithm is required to generate all possible paths. The advantage of this model is that the number of paths can be limited to create a heuristic algorithm.

The alternative formulation is the flow conservation model, which does not need a path variable for each path. The flow conservation model takes the weight of each arc into account, so the objective is to
minimise

$$\sum_{a \in \mathcal{A}} f_a d_a, \tag{3.10}$$

subject to

$$\sum_{(i,j)\in\sigma(i)} f_{ij} - \sum_{(j,i)\in\delta(i)} f_{ji} = \begin{cases} 1 & i = s \\ -1 & i = d \\ 0 & \text{otherwise} \end{cases}, \quad \forall i \in \mathcal{V}, \tag{3.11}$$

where $\sigma(i)$ is a function that returns a set of all the adjacent arcs to i and $\delta(i)$ is a function that returns a set of all the arcs that i is adjacent to and

$$f_a \geq 0, \ \forall a \in \mathcal{A}. \tag{3.12}$$

The flow conservation model in (3.10) - (3.12) is a compact formulation of shortest-path problem. The number of flow variables are equal to the number of arcs $|\mathcal{A}|$, in the graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$. The weight on each arc corresponds to the distance d_a travelled on the arc, with $d_a \geq 0$. When each segment's travel time is known in advance, the distance of each segment can be replaced with a fixed time t_s. Similarly, for the path-based model d_ρ can be replaced with t_ρ.

Next, models are extended such that the time is not fixed and depends on the velocity travelled on each segment.

3.2.1 *Path formulation with linearised velocities and durations*

The velocity the BEV travels on segment s is a decision variable v_s. Since the relationship between travel time and velocity is not linear, the duration that a vehicle will travel for a distance d_s with velocity v_s needs to be linearised. A new variable t_s is introduced which is the start time on segment s and depends on all the previous start times and durations for path ρ. Let $\mathcal{S}(\rho)$ be the set of all segments associated with path ρ. For the variable time formulation, the objective is to
minimise

$$\sum_{\rho \in \mathcal{P}} (t_{|\mathcal{S}(\rho)|-1} + \tau_{|\mathcal{S}(\rho)|-1}) \zeta_\rho, \tag{3.13}$$

where $\tau_{|\mathcal{S}(\rho)|-1}$ is the duration on the last segment of path ρ, $t_{|\mathcal{S}(\rho)|-1}$ is the start time on the last segment of ρ and ζ_ρ is the binary decision variable to determine whether a path is selected or not.

The objective function (3.13) is not linear and needs to be linearised. For this purpose introduce continuous variables

$$z_\rho = \zeta_\rho(\tau_{|\mathcal{S}(\rho)|-1} + t_{|\mathcal{S}(\rho)|-1}),\ \forall \rho \in \mathcal{P}, \tag{3.14}$$

z_ρ can be used in the linearisation constraints. To ensure z_ρ is zero when ζ_ρ is zero, Big M constraints are added as

$$z_\rho \leq M\zeta_\rho,\ \forall \rho \in \mathcal{P}. \tag{3.15}$$

When ζ_ρ is one,

$$z_\rho = t_{|\mathcal{S}(\rho)|-1} + \tau_{|\mathcal{S}(\rho)|-1},\ \forall \rho \in \mathcal{P}. \tag{3.16}$$

Constraint set (3.16) is split into

$$z_\rho \leq t_{|\mathcal{S}(\rho)|-1} + \tau_{|\mathcal{S}(\rho)|-1},\ \forall \rho \in \mathcal{P} \tag{3.17}$$

and

$$z_\rho \geq t_{|\mathcal{S}(\rho)|-1} + \tau_{|\mathcal{S}(\rho)|-1}\ \forall \rho \in \mathcal{P}. \tag{3.18}$$

To ensure feasibility when ζ_ρ is not selected, constraint set (3.18) is modified to a Big M constraint, such that

$$z_\rho \geq t_{|\mathcal{S}(\rho)|-1} + \tau_{|\mathcal{S}(\rho)|-1} - (1-\zeta_\rho)M\ \forall \rho \in \mathcal{P}. \tag{3.19}$$

In the objective function z_ρ is used such that the objective is to
minimise

$$\sum_{\rho \in \mathcal{P}} z_\rho \tag{3.20}$$

subject to

$$\tau_{s-1} + t_{s-1} = t_s,\ \forall s \in \mathcal{S}(\rho)\backslash\{0\},\ \rho \in \mathcal{P}, \tag{3.21}$$

which warrants that each start time depends on all the previous start times and durations. At most one path should be selected, thus

$$\sum_{\rho \in \mathcal{P}} \zeta_\rho = 1. \tag{3.22}$$

τ_s depends on v_s, which should also be linearised. By adding the linearisation constraints,

$$v_s = \sum_{x \in \mathcal{X}} v_{xs}\alpha_{xs},\ \forall s \in \mathcal{S}(\rho),\ \forall \rho \in \mathcal{P}, \tag{3.23}$$

$$\tau_s = \sum_{x \in \mathcal{X}} \tau_{xs}\alpha_{xs},\ \forall s \in \mathcal{S}(\rho),\ \forall \rho \in \mathcal{P}, \tag{3.24}$$

where v_{xs} and τ_{xs} are breakpoints for all $x \in \mathcal{X}$. α_{xs} is a decision variable used to form a convex combination of related breakpoints that satisfies,

$$\sum_{x \in \mathcal{X}} \alpha_{xs} = 1,\ \forall s \in \mathcal{S}(\rho),\ \forall \rho \in \mathcal{P}, \tag{3.25}$$

The decision variable α_{xs} is allowed to take a value greater than zero only when the binary decision variable h_{xs} associated with α_{xs} or the neighbouring $h_{(x-1)s}$ is non-zero; this allows a convex combination of any two adjacent breakpoints as given by,

$$\alpha_{xs} \leq h_{(x-1)s} + h_{xs},\ \forall x \in \mathcal{X} \setminus \{0\},\ \forall s \in \mathcal{S}(\rho),\ \forall \rho \in \mathcal{P}, \tag{3.26}$$

where $\mathcal{X} = \{0, \ldots, X\}$, with 0 the first breakpoint index and X the last breakpoint index. To ensure that there are only two values of α_{xs} which can be greater than zero for each segment $s \in \mathcal{S}$,

$$\sum_{x \in \mathcal{X} \setminus \{X\}} h_{xs} = 1, \ \forall s \in \mathcal{S}(\rho), \ \forall \rho \in \mathcal{P}, \tag{3.27}$$

where the domain of the decision variables are

$$0 \leq \alpha_{xs} \leq 1, \ \forall x \in \mathcal{X}, \ \forall s \in \mathcal{S}(\rho), \ \forall \rho \in \mathcal{P}, \tag{3.28}$$

$$\zeta_p \in \{0, 1\}, \ \forall \rho \in \mathcal{P}, \tag{3.29}$$

$$h_{xs} \in \{0, 1\}, \ \forall x \in \mathcal{X}, \ \forall s \in \mathcal{S}(\rho), \ \forall \rho \in \mathcal{P}, \tag{3.30}$$

$$t_s, v_s, \tau_s \geq 0, \ \forall s \in \mathcal{S}(\rho), \ \forall \rho \in \mathcal{P}, \tag{3.31}$$

$$z_p \geq 0, \ \forall \rho \in \mathcal{P}. \tag{3.32}$$

3.2.2 *Flow conservation with linearised velocities and durations*

The corresponding flow conservation model is to
minimise

$$t_d \tag{3.33}$$

subject to

$$\sum_{(i,j) \in \sigma(i)} f_{ij} - \sum_{(j,i) \in \delta(i)} f_{ji} = \begin{cases} 1 & i = s \\ -1 & i = d \\ 0 & \text{otherwise} \end{cases}, \quad \forall i \in \mathcal{V}, \tag{3.34}$$

where $\sigma(i)$ is a function that returns a set of all the arcs adjacent to i, and $\delta(i)$ is a function that returns a set of all the arcs that i is adjacent to. The start time variables need to be connected with the flow variables using

$$t_v = \min_{l = \sigma(v)} \Big(t_l + \tau_{lv} + M(1 - f_{lv}) \Big), \forall v \in \mathcal{V}, \tag{3.35}$$

where t_v is the start time for node v, τ_{lv} is the duration on arc (l, v) and the function $\gamma(v)$ returns all the vertices that v is adjacent to, and M is some large constant real value. Equation (3.35) can be written as linear constraints,

$$t_v \leq t_l + \tau_{lv} + M(1 - f_{lv}), \ \forall l \in \gamma(v), \ v \in V, \ v \neq l, \tag{3.36}$$

$$t_v \geq t_l + \tau_{lv} - M(1 - f_{lv}), \ \forall l \in \gamma(v), \ v \in V, \ v \neq l. \tag{3.37}$$

The linearisation constraints for the velocity and duration are

$$v_a = \sum_{x \in \mathcal{X}} v_{xa} \alpha_{xa}, \ \forall a \in \mathcal{A}, \tag{3.38}$$

$$\tau_a = \sum_{x \in \mathcal{X}} \tau_{xa} \alpha_{xa}, \ \forall a \in \mathcal{A}, \tag{3.39}$$

where v_{xa} and τ_{xa} are breakpoints sampled from a multiplicative inverse function that involves the velocity and duration the BEV travels on an arc a. The variables v_a and τ_a are linear combinations of breakpoints v_{xa} and τ_{xa} respectively and form convex combinations that satisfies,

$$\sum_{x \in \mathcal{X}} \alpha_{xa} = 1, \ \forall a \in \mathcal{A}, \tag{3.40}$$

only two adjacent values in the linear combinations can be greater than zero and this is realised with

$$\alpha_{xa} \leq h_{(x-1)a} + h_{xa}, \ \forall x \in \mathcal{X} \setminus \{0\}, \ \forall a \in \mathcal{A}, \tag{3.41}$$

where $\mathcal{X} = \{0, \ldots, X\}$, with 0 the first breakpoint index and X the last breakpoint index. To ensure only one breakpoint index variable can be selected,

$$\sum_{x \in \mathcal{X} \setminus \{X\}} h_{xa} = 1, \ \forall a \in \mathcal{A}, \tag{3.42}$$

which allows for two adjacent α_{xa} values to be greater than zero. The domain of the decision variables are

$$0 \leq \alpha_{xa} \leq 1, \ \forall x \in \mathcal{X}, \ \forall a \in \mathcal{A}, \tag{3.43}$$

$$h_{xa} \in \{0, 1\}, \ \forall x \in \mathcal{X}, \ \forall a \in \mathcal{A}, \tag{3.44}$$

$$t_a, v_a, \tau_a \geq 0, \ \forall a \in \mathcal{A}. \tag{3.45}$$

The linearisation constraints of the flow conservation model are similar to the piecewise linearisation method of the path-based model.

With both models, (3.20) - (3.32) and (3.33) - (3.45), there are no explicit restrictions on the velocity, and consequently the duration on each segment. The obvious solution is to use the maximum velocity that corresponds to $x \in \mathcal{X}$, which reduces both problems to the shortest path problem. These models serve as a basis for this thesis and can be extended to allow the velocity variable on each segment to depend on a variable that represents energy usage.

3.2.3 *Initial model verification*

The outputs of the models can be verified with a small example, Figure 14 shows generated results from the flow conservation model and the path-based model for which node 711 was selected as the starting node and node 700 as the ending node. A single output graph shows the results of both models since they are the same. Another easy verification of the models up until now is a simple shortest path algorithm since the velocities should be chosen at maximum, reducing the problem to a shortest path problem. For the linearisation of velocities and durations, eleven evenly spaced breakpoints were chosen from 0 $\mathrm{m\,s^{-1}}$ to 33.33 $\mathrm{m\,s^{-1}}$. As expected, the velocities were maximum for all the selected arcs since there are no other restrictions on the velocities. The durations corresponded to the simulated values since a breakpoint was selected.

The flow conservation model in Section 3.2.2 and path-based model in Section 3.2.1 do not include energy constraints that take the characteristics of the vehicle into account. The only simulation aspect taken into account is the duration a vehicle travels for a specified velocity. In the next section simulations of a BEV with velocity v on an arc or segment accounting for the road slope, wind direction and wind speed are used for the extended optimisation models.

(a) Input Graph. (b) Output Graph.

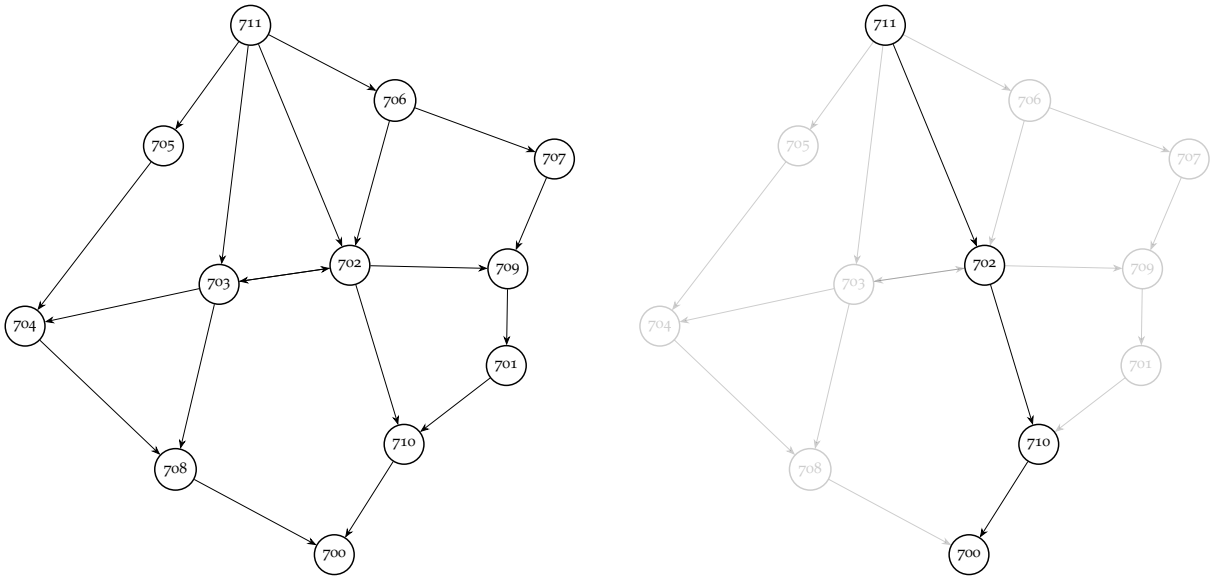


Figure 14: Input and output graphs of the flow conservation and path-based model.

3.3 INITIAL PRACTICAL MODELS

The energy usage on each segment of the road is of extreme importance and requires the modelling of an electric vehicle. Before energy variables can be linearised, the relationship between the velocity of a vehicle and the energy required to maintain the velocity v for a distance d needs to be determined.

3.3.1 *Modelling an electric vehicle*

Aerodynamic drag, tyre rolling resistance and gravity are the main forces that directly oppose the movement of a vehicle. Figure 15 shows a simple free body diagram for a vehicle on a segment with length x and slope θ.

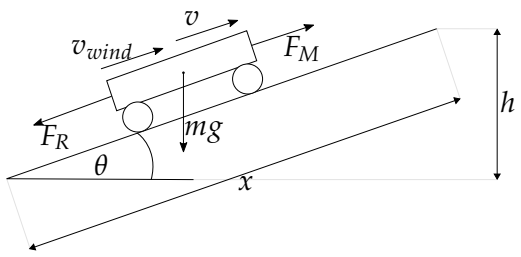


Figure 15: Simple free body diagram.

The force supplied by the electric motor is F_M, when applied over the distance x, the work done by the electric motor is

$$W_M = F_M x. \tag{3.46}$$

For a constant speed v, equation (3.46) can be written as

$$W_M = F_M v t, \tag{3.47}$$

where

$$v = \frac{x}{t}. \tag{3.48}$$

The force F_R includes the rolling resistance and wind resistance,

$$F_R = F_d + F_r, \tag{3.49}$$

the drag force of the vehicle is given by,

$$F_d = \frac{1}{2}\rho C_d A(v - v_{wind})^2, \tag{3.50}$$

where ρ is the mass density of air, v is the constant velocity of the vehicle on the slope, v_{wind} is the wind velocity relative to the ground, C_d is the drag coefficient and A is the reference area [61]. Define the rolling resistance coefficient as the magnitude of force needed to push a wheeled vehicle forward.

$$F_r = C_{rr}N = C_{rr}mgcos(\theta) \tag{3.51}$$

The effort required to overcome and the resistance to motion is given by

$$W_R = F_R vt. \tag{3.52}$$

Given that the vehicle is on a slope, the potential energy at the end of the slope is

$$U_g = mgh = mgx\sin\theta, \tag{3.53}$$

which is included in the total energy required by the electric motor. The total energy required by the electric motor with efficiency η_M is then given by

$$E_M = \frac{1}{\eta_M}\Big[mgx\sin\theta + xF_R\Big]. \tag{3.54}$$

The efficiency η_M needs to be determined empirically. The total work done by the motor on a segment with distance x can be written as

$$W_M = \frac{x}{\eta_M(v)}\Big[mg\sin\theta + \frac{1}{2}\rho C_d A(v - v_{wind})^2 + C_{rr}mgcos(\theta)\Big], \tag{3.55}$$

assuming the vehicle is not accelerating. These equations do not take acceleration or deceleration between segments into account, but W_M can be used in the optimisation model to approximate the energy capacity of the battery at the beginning of each segment. Let the total work done on each segment depend on the constant velocity travelled on the segment, $e_s(v)$. For the path-based model, the total work done on segment s is

$$e_s = W_M(s, v) = \sum_{x\in\mathcal{X}} e_{xs}a_{xs}, \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}. \tag{3.56}$$

To determine the battery state of the vehicle after each segment, use

$$E_s = E_{(s-1)} + e_s,\ \forall s \in \mathcal{S}(\rho)/\{0\},\ \rho \in \mathcal{P}, \tag{3.57}$$

which captures the energy in the battery for the previous segment and the work done on the current segment. For the initial segment with full battery capacity ϵ_H,

$$E_s = \epsilon_H + e_s, s = \mathcal{S}(\rho)^{(0)},\ \forall,\ \rho \in \mathcal{P}. \tag{3.58}$$

The total energy in the battery at each segment is capped to battery specifications,

$$\epsilon_L \leq E_s \leq \epsilon_H, s \in \mathcal{S}(\rho),\ \rho \in \mathcal{P}, \tag{3.59}$$

the total work done on a segment is limited to the maximum energy capacity of the battery,

$$-\epsilon_H \leq e_s \leq \epsilon_H, s \in \mathcal{S}(\rho),\ \rho \in \mathcal{P}. \tag{3.60}$$

For the flow conservation model, similar constraints are added,

$$e_a = W_M(a, v) = \sum_{x\in\mathcal{X}} e_{xa}\alpha_{xa},\ \forall a \in \mathcal{A}. \tag{3.61}$$

To linearise the work done on each arc,

$$E_k \leq E_b + e_{bk} + M(1 - f_{bk}),\ \forall b \in \gamma(k),\ \forall k \in \mathcal{V}, b \neq k, \tag{3.62}$$

$$E_k \geq E_b + e_{bk} - M(1 - f_{bk}),\ \forall b \in \gamma(k),\ \forall k \in \mathcal{V}, b \neq k, \tag{3.63}$$

to capture the battery energy-state after each vertex was explored. The total energy in the battery at each vertex is capped to battery specifications,

$$\epsilon_L \leq E_v \leq \epsilon_H,\ v \in \mathcal{V}. \tag{3.64}$$

The total work done on an arc should also be limited to the maximum energy capacity of the battery,

$$-\epsilon_H \leq e_a \leq \epsilon_H,\ a \in \mathcal{A}. \tag{3.65}$$

Both models account for energy usage but fail to account for variable wind speeds on a segment. Wind speed depends on the start time of each segment when this is taken into account, the model can easily be adapted to account for other weather conditions.

3.3.2 *Solar irradiance*

Although the thesis's focus is not numerical weather prediction (NWP), it still suffices to give some background on the topic. Terrain roughness can cause variability in the incoming solar radiation field. Figure 16, introduced by [62] clearly illustrates the major topographic processes that affects solar radiation in rugged regions for clear-sky conditions. Point B shows the solar flux is intercepted by a higher elevation (the mountain to the right), which causes a shadow. Point B is still affected by diffuse irradiance and the neighbouring terrain-reflected irradiance. Figure 17 illustrates the origin of diffuse- and circumsolar irradiance as an auxiliary to Figure 16.

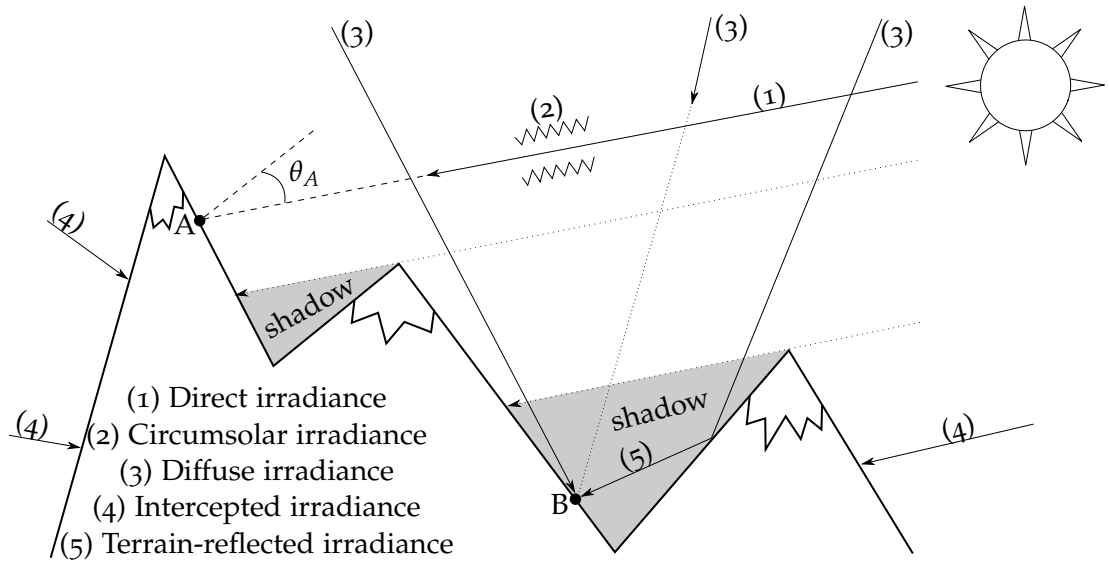


Figure 16: Major topographic processes that affect solar radiation flux in mountainous regions [62].

Loutzenhiser et al. [63] used empirical data for comparison of multiple solar irradiance simulation models [64–69]. The Isotropic sky model [70,71] is the simplest model and performed the worst, but has the advantage of not relying on empirical data of a region. The Perez formulation [69] provided the best results but relied on empirical data to quantify the diffuse components.

For the simulation, we use the Hay–Davies model [68] where the total irradiance on the tilted solar panel is:

$$I_T = (I_b + I_d A_i) R_b + I_d (1 - A_i) \Big(\frac{1 + cos(\beta)}{2} \Big) + I \rho_g \Big(\frac{1 - cos(\beta)}{2} \Big), \tag{3.66}$$

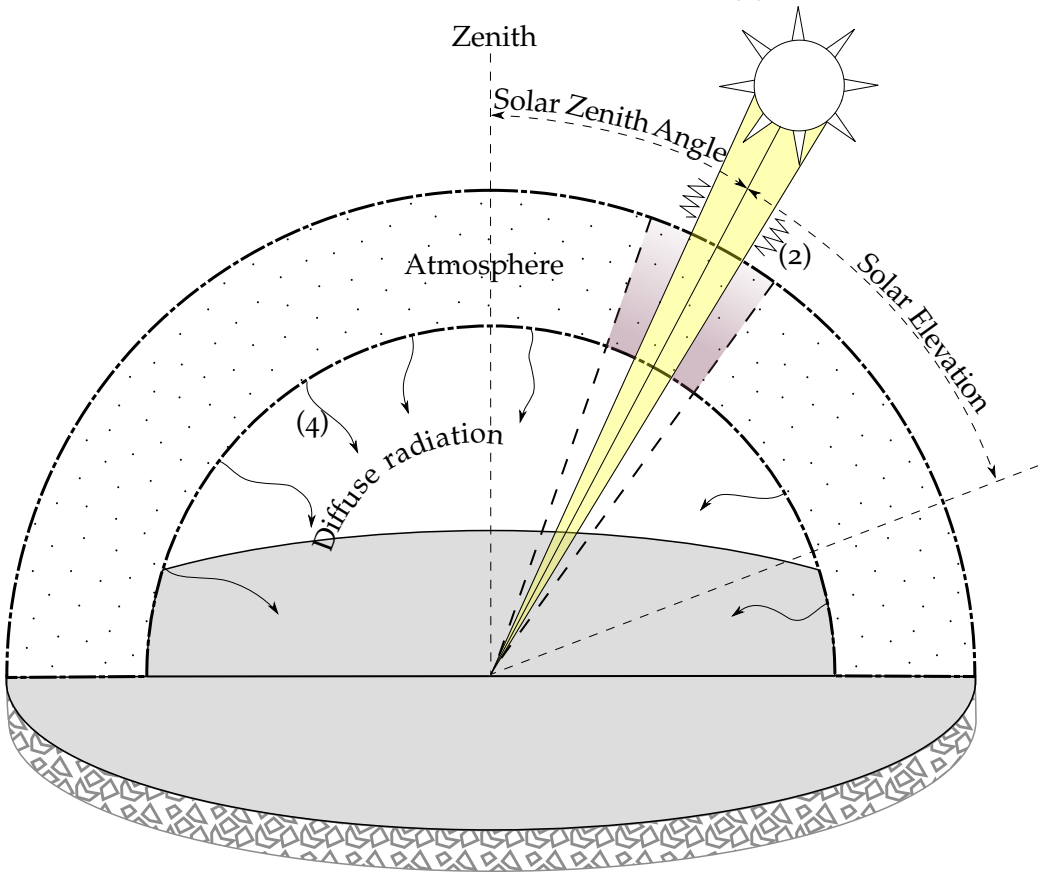


Figure 17: Auxiliary illustration for Figure 16.

where A_i is an anisotropic index, in the case of no beam, the calculated diffuse is isotropic, e.g. $A_i = 0$. β is the solar panel tilt angle from the horizon. R_b is the ratio of tilted and horizontal solar beam irradiance. I_b is the direct-normal component of solar irradiance on the horizontal surface, and I_d is the global diffuse horizontal solar irradiance and I is the global horizontal solar irradiance. ρ_g is the hemispherical-hemispherical ground reflectance. The total energy gained from the solar panel for a segment is

$$E_{solar} = \int_{t_s}^{t_f} I_T(t) A_p \mu_p \, dt, \tag{3.67}$$

where A_p is the area of the solar panel and μ_p is the efficiency of the solar panel. E_{solar} depends on the starting and ending times of each segment and can be added to the energy graph for a specified starting time and velocity.

3.3.3 *Wind direction and -speed*

Wind direction describes the direction from which it originates, with 0° a wind that originates from the north and blows south, Figure 18 illustrates this concept. By using the dot product between the vehicle direction and the wind direction, the wind speed component is determined. The longitude and latitude of the starting and ending nodes of each segment determine the vehicle's direction. Every longitude and latitude can be mapped to a Cartesian plane, and the edge cases can be handled separately by translating them to a more suitable position. The vehicle direction is expressed as a unit vector,

$$\boldsymbol{p} = \frac{1}{\sqrt{(x_t - x_s)^2 + (y_t - y_s)^2}} \begin{bmatrix} x_t - x_s \\ y_t - y_s \end{bmatrix}, \tag{3.68}$$

where x_s, x_t are the longitudes and y_s, y_t are the latitudes of the source and target nodes respectively. Assume the vehicle's travelling direction is the same as the arc the vehicle should move on, starting at the source node and ending at the target node.

The next step is to get the wind direction on the same axis as the vehicle direction. Since the wind direction indicates where it originates from, and the axis of a Cartesian plane is a 90° counter-clockwise rotation of the wind direction, use the following to align the direction in which the wind flows with,

$$\boldsymbol{w} = v_{wind} \begin{bmatrix} sin(\alpha) \\ cos(\alpha) \end{bmatrix}, \tag{3.69}$$

where α is the wind direction and v_{wind} is the wind speed.

Figure 18: Illustration of wind direction and vehicle direction.

To determine the wind component in Figure 18, consider the dot product between $\boldsymbol{w}$ and $\boldsymbol{p}$

$$\boldsymbol{p} \cdot \boldsymbol{w} = |\boldsymbol{p}||\boldsymbol{w}|cos(\theta). \tag{3.70}$$

Since $\boldsymbol{p}$ is a unit vector,

$$|\boldsymbol{w}|cos(\theta) = \boldsymbol{p} \cdot \boldsymbol{w}, \tag{3.71}$$

The wind component is described with the vector,

$$\boldsymbol{k} = |\boldsymbol{w}|cos(\theta)\boldsymbol{p} = (\boldsymbol{p} \cdot \boldsymbol{w})\boldsymbol{p}. \tag{3.72}$$

For calculating the drag force we can use,

$$v_{wind} = \begin{cases} -|\boldsymbol{k}|, & \text{if } \frac{\boldsymbol{k}}{|\boldsymbol{k}|} = -\boldsymbol{p}, \\ |\boldsymbol{k}|, & \text{if } \frac{\boldsymbol{k}}{|\boldsymbol{k}|} = \boldsymbol{p}, \end{cases} \tag{3.73}$$

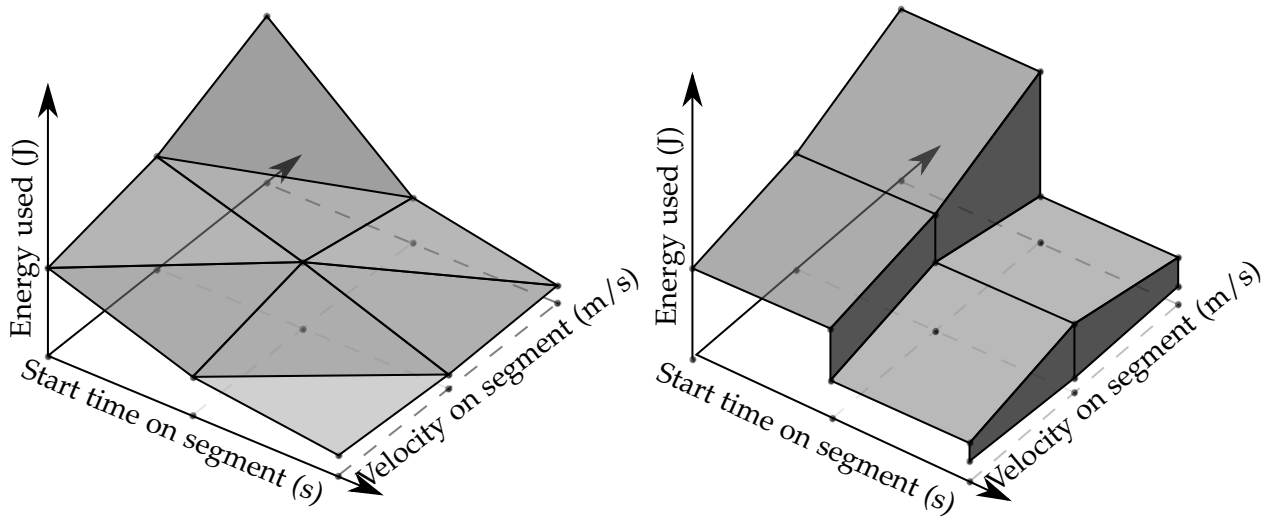
in equation (3.50).

3.3.4 *Modelling energy graphs in a linear fashion*

The energy graph incorporates multiple environmental- and vehicle factors. The weather data and physics of the vehicle are non-linear which results in a non-linear energy graph for each segment. In order to include these graphs in the MILP models, the energy graphs need to be linearised. Assume that the velocity and start times forms a grid for the sake of simplicity, Figure 19c illustrates this idea.

(a) Triangle method.

(b) *Block method*.



(c) Example of a top-view energy graph.

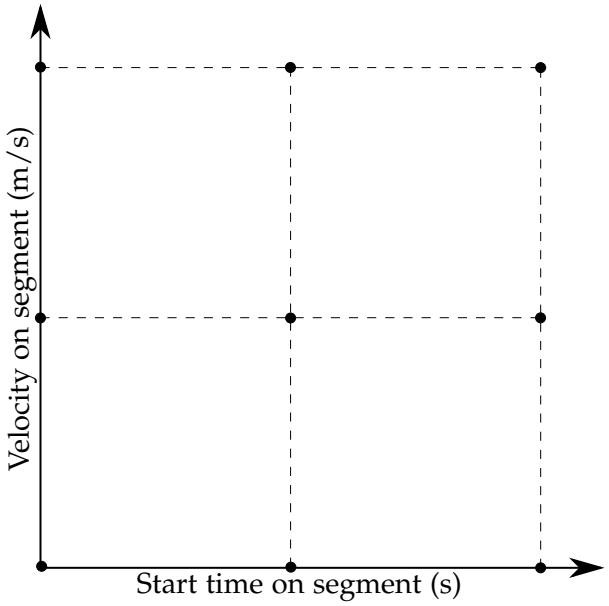


Figure 19: Illustrations to represent energy graphs.

There are multiple methods to do this, two specific approaches are followed, visual representation of the approaches are shown in Figure 19. The *triangle method* in Figure 19a fits non-linear functions better than with the *block method* in Figure 19b, but formulating the problem as triangles has computation repercussions. The obvious drawback of the *block method* is the constant energy value for a whole time frame, where the *triangle method* linearly approximates the energy difference between time-frames. As the number of data-points increase, both Figure 19a and Figure 19b approach the same non-linear values. In cases where there are computation-time and availability of data trade-off, for more data-points the *block method* may be a better option than the *triangle method*. When there is a lack of comprehensive data, the *triangle method* may be more beneficial than the *block method*.

Constraint sets (3.74) - (3.79) describes the *triangle method* of [51] where (v_x, t_y) are sampling coordinates for $x \in \mathcal{X}, y \in \mathcal{Y}$, e_{xy} is the function evaluated at each breakpoint (v_x, t_y). $\beta_{xy} \in [0,1]$ is a continuous variable (one per breakpoint), used for computing the convex combinations for the three-dimensional space. β_{xy} should be defined as a Special Ordered Set of Type 3 (SOS3) but current MILP solvers do not have such functionality. Create binary decision variables $\kappa_{xy}^{(u)}, \kappa_{xy}^{(l)}$ for the upper and lower triangle in the rectangle. Let v be the velocity of the BEV on a segment in Figure 19a,

$$v = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \beta_{xy} v_x, \tag{3.74}$$

and the start time on a segment be,

$$t = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \beta_{xy} t_y, \tag{3.75}$$

with the energy used,

$$e = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \beta_{xy} e_{xy}, \tag{3.76}$$

where v_x, t_y and e_{xy} are breakpoints associated with $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. The variables v, t and e are linear combinations of v_x, t_y and e_{xy}, which form convex combinations that satisfies

$$\sum_{x \in \mathcal{X} \setminus \{X\}} \sum_{y \in \mathcal{Y} \setminus \{Y\}} \beta_{xy} = 1, \tag{3.77}$$

where $\mathcal{X} = \{0, \ldots, X\}$ and $\mathcal{Y} = \{0, \ldots, Y\}$. The auxiliary constraints that force one triangle to be selected, either an upper or lower triangle of the rectangle, which is identified by the associated breakpoint, are given by

$$\sum_{x \in \mathcal{X} \setminus \{X\}} \sum_{y \in \mathcal{Y} \setminus \{Y\}} \kappa_{xy}^{(u)} + \kappa_{xy}^{(l)} = 1. \tag{3.78}$$

The selected triangle is a convex combination of three points, the selected breakpoint and two adjacent points, and this is modelled with

$$\beta_{xy} \leq \kappa_{xy}^{(u)} + \kappa_{xy}^{(l)} + \kappa_{x(y-1)}^{(u)} + \kappa_{(x-1)y}^{(l)} + \kappa_{(x-1)(y-1)}^{(u)} + \kappa_{(x-1)(y-1)}^{(l)} \quad \forall x \in \mathcal{X} \setminus \{0\}, y \in \mathcal{Y} \setminus \{0\}. \tag{3.79}$$

To describe the *block method*, Special Ordered Set of Type 2 (SOS2) with big$M$ constraints are applied. Figure 19b can be modelled with,

$$v = \sum_{x \in \mathcal{X}} v_x \alpha_x, \tag{3.80}$$

where v is the velocity on the segment e is the energy used on a segment and is expressed as the convex combination,

$$e \leq \sum_{x \in \mathcal{X}} \alpha_x e_{xy} + M(1 - \kappa_y), \qquad \forall y \in \mathcal{Y}, \tag{3.81}$$

$$e \geq \sum_{x \in \mathcal{X}} \alpha_x e_{xy} - M(1 - \kappa_y), \qquad \forall y \in \mathcal{Y}, \tag{3.82}$$

where κ_y is the variable to select a start time t between t_y and $t_{(y+1)}$, according to

$$t \leq \sum_{y \in \mathcal{Y} \setminus \{Y\}} \kappa_y t_{(y+1)}. \tag{3.83}$$

$$t \geq \sum_{y \in \mathcal{Y} \setminus \{Y\}} \kappa_y t_y, \tag{3.84}$$

The constraints for linear piecewise approximation are still required,

$$\alpha_x \leq h_{(x-1)} + h_x, \ \forall x \in \mathcal{X}, \tag{3.85}$$

$$\sum_{x\in\mathcal{X}} \alpha_x = 1, \tag{3.86}$$

$$\sum_{x\in\mathcal{X}\setminus\{X\}} h_x = 1. \tag{3.87}$$

To ensure a single start time is selected,

$$\sum_{y\in\mathcal{Y}\setminus\{Y\}} \kappa_y = 1. \tag{3.88}$$

In the next section, the path-based and flow conservation models are extended by using the *block-* and *triangle method*.

3.3.5 *Extended path-based model*

To consider the energy, a multi-dimensional linear piecewise approximation needs to be formulated using the start time, velocity and energy used on each segment. Describe each point on the plane as a convex combination with β_{xys} as coefficient,

$$v_s = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \beta_{xys} v_{xs}, \qquad \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}, \tag{3.89}$$

$$t_s = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \beta_{xys} t_{ys}, \qquad \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}, \tag{3.90}$$

$$e_s = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \beta_{xys} e_{xys}, \qquad \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}(d) \tag{3.91}$$

such that

$$\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \beta_{xys} = 1, \qquad \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}, \tag{3.92}$$

$$\sum_{x\in\mathcal{X}\setminus\{X\}}\sum_{y\in\mathcal{Y}\setminus\{Y\}} \kappa^{(u)}_{xys} + \kappa^{(l)}_{xys} = 1, \qquad \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}, \tag{3.93}$$

$$\begin{aligned}\beta_{xys} \le \kappa^{(u)}_{xys} + \kappa^{(l)}_{xys} + \kappa^{(u)}_{x(y-1)s} + \kappa^{(l)}_{(x-1)ys} + \kappa^{(u)}_{(x-1)(y-1)s} + \kappa^{(l)}_{(x-1)(y-1)s} \\ \forall x \in \mathcal{X}\setminus\{0\}, y \in \mathcal{Y}\setminus\{0\}, s \in \mathcal{S}(\rho), \rho \in \mathcal{P}.\end{aligned} \tag{3.94}$$

The path-based model can be extended by considering the energy consumed on each segment s, given velocity v_s and the time on which the vehicle starts on the segment t_s. The binary decision variables κ_{ys} corresponds to all the specified sequential start times on each segment, the start time t_s can be any value between two adjacent start times, which is given by

$$t_s \le \sum_{y\in\mathcal{Y}\setminus\{Y\}} \kappa_{ys} t_{(y+1)s}, \qquad \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}, \tag{3.95}$$

$$t_s \ge \sum_{y\in\mathcal{Y}\setminus\{Y\}} \kappa_{ys} t_{ys}, \qquad \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}. \tag{3.96}$$

To allow only a single start time to be selected for each segment,

$$\sum_{y\in\mathcal{Y}\setminus\{Y\}} \kappa_{ys} = 1, \qquad \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}. \tag{3.97}$$

The energy used, given the start time and velocity of a segment can be expressed as

$$e_{s\rho} = \sum_{x\in\mathcal{X}} \alpha_{xs} e_s(x, y). \tag{3.98}$$

Relate κ_{ys} to the energy usage since it indicates a specified start time. Big M constraints are required to ensure that the energy usage on the selected start time becomes an equality,

$$e_{s\rho} \leq \sum_{x \in \mathcal{X}} \alpha_{xs} e_s(x,y) + M(1-\kappa_{ys}), \qquad \forall y \in \mathcal{Y}, \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}, \tag{3.99}$$

$$e_{s\rho} \geq \sum_{x \in \mathcal{X}} \alpha_{xs} e_s(x,y) - M(1-\kappa_{ys}), \qquad \forall y \in \mathcal{Y}, \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}. \tag{3.100}$$

M needs to be chosen with care because a too large M value can cause numerical instability. In the implementation M was chosen as two times the battery capacity since the energy usage on a segment should not be more than the battery capacity of the vehicle.

3.3.6 *Extended flow conservation model*

To consider the energy usage, a multi-dimensional linear piecewise approximation using the start time, velocity and energy used on each arc needs to be formulated.

$$\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \beta_{xya} = f_a, \qquad \forall a \in \mathcal{A}, \tag{3.101}$$

allows the convex combinations associated with β_{xya} to be disabled for arc a when there is no flow over arc an a. Similar convexity constraints are required for the velocity, energy and start time. The velocity is given as a linear combination of breakpoints associated with $x \in \mathcal{X}$,

$$v_a = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \beta_{xya} v_{xa}, \qquad \forall a \in \mathcal{A}, \tag{3.102}$$

and the energy is a linear combination of breakpoints associated with both $x \in \mathcal{X}$ and $y \in \mathcal{Y}$,

$$e_a = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \beta_{xya} e_{xya}, \qquad \forall a \in \mathcal{A}. \tag{3.103}$$

Unlike the path-based model, the start time is not associated with a segment, but rather a vertex. Since β_{xya} has an arc index x, vertices associated with the start time need to be related to arcs. The function $\sigma(v)$ returns the arcs where the vertex v is the source vertex. The start times are then written as a linear combination,

$$t_v = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \beta_{xya} t_{yv}, \qquad \forall a \in \sigma(v),\ v \in \mathcal{V}, \tag{3.104}$$

The triangle selection variables $\kappa^{(u)}_{xya}$, $\kappa^{(l)}_{xya}$ should be zero if there is no flow over an arc a,

$$\sum_{x \in \mathcal{X} \setminus \{X\}} \sum_{y \in \mathcal{Y} \setminus \{Y\}} \kappa^{(u)}_{xya} + \kappa^{(l)}_{xya} = f_a, \qquad \forall a \in \mathcal{A}. \tag{3.105}$$

To ensure that at most three adjacent breakpoints are selected for the convex combinations, v_a, e_a and t_v,

$$\beta_{xya} \leq \kappa^{(u)}_{xys} + \kappa^{(l)}_{xya} + \kappa^{(u)}_{x(y-1)a} + \kappa^{(l)}_{(x-1)ya} + \kappa^{(u)}_{(x-1)(y-1)a} + \kappa^{(l)}_{(x-1)(y-1)a} \forall x \in \mathcal{X} \setminus \{0\}, y \in \mathcal{Y} \setminus \{0\}, a \in \mathcal{A}, \tag{3.106}$$

β_{xya}, for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$ will be zero when there is no flow over arc a.

Add a binary variable κ_{vy} for every start time t_{yv}, where $v \in \mathcal{V}$ and $y \in \mathcal{Y}$. $\mathcal{Y}$ is a set containing all the indices for the start times. Start times are constrained to the provided start times,

$$t_v \leq \sum_{y \in \mathcal{Y} \setminus \{Y\}} \kappa_{ay} t_{(y+1)v} - M(f_a - 1), \qquad \forall a \in \sigma(v),\ v \in \mathcal{V}, \tag{3.107}$$

$$t_v \geq \sum_{y \in \mathcal{Y} \setminus \{Y\}} \kappa_{ay} t_{yv} + M(f_a - 1), \qquad \forall a \in \sigma(v),\ v \in \mathcal{V}, \tag{3.108}$$

where only a single start time can be chosen if there exist a flow f_a,

$$\sum_{y \in \mathcal{Y} \setminus \{Y\}} \kappa_{ay} = f_a, \qquad \forall a \in \sigma(v),\ v \in \mathcal{V}. \tag{3.109}$$

The energy usage for an arc depends on the start time. To consider factors such as wind speed and wind direction, constraints similar to (3.99) and (3.100), are included

$$e_a \leq \sum_{x \in \mathcal{X}} \alpha_{xa} e_{xya} + M(1 - \kappa_{ay}), \qquad \forall a \in \mathcal{A}, \forall y \in \mathcal{Y}, \tag{3.110}$$

$$e_a \geq \sum_{x \in \mathcal{X}} \alpha_{xa} e_{xya} - M(1 - \kappa_{ay}), \qquad \forall a \in \mathcal{A}, \forall y \in \mathcal{Y}, \tag{3.111}$$

in the formulation. The value of M is chosen as two times the battery capacity.

3.3.7 *Results and initial model comparison*

A single arc/segment is used to compare the techniques for modelling energy graphs and determine a reasonable number of velocity steps. Figure 20a shows the results for the different techniques described in the previous sections. Table 1 shows the vehicle characteristics chosen to generate the results. The vehicle characteristics such as drive-train efficiency, solar panel efficiency, frontal area and drag coefficient are approximations of the North-West University's Solar car that competed in the 2018 Sasol Solar Challenge. The battery capacity and solar panel area are typical for solar powered BEVs that compete in solar races [72].

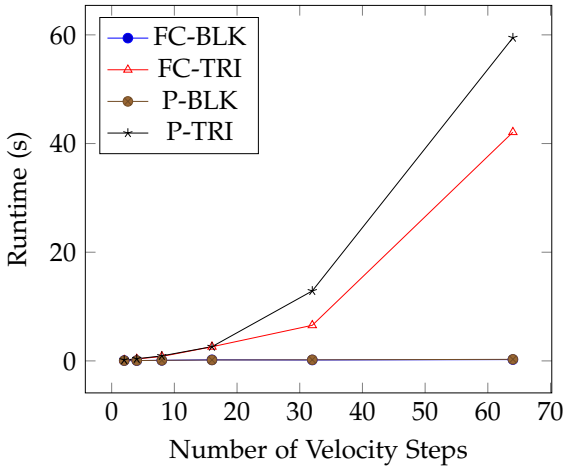
Table 1: Solar powered vehicle characteristics used in optimisation runs.

Name	**Symbol**	**Value**
Battery capacity	ϵ_H	18 MJ
Minimum battery energy	ϵ_L	3.6 MJ
Drive-train efficiency	η_M	0.93
Solar panel area	A_p	4 m^2
Solar panel efficiency	μ_p	0.2
Tyre roll coefficient	C_{rr}	0.026
Drag coefficient	C_d	0.225
Frontal area	A	0.7 m^2

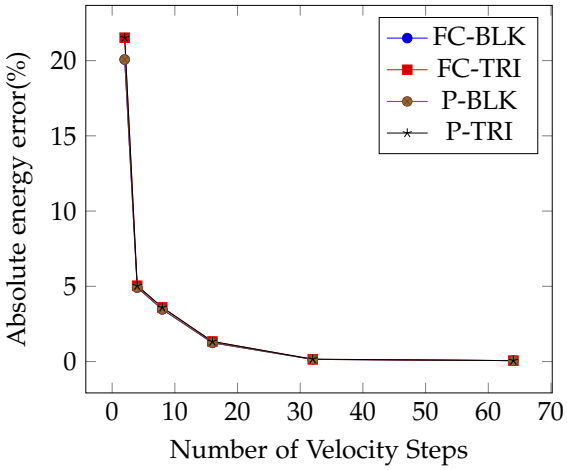
The naming conventions for the models are as follows, the prefix FC is for the flow conservation model and P for the path-based model, this is followed by TRI when the triangle method for the energy graphs is used and BLK when the block method is used. The detailed table of results is given in Appendix A.

The velocity steps are varied to observe the effect on the performance of the models. From Figure 20a, it is evident that the triangle method does not scale as well as the block method. The runtime grows exponentially as the number of velocity steps increases; this is not the case for the block method since the binary selection variable κ_{*y} does not have an index for any element in $\mathcal{X}$. The flow conservation model seems to scale better than the path-based model, but more experiments are needed to verify the path-based and flow conservation models' scalability. A single arc/segment is not a good test instance for comparing the path-based model and flow

(a) Runtime scalability of different approaches and velocity steps.



(b) Energy error of different approaches and velocity steps.



(c) Objective value as granularity of velocity increase.

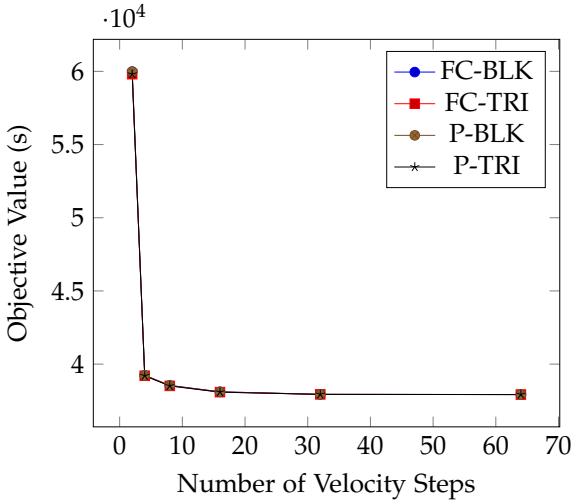


Figure 20: Effect of increased velocity steps.

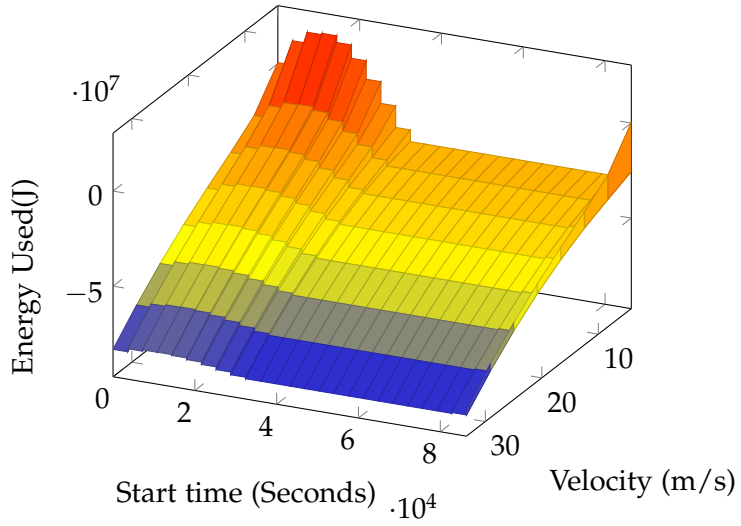
conservation model, but this experiment gives a decent indication of how many velocity steps to choose per segment. Different types of graphs will be considered at a later stage of the thesis.

The optimal objective value should be the same for the path-based model and the flow conservation model whenever the same energy graph method and velocity steps are used; this is apparent in Figure 20c. These values are close to the simulated values, Figure 20b indicates the energy error on the optimisation solution. As the granularity increases, the absolute energy error decreases, which shows the model corresponds to the simulated energy used. Figure 20c and Figure 20b strongly correlate, as the absolute error on the energy graph decreases, the optimal objective value nears the "true" optimal[2].

Figure 21 shows the difference between generating an energy graph with the block method and with the triangle method. Figure 21a shows the block method, and Figure 21b shows the triangle method, both with eight evenly spaced velocity points. Twenty-four start times, separated by one-hour, was used of which the first started at 8 AM. In addition, a latitude of -29.86306279 and

2 There can only exist a single optimal value (multiple solutions can exist), for each given dataset and model. Multiple optimal values refer to different granular datasets, such as increased velocity steps. The "true" optimal refers to the optimal solution with infinitesimal granular inputs which are theoretical.

(a) Energy graph used with the block method.



(b) Energy graph used with the triangle method.

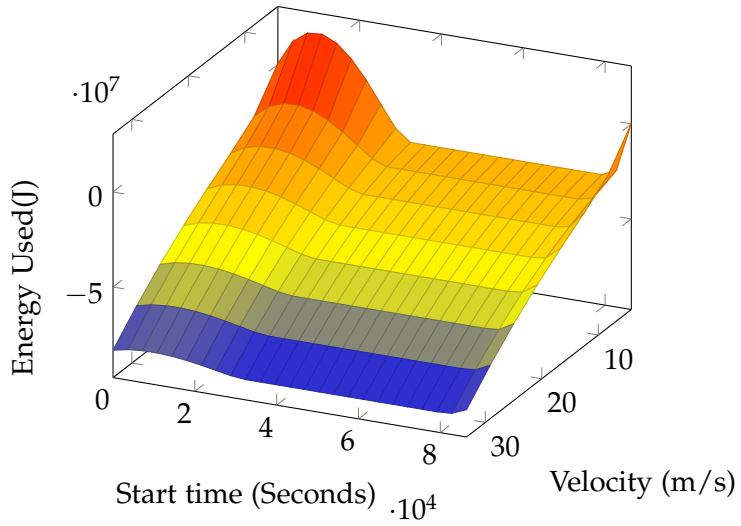


Figure 21: Difference between block-based and triangle-based energy graphs. Negative energy shows the energy the vehicle uses and positive energy is the energy the vehicle gains.

a longitude of 31.01972238 were used. From both figures, the sunrise is clear. For all the solutions, the initial start time is after 9 AM. The error on the energy used was very close for the block and triangle method, and there seems to be no benefit for using the triangle method other than less Big M constraints in the model. Even though the triangle method may not be feasible for large datasets, the method can validate smaller datasets' results.

3.4 ALGORITHMIC IMPROVEMENTS

The path-based model allows the formulation to be used as a heuristic when the number of paths is limited. With a smaller search space, feasible solutions may be found with relative ease, although there is no solution quality guarantee with such an approach. When a solution is found with a reduced search space, it can be used as a warm-start for the original model or the flow conservation model. The subsets of paths we consider are the shortest paths. Although the number of paths to generate may not be apparent, starting at the shortest path is a reasonable assumption.

3.4.1 *k-shortest paths*

There are multiple variations of the k-shortest path, a loopless variation and a loopy variation. With the loopless variation, each path must contain distinct arcs. Multiple other similar algorithms and variations of the problem were published, including Yen's algorithm [73] for loopless k-shortest paths and Eppstein's algorithm [74] to loopy k-shortest paths. The k-shortest path algorithm is used as a preprocessor for the path-based model to limit the number of generated paths. The focus is on the implementation of a loop-less algorithm since the initial assumption was that all paths are directed path graphs. Dijkstra's single-source shortest path algorithm and the Bellman-Ford algorithm [22] can be extended to generate more than one path. Algorithm 5

Algorithm 5 Dijkstra's algorithm generalised to k-shortest paths.

Input: Finite weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, source node s, target node t, k number of paths to find.
Output: $\mathcal{P}$ a set of paths from s to t.

1: $\mathcal{P} \leftarrow \emptyset$
2: $c_u \leftarrow 0, \qquad \forall u \in \mathcal{V}$
3: $\mathcal{P}_s \leftarrow \{s\}$ with cost $C_s = 0$
4: **while** $\mathcal{B}$ is not empty and $c_t < k$ **do**
5: $\quad \mathcal{P}_u \leftarrow$ extract path with minimum cost from $\mathcal{B}$
6: $\quad \mathcal{B} \leftarrow \mathcal{B} - \{\mathcal{P}_u\}$
7: $\quad c_u \leftarrow c_u + 1$
8: $\quad$ **if** $u = t$ **then**
9: $\quad\quad \mathcal{P} \leftarrow \mathcal{P} \bigcup \{\mathcal{P}_u\}$
10: $\quad$ **end if**
11: $\quad$ **if** $c_u \leq k$ **then**
12: $\quad\quad$ **for each** vertex v adjacent to u **do**
13: $\quad\quad\quad$ **if** $(u, v) \notin \mathcal{P}_u$ **then**
14: $\quad\quad\quad\quad \mathcal{P}_v \leftarrow \mathcal{P}_u \bigcup (u, v)$ with cost $C_v + w_{uv}$
15: $\quad\quad\quad\quad \mathcal{B} \leftarrow \mathcal{B} \bigcup \mathcal{P}_v$
16: $\quad\quad\quad$ **end if**
17: $\quad\quad$ **end for**
18: $\quad$ **end if**
19: **end while**
20: **return** $\mathcal{P}$

generates the k-shortest directed path graphs using a generalised version of Dijsktra's algorithm. Loops can easily be added by excluding line 13 and changing line 14 to concatenate P_u with (u, v). The algorithm can be viewed as entities racing to the target node each taking another path and the first k to reach the target node are the winners. Since the path with the minimum cost is considered at each step, the paths are sorted from the shortest path to the longest path of the k-paths. In line 4 the algorithm will keep exploring the graph as long as the count on the target node c_t is less than k. The path with the minimum cost is selected on line 5 and removed from the queue of paths $\mathcal{B}$ in line 6. The count of the explored vertex where the path ends is increased in line 7. When the path ends with the target node t, the path $\mathcal{P}_t$ is added to the resulting paths $\mathcal{P}$. At each step, when k explorations were not done on a vertex, each adjacent vertex is added to distinct paths in the queue $\mathcal{B}$, this is shown in lines 12-17. Once the algorithm terminates, less or equal to k shortest paths in $\mathcal{P}$ is returned.

3.4.2 *Path-based model as a parallel problem*

The advantage of the path-based formulation is the ability to limit the paths to create a heuristic. Unlike the flow conservation model, the path-based formulation can be divided into $|\mathcal{P}|$ independent problems. Consider the initial path-based formulation in section 3.3.5; the problem can be divided into two steps. The first step is to solve the initial path-based formulation for each path independently and drop the path selection constraints. Each problem solved, with a single path, can either be infeasible, optimal or undetermined. Infeasible paths can be excluded from the original set of paths. From all the sub-problems solved to optimality[3], only the sub-problem with the best objective value needs to be stored. When the set of undetermined sub-problems is empty, the best solution is optimal to the original problem. When the undetermined subproblems are not empty, the sub-problems' best solution may not be optimal to the original problem. Instead, a new set of paths is created which include all the paths from the undetermined set and the best solution from the optimal subproblems, and a smaller MILP problem is solved. The single path sub-problem is to minimise

$$t_{|\mathcal{S}|-1} + \tau_{|\mathcal{S}|-1}, \tag{3.112}$$

subject to

$$\tau_{s-1} + t_{s-1} \leq t_s, \ \forall s \in \mathcal{S}\backslash\{0\}, \tag{3.113}$$

which warrants that each starting time depends on all the previous starting times and durations, τ_s depends on v_s, which should also be linearised. Add the linearisation constraints for velocities with

$$v_s = \sum_{x\in\mathcal{X}} v_{xs}\alpha_{xs}, \ \forall s \in \mathcal{S}, \tag{3.114}$$

where v_{xs} is the breakpoint value for segment s. To linearise durations add,

$$\tau_s = \sum_{x\in\mathcal{X}} \tau_{xs}\alpha_{xs}, \ \forall s \in \mathcal{S}, \tag{3.115}$$

with τ_{xs} the breakpoint value for x on segment s. Additional SOS2 constraints are needed to select a single line segment[4], which uses α_{xs} as a convex combination between two breakpoints, these constraints are usually omitted, but can be expressed as

$$\alpha_{xs} \leq h_{(x-1)s} + h_{xs}, \ \forall x \in \mathcal{X}, \ \forall s \in \mathcal{S}, \tag{3.116}$$

$$\sum_{x\in\mathcal{X}} \alpha_{xs} = 1, \ \forall s \in \mathcal{S}, \tag{3.117}$$

$$\sum_{x\in\mathcal{X}\backslash\{X\}} h_{xs} = 1, \ \forall s \in \mathcal{S}, \tag{3.118}$$

$$t_s \leq \sum_{y\in\mathcal{Y}\backslash\{Y\}} \kappa_{ys} t_{(y+1)s}, \qquad \forall s \in \mathcal{S}, \tag{3.119}$$

$$t_s \geq \sum_{y\in\mathcal{Y}\backslash\{Y\}} \kappa_{ys} t_{ys}, \qquad \forall s \in \mathcal{S}. \tag{3.120}$$

3 Sub-problems solved to optimality only indicates a feasible solution to the original problem.

4 Line segments refers to the linearised parts of SOS2 implementations and not the segment of the problem description which is in $\mathcal{S}$.

To allow only a single start time to be selected on each segment,

$$\sum_{y \in \mathcal{Y} \setminus \{Y\}} \kappa_{ys} = 1, \qquad \forall s \in \mathcal{S}. \tag{3.121}$$

The energy used, given the start time and velocity on a segment can be expressed as

$$e_s = \sum_{x \in \mathcal{X}} \alpha_{xs} e(x, y). \tag{3.122}$$

κ_{ys} can be related to the energy usage since it indicates a specified start time. Big M constraints are required to ensure that the energy usage on the selected start time becomes an equality,

$$e_s \leq \sum_{x \in \mathcal{X}} \alpha_{xs} e(x, y) + M(1 - \kappa_{ys}), \qquad \forall y \in \mathcal{Y}, \forall s \in \mathcal{S}, \tag{3.123}$$

$$e_s \geq \sum_{x \in \mathcal{X}} \alpha_{xs} e(x, y) - M(1 - \kappa_{ys}), \qquad \forall y \in \mathcal{Y}, \forall s \in \mathcal{S}. \tag{3.124}$$

The domains of the decision variables are

$$0 \leq \alpha_{xs} \leq 1, \ \forall x \in \mathcal{X}, \ \forall s \in \mathcal{S}, \tag{3.125}$$

$$h_{xs} \in \{0, 1\}, \ \forall x \in \mathcal{X}, \ \forall s \in \mathcal{S}, \tag{3.126}$$

$$\kappa_{ys} \in \{0, 1\}, \ \forall y \in \mathcal{Y}, \ \forall s \in \mathcal{S}, \tag{3.127}$$

$$e_s, t_s, v_s, \tau_s \geq 0, \ \forall s \in \mathcal{S}. \tag{3.128}$$

Algorithm 6 Pseudocode for solving independent sub-problems for the path-based model.

Input: $\mathcal{P}$ a set of paths from s to t.
Output: A single path p, or infeasible.

```
s_p ← solve_subproblem(p), ∀p ∈ P
j ← no solution
if s_p is undetermined or feasible with relative gap > 0, ∀p ∈ P then
    Q ← Q ⋃ {s_p}
end if
if s_p is optimal, ∀p ∈ P then
    j ← p if s_p is a better solution than s_j
end if
if j has solution and |Q| = 0 then
    return j
end if
if time-limit reached and j is solution then
    return j
else
    return no solution
end if
return solve(Q ⋃ {j})
```

Algorithm 6 provides an outline of the path-based model as a parallel problem. The time-limit determines the halting condition of Algorithm 6. The algorithm will stop prematurely only when all the subproblems are solved to optimality before the time-limit was reached, or there is no solution. The paths between two vertices can be exponential, given the number of vertices. In cases where an exponential number of subproblems need to be solved, speed-up can be negatively

impacted by limited computational resource availability or increased costs. Although the problem can be solved as independent subproblems does not mean it is practical. For dense graphs, the number of paths between the source and target node can be significant, leading to a large number of subproblems. Solving many subproblems in parallel may require expensive hardware with many processing cores. For example, a dataset with a thousand paths will already be bottlenecked with consumer hardware. This algorithm does not reduce the search space; it only solves the subproblems in parallel and is heuristic when a time-limit is set. A more reasonable approach is to use the path-based heuristic in conjunction with the flow conservation model.

3.5 MODEL IMPROVEMENTS

Building upon the previous models, charging stations may be added as well as the energy needed to accelerate from one speed to another. One approach to accommodate charging stations is to change the input data. Details on charging stations are given in Section 3.5.1. When considering acceleration in the models, adjustments to the formulation are needed due to the non-linear nature of acceleration.

3.5.1 *Charging stations*

Charging stations can be accommodated in the models described in Section 3.2, by representing them as artificial vertices within the input graphs. As an example, Figure 22a shows the addition of vertex v_3 to represent a charging station. Paths generated from the input graphs are directed path graphs and the charging station in Figure 22a will not generate a path containing v_3, since there is no way to define a directed path graph between v_1 and v_4 that contains distinct vertices and which contains arcs c_{23} and c_{32}. A way around this is to replace v_2 with two auxiliary vertices x, y and add an arc between them with a zero weight, Figure 22b illustrates this. One thing to note is that

$$w_{12} + w_{24} = w_{1x} + w_{xy} + w_{y5}. \tag{3.129}$$

Energy graphs need to be generated differently for charging station arcs. The energy graph generated at each charging station depends on the charging duration of a vehicle on that arc. Therefore a simple reciprocal function is used to represent charging stations,

$$f(t) = \frac{-k}{rt+1} + k, \tag{3.130}$$

where k is the capacity of the battery being charged and r the charging rate. This function can be swapped for any function specific to the charging station.

3.5.2 *Energy used to accelerate between segments*

The optimisation models in Section 3.3 do not account for the energy when accelerating or decelerating between segments/arcs. These additions require extra linearisation variables and constraints since the relationship between acceleration and speed is not linear. From equation (3.46) recall that

$$\boldsymbol{F_M} \cdot d\boldsymbol{x} = \boldsymbol{F_M} \cdot \boldsymbol{v}dt = m\boldsymbol{a} \cdot \boldsymbol{v}dt = m\frac{d\boldsymbol{v}}{dt} \cdot \boldsymbol{v}dt, \tag{3.131}$$

(a) Charging station arcs causing a cycle on a vertex. (b) Charging station arcs not causing a cycle on a vertex.

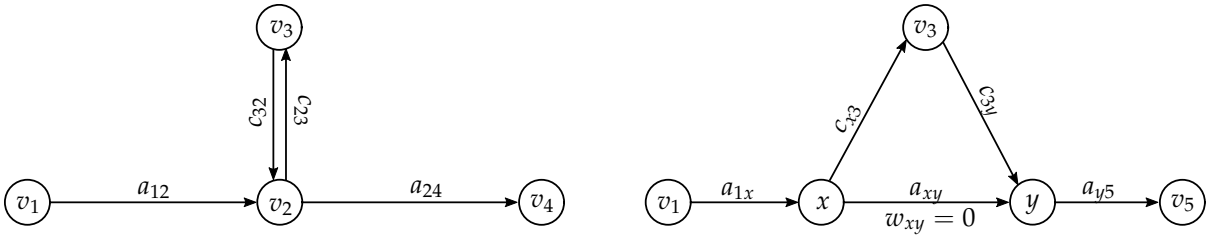


Figure 22: Representing charging stations in a graph.

for an infinitesimal displacement $d\boldsymbol{x}$ and -time interval dt. Apply the product rule, assuming m is constant,

$$md(\boldsymbol{v}\cdot\boldsymbol{v}) = m(d\boldsymbol{v}\cdot\boldsymbol{v} + \boldsymbol{v}\cdot d\boldsymbol{v}) = 2m(d\boldsymbol{v}\cdot\boldsymbol{v}) \implies m(d\boldsymbol{v}\cdot\boldsymbol{v}) = \frac{md(\boldsymbol{v}\cdot\boldsymbol{v})}{2}, \tag{3.132}$$

thus,

$$\boldsymbol{F_M}\cdot d\boldsymbol{x} = m\frac{d\boldsymbol{v}}{dt}\cdot\boldsymbol{v}dt = m(d\boldsymbol{v}\cdot\boldsymbol{v}) = \frac{m}{2}d(\boldsymbol{v}\cdot\boldsymbol{v}) \tag{3.133}$$

The work done between time t_1 and t_2 can be calculated from equation (3.131) as,

$$E_k = \int_{t_1}^{t_2}\boldsymbol{F_M}\cdot d\boldsymbol{x} = \int_{t_1}^{t_2} m\frac{d\boldsymbol{v}}{dt}\cdot\boldsymbol{v}dt. \tag{3.134}$$

It is evident that the work done depends on the initial and ending speed that relates to t_1 and t_2. Use v_1 as the speed at t_1 and v_2 as the speed at t_2, then,

$$E_k = \int_{\boldsymbol{v_1}}^{\boldsymbol{v_2}}\frac{m}{2}d(\boldsymbol{v}\cdot\boldsymbol{v}) = \frac{1}{2}m|\boldsymbol{v_2}|^2 - \frac{1}{2}m|\boldsymbol{v_1}|^2. \tag{3.135}$$

The difference in kinetic energy between segments s_1 and s_2 can be used to estimate the energy used to accelerate/decelerate from v_1 to v_2.

To incorporate equation (3.135) into the optimisation models, the energy graphs need to be changed. Previously, each energy graph estimated the energy used/gained on a segment with the start time and speed on a segment. For the path-based model constraint set (3.89) is changed to,

$$v_s^2 = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}}\beta_{xys}v_{xs}^2, \qquad \forall s\in\mathcal{S}(\rho), \rho\in\mathcal{P}, \tag{3.136}$$

for the triangle method and constraint set (3.23),

$$v_s^2 = \sum_{x\in\mathcal{X}} v_{xs}^2 a_{xs},\ \forall s\in\mathcal{S}(\rho),\ \forall\rho\in\mathcal{P}, \tag{3.137}$$

for the SOS2. The squared speed is used as a linear variable in the MILP models, since

$$v_s \geq 0. \tag{3.138}$$

Now add E_k to the energy used per segment, modifying constraint set (3.56) to,

$$e_s = \sum_{x\in\mathcal{X}} e_{xs}a_{xs} + \frac{1}{2}mv_s^2 - \frac{1}{2}mv_{(s-1)}^2,\ \forall s\in\mathcal{S}(\rho)/\{0\}, \rho\in\mathcal{P}, \tag{3.139}$$

$$e_s = \sum_{x\in\mathcal{X}} e_{xs}a_{xs} - \frac{1}{2}mv_s^2,\ s=0, s\in\mathcal{S}(\rho),\ \rho\in\mathcal{P}. \tag{3.140}$$

The same applies for the flow conservation model by replacing constraint set (3.38) with

$$v_a^2 = \sum_{x\in\mathcal{X}} v_{xa}^2\alpha_{xa},\ \forall a\in\mathcal{A}, \tag{3.141}$$

and constraint set (3.102) with

$$v_a^2 = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \beta_{xya} v_{xa}^2, \qquad \forall a \in \mathcal{A}. \tag{3.142}$$

Since incident arcs to each vertex are considered in the flow conservation model and E_k cannot be added to each constraint as with the path-based model, constraint set (3.62) and (3.63) are extended to accommodate the adjacent vertices as well. Modify the Big M constraints to account for both adjacent and incident arcs, which can be achieved by,

$$E_u \leq E_b + e_{bk} + \frac{1}{2} m v_{bu}^2 - \frac{1}{2} m v_{ug}^2 + M(2 - f_{bu} - f_{ug}),\ \forall b \in \gamma(u),\ g \in \sigma(u),\ \forall u \in \mathcal{V}, b \neq u, \tag{3.143}$$

$$E_u \geq E_b + e_{bu} + \frac{1}{2} m v_{bu}^2 - \frac{1}{2} m v_{ug}^2 - M(2 - f_{bu} - f_{ug}),\ \forall b \in \gamma(u),\ g \in \sigma(u)\ \forall u \in \mathcal{V}, b \neq u, \tag{3.144}$$

when an incident arc and an adjacent arc is active, a single constraint

$$E_u = E_b + e_{bu} + \frac{1}{2} m v_{bu}^2 - \frac{1}{2} m v_{ug}^2,\ b \in \gamma(u),\ g \in \sigma(u), b \neq u \tag{3.145}$$

is active per vertex $u \in \mathcal{V}$.

The drawback to using v^2 is the general loss of accuracy. Finer granularity of data can improve accuracy at the cost of a larger input set, computation time and memory usage.

3.6 SUMMARY

The input data was described as graphs; every vertex corresponds to a GIS location and the arc between vertices an approximation of the physical road segment. The data needed to be discrete before formulating the problem as a MILP problem and different approaches were provided as alternatives. Weather predictions are collected per segment/arc and is added as meta-data to each arc. This chapter introduced two different MILP models to address the single-vehicle routing problem with a solar panel. The basic model was described initially, and complexity was systematically added to the models to make the thought process clear. The BEV physics were described from a free body diagram and an expression to calculate the energy on a slope with a constant velocity was determined. Practical solar irradiation calculations were described with illustrations which were used in the simulations. The vector calculations needed for the wind speed and wind direction were described in detail. Segment energy graph generation used wind calculations and solar irradiation calculations to simulate each segment's vehicle energy usage. Finally, model improvements were described; this included charging stations and acceleration energy between segments. In the next chapter, the results for models and simulations are discussed.

4

RESULTS AND ANALYSIS

4.1 INTRODUCTION

In this chapter, the results of employing the proposed models are analysed and discussed. Vehicle parameters provided in Table 1 are used to generate results in this chapter. First, descriptions of the naming conventions are given. The initial prefix, either FC or P, indicates the primary model used, FC for the flow conservation model and P for the path-based model. The mid-part of the model name shows the energy graph method used in the model, TRI or BLK. TRI is the triangle method that uses Special Ordered Set of Type 3 (SOS3) to model energy graphs and BLK is a block method that uses Special Ordered Set of Type 2 (SOS2) to model energy graphs. Feasible solutions can be used to improved the total runtime and solution quality when used as a warm-start. These feasible solutions are obtained from heuristic algorithms. Finally, the suffix is an indicator for either warm-starts or using heuristic implementations. FC and P are used if no warm-start or heuristic was used, Kn is used to show a heuristic implementation and FKn for using warm-start solutions. Kn is for the n shortest path heuristic, and FKn warm-starts the model with the solution of Kn. Peak memory and time shown in tables are the measurement using `/usr/bin/time -v`[1] and not the performance measure of the solver, which is the actual total memory usage and computation time reported by the operating system [75]. A visualisation of the datasets used can be found in Appendix B.

The results were generated with an AMD RYZEN™ 3900XT@3.8GHz with 64 GB system memory and the models were implemented with IBM ILOG CPLEX 12.8[2].

Multiple datasets with different properties are used to compare the model implementations. The datasets represent real locations across the world, with different weather conditions and geography. The same time-of-year was used to generate weather data for the locations involved. Weather data used in these optimisation runs were obtained using `solarpy`[3] for clear sky radiation and `Weatherbit` for weather forecasts. Table 2 shows the number of arcs and vertices each dataset has, along with the source ID, target ID and the number of paths that exist between the source and the target.

FC-TRI-FC is the flow conservation model with SOS3 energy graphs. P-TRI-P is the path-based model with SOS3 energy graph constraints, where all possible paths between the source and destination vertices are generated. P-TRI-Kn is the path-based model, with SOS3 energy graph constraints, limiting the number of paths generated to n, making P-TRI-Kn a heuristic algorithm. The proposed optimisation models minimise total travel time of a Battery Electric Vehicle (BEV) with an energy budget, known weather conditions and twelve velocity knots ranging from $0\,\mathrm{m\,s^{-1}}$ to $33.33\,\mathrm{m\,s^{-1}}$. The BEV has a solar panel attached unless otherwise stated.

When considering flow conservation models with warm-starts, it is crucial to know that the preprocessing computation time is not accounted for in the result tables. The intention is to see if

1 `/usr/bin/time` should not be confused with `time`, for more details visit `https://man7.org/linux/man-pages/man1/time.1.html`

2 The application was compiled with `gcc` 10.2.0

3 `solarpy` is a python library to approximate solar irradiation on a plane for a given timestamp.

Table 2: Details of datasets used to generate results.

General Location	Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{A}\rvert$	Source ID	Target ID	Number of paths
Freestate, South Africa	*smalloo1*	17	17	1	16	1
North-West, South Africa	*smalloo2*	25	24	17	34	2
Lisbon, Portugal	*smalloo3*	61	97	44	60	8
Northern Cape, South Africa	*smalloo4*	71	97	249	168	132
Easter Cape, South Africa	*smalloo5*	14	18	262	256	6
Mato Grosso do Sul, Brazil	*smalloo6*	33	49	275	291	16
Missouri, United States	*smalloo7*	18	24	300	309	9
Saskatchewan, Canada	*smalloo8*	28	56	310	319	64
Agadez, Niger	*smalloo9*	17	36	349	351	8
Chongqing, China	*smallo10*	37	93	382	355	440
Lesotho	*mediumoo1*	432	437	473	570	3
Port Harcourt, Nigeria	*mediumoo2*	185	186	850	861	4
Madagascar	*mediumoo3*	225	444	1033	1255	30
Northern Cape, South Africa	*mediumoo4*	30	108	1278	1258	187478

partial warm-start solutions can be beneficial for the flow conservation models. The total runtime of a flow conservation model with warm-starts, including the preprocessing, is the sum of the path-based heuristic and the flow conservation model runtime, e.g. the total runtime of F-TRI-FK1 is runtime(P-TRI-K1) + runtime(F-TRI-FK1).

The next two sections' results of various datasets with a solver time-limit of six minutes are given and discussed, followed by results of datasets that did not solve within six minutes and for which a time-limit of one hour is applied.

4.2 RESULTS OF VARIOUS DATASETS

In this section, the triangle method is used to model energy graphs for all the different implementations. Although the modelling of energy graphs with the triangle method is more elegant than the block method, solvers do not have the same level of support for SOS3 as they have with SOS2. Two sets of results are calculated, one with the energy used between acceleration accounted for between segments, and one without acceleration. The results are only shown for datasets where at least one feasible solution was found.

4.2.1 *Results of models with triangle-based energy graphs and a solver time-limit of six minutes*

Result tables to follow show the model name, total runtime, the peak memory usage, the best objective value obtained and the relative gap when a solar panel is attached to the BEV. When the relative gap is marked with $*$, it indicates the path-based heuristic was used, and the relative gap does not reflect the gap of the exact model but instead gives some indication of completion of its own search space. Table 3 shows the results for the datasets that obtained a feasible solution within a six-minute solver time-limit. Some datasets, such as *smalloo4* are excluded from Table 3 because no solution was found.

Table 3 shows that both the path-based and flow conservation models result in an optimal value of 2296.17 seconds under 4 seconds of total computation time for *smalloo1*. The heuristic implementations also obtain the optimal solution because the shortest path is the optimal solution.

Table 3: Results for models with triangle-based energy graphs and a solver time-limit of six minutes for *smalloo1*, *smalloo2*, *smalloo3* and *smalloo5*.

Dataset	Model	Runtime	Peak Memory (KB)	Objective Value	Relative Gap
smalloo1	FC-TRI-FC	3.71379	872472	2296.17	0
	FC-TRI-FK1	2.86928	524432	2296.17	0
	P-TRI-K1	3.64478	739984	2296.17	*0
	P-TRI-P	3.7275	761564	2296.17	0
smalloo2	FC-TRI-FC	9.60379	1247668	1839.95	0
	FC-TRI-FK1	8.90337	905832	1839.95	0
	FC-TRI-FK2	6.47111	942500	1839.95	0
	P-TRI-K1	4.35747	931096	1877.45	*0
	P-TRI-K2	56.1519	2490128	1839.95	*0
	P-TRI-P	58.3981	2407928	1839.95	0
smalloo3	FC-TRI-FC	194.606	8211420	662.058	0
	FC-TRI-FK1	248.883	7609988	662.058	0
	FC-TRI-FK2	301.196	5631204	662.058	0
	FC-TRI-FK3	300.099	5762752	662.058	0
	P-TRI-K1	3.49898	882632	662.058	*0
	P-TRI-K2	43.5951	1759128	662.058	*0
	P-TRI-K3	356.724	3821868	662.058	0
smalloo5	FC-TRI-FC	360.526	1421656	17396	0.463426
	FC-TRI-FK1	366.505	2330128	14142.6	0.318497
	FC-TRI-FK2	382.348	1551168	15605.8	0.301825
	FC-TRI-FK3	371.962	1732992	15307.6	0.384945
	P-TRI-K1	360.819	901524	14142.6	*0.170615
	P-TRI-K2	360.338	3089012	18045.5	*0.450088
	P-TRI-K3	360.493	3295772	30211.9	*0.682298
	P-TRI-P	360.494	3256144	210100	0.955572

The flow conservation model uses slightly more memory than the path-based model. The *smalloo1* dataset only has 14 vertices and 18 arcs, which is a relatively small dataset, and the results do not reflect the performance of the model implementations, but it is useful for implementation verification with visual inspection.

The *smalloo2* dataset has 25 vertices, and 25 arcs, which is also a small dataset. The flow conservation model performs significantly better than the path-based model. P-TRI-K1 has an optimal value of 1877.45, which is higher since it is a heuristic solution where only one path was used. The second shortest path is the optimal path, *smalloo2* only have two paths. The runtime of the flow conservation model showed some improvement when the heuristic solution was used as a warm-start.

Dataset *smalloo3* has 61 vertices and 97 arcs and the arc distances are relatively small; most arcs are shorter than 500 m. The dataset has multiple possible paths, but once again, the shortest path is the optimal path. The path-based heuristics are extremely fast, since multiple arcs are eliminated, which makes the Linear Programming (LP) significantly smaller. The SOS3 associated with each arc increases the model size as the number of arcs increase. There can be an exponential

number of binary variables for the path-based model when the number of paths is exponential; this is the case for dense graphs. After the time-limit of 360 seconds, CPLEX did not supply a single solution for the path-based model. The heuristics found a solution, but it is only a coincidence that the shortest path is optimal. The results can be verified with the optimal solution of the flow conservation model.

Dataset *small005* dataset has 14 vertices and 18 arcs with an average edge degree of 1.29. Dataset *small005* is a more densely connected graph than the previous datasets that were solved. The arcs span distances ranging up to 11 km which is considerably more than the previous datasets. All the models reached the time-limit of 360 seconds. Once more, the shortest path heuristic provided the best-known result but did not solve to optimality. Using the path-based heuristic solution as a warm-start significantly affected the solution quality provided by the flow conservation model. The path-base models had a higher memory usage than the flow conservation model.

Unfortunately, the model implementation did not find a feasible solution within the time-limit of 360 seconds by using SOS3 constraints for the energy graphs for all the datasets. Only *small001*, *small002*, *small003* and *small005* had feasible solutions. SOS3 is not a standard implementation in solvers, and there are no speed-ups to using it. An alternative approach is to use SOS2 for the energy graphs—the results for when energy graphs use SOS2 are discussed in Section 4.2.3.

4.2.2 *Results of models with triangle-based energy graphs accounting for acceleration and a solver time-limit of six minutes*

In this section, the energy used to accelerate from one velocity to another between segments are included in the Mixed Integer Linear Programming (MILP) models. Only the results for datasets where at least one feasible solution was found are shown in Table 4.

Table 4 shows the results for the *small001*, *small002*, *small003* and *small005*, where a SOS3 approach was used for modelling the energy graph and with acceleration included in the models. The results are similar to what is provided in Table 3, since the battery is not drained when driving at maximum speed throughout the whole route.

The flow conservation model performs better than the path-based model with respect to computation time and memory usage on *small002*. With the k-shortest path heuristic, when only a single path was selected, it was 24.45% faster and used 20.76% less memory, but had an objective value 2.04% worse than the optimal solution of the original problem. Using the result of the k-shortest path heuristic as a warm-start for the flow conservation model increased the overall run-time.

For *small003*, the path-based model did not solve within the given solver time-limit of 360 seconds and had a peak memory usage of 11.6 GB. The k-shortest path heuristic limited to only the shortest path had an extremely fast execution time. The shortest path was the optimal path, but it is not always the case. In cases of restricted speed on a segment of the shortest path or the influence of weather in such a way that makes another path more attractive, may lead to optimal solutions that are not the shortest path. When more paths were considered, the run-time increased significantly. With $k = 3$ the heuristic did not solve to optimality[4].

4 Optimality with the heuristic refers to the best possible solution the heuristic can provide with limited paths, not optimality of the original problem that includes the complete search space.

Table 4: Results for models with triangle-based energy graphs which accounts for acceleration and six-minute solver time-limit using *smalloo1*, *smalloo2*, *smalloo3* and *smalloo5*.

Dataset	Model	Runtime	Peak Memory (KB)	Objective Value	Relative Gap
smalloo1	FC-TRI-FC	3.72239	832940	2296.17	0
	FC-TRI-FK1	3.50122	819936	2296.17	0
	P-TRI-K1	3.46102	815072	2296.17	*0
	P-TRI-P	3.30325	811460	2296.17	0
smalloo2	FC-TRI-FC	6.16336	1209060	1839.95	0
	FC-TRI-FK1	12.4038	1225116	1839.95	0
	FC-TRI-FK2	7.47612	1196064	1839.95	0
	P-TRI-K1	4.65639	958196	1877.45	*0
	P-TRI-K2	59.7702	2546964	1839.95	*0
	P-TRI-P	59.663	2518856	1839.95	0
smalloo3	FC-TRI-FC	94.1978	5497064	662.058	0
	FC-TRI-FK1	205.091	6124208	662.058	0
	FC-TRI-FK2	97.5271	5696020	662.058	0
	FC-TRI-FK3	77.4308	5504292	662.058	0
	P-TRI-K1	3.51932	861464	662.058	*0
	P-TRI-K2	211.253	2807076	662.058	*0
	P-TRI-K3	361.326	3625944	673.79	*0.856972
smalloo5	FC-TRI-FC	360.341	1820360	27444.3	0.896535
	FC-TRI-FK1	387.972	3022124	9204.14	0.203059
	FC-TRI-FK2	370.755	2106076	9272.27	0.237389
	FC-TRI-FK3	374.214	1757176	11061.5	0.526702
	P-TRI-K1	360.12	1226196	15763.2	*0.400567
	P-TRI-K2	366.237	3934336	20326.2	*0.594929
	P-TRI-K3	360.285	2143128	45340.9	*0.832364

The source node of *smalloo5* is in Swellendam, Western Cape, South Africa and the destination node in Guguletu, Western Cape, South Africa. Not a single model solved to optimality, but the flow conservation model with warm-start solutions had the best objective values and relative gaps. Considering only the flow conservation model with warm-starts is not sufficient since these models' actual total run-time is the sum of the path-based heuristics and the flow conservation model. Thus the actual run-time of FC-TRI-FK1 is 748.092 seconds, similar for FC-TRI-FK2 and FC-TRI-FK3. A fair comparison would be the same total run-time which includes the heuristic algorithms. The main reason for the separate total run-time is to compare the effect of a heuristic solution as a warm-start to one without a warm-start; this may indicate whether a problem specific heuristic will be beneficial to the model implementation.

The six-minute time-limit is chosen for practicality. The route optimisation should be relatively fast since the route calculation should happen moments before driving. It is clear that the models do not scale well with SOS3 energy graphs, the results of energy graphs that use SOS2 follows in the next subsection.

4.2.3 *Results of various datasets, with rectangular-based energy graphs and a solver time-limit of six minutes*

Table 5: Results for models with rectangular-based energy graphs and six-minute solver time-limit using *small001*, *small002*, *small003*, *small005* and *small009*.

Dataset	Model	Runtime	Peak Memory (KB)	Objective Value	Relative Gap
small001	FC-BLK-FC	0.293511	127672	2296.17	0
	FC-BLK-FK1	0.194257	68468	2296.17	0
	P-BLK-K1	0.246475	99404	2296.17	*0
	P-BLK-P	0.25291	100120	2296.17	0
small002	FC-BLK-FC	0.473313	167592	1839.95	0
	FC-BLK-FK1	0.307103	82532	1839.95	0
	FC-BLK-FK2	0.286997	89220	1839.95	0
	FC-BLK-FK3	0.280605	89152	1839.95	0
	P-BLK-K1	0.335495	137332	1877.45	*0
	P-BLK-K2	1.08827	226832	1839.95	*0
	P-BLK-K3	1.11511	229500	1839.95	*0
	P-BLK-P	1.13672	229972	1839.95	0
small003	FC-BLK-FC	5.13938	592464	662.058	0
	FC-BLK-FK1	4.5548	473708	662.058	0
	FC-BLK-FK2	4.70369	473680	662.058	0
	FC-BLK-FK3	4.32953	475096	662.058	0
	P-BLK-K1	0.367286	146224	662.058	*0
	P-BLK-K2	1.01974	222296	662.058	*0
	P-BLK-K3	3.3168	315396	662.058	*0
	P-BLK-P	30.7036	1491840	662.058	0
small005	FC-BLK-FC	1.37122	165876	11866.6	0
	FC-BLK-FK1	0.898492	104496	11866.6	0
	FC-BLK-FK2	0.924648	104224	11866.6	0
	FC-BLK-FK3	0.886929	103384	11866.6	0
	P-BLK-K1	0.410791	92984	11866.6	*0
	P-BLK-K2	2.0587	171228	11866.6	*0
	P-BLK-K3	14.2027	259136	11866.6	*2.00888e-05
	P-BLK-P	66.8271	727668	11866.6	0
small009	FC-BLK-FC	360.046	827568	368599	0.295897
	FC-BLK-FK1	54.0387	496532	348030	0
	FC-BLK-FK2	54.2967	491376	348030	0
	P-BLK-K1	49.792	285164	348030	*9.84764e-05
	P-BLK-K2	360.03	2059528	348030	*0.402317

Table 5 shows the results for *small001*, *small002*, *small003*, *small005* and *small009*, with a time-limit of six minutes, SOS2 for the energy graphs, a solar panel attached to the BEV and not accounting for acceleration. The rectangular-based energy graphs have a clear computational advantage over triangle-based energy graphs. The flow conservation model is more than 12 times faster, and the path-based model more than 14 times faster when using SOS2 over SOS3 for the energy

graphs for *smalloo1*. The peak memory usage of all models is also much lower when using rectangular-based energy graphs over triangle-based energy graphs. For *smalloo1* the error on the energy consumed is neglectable, but may vary depending on the size and complexity of the dataset. The energy errors will be discussed at a later stage in this chapter.

Compared to the same setup but with triangle-based energy graphs, the models with rectangular-based energy graphs solve significantly faster than triangle-based energy. The *smalloo2* dataset with triangle-based energy graphs using the flow conservation model solved to optimality in 9.6 seconds, where the flow conservation model with rectangular-based energy graphs solved to optimality in 0.47 seconds. The peak memory usage of the flow conservation model with rectangular energy graphs only used 13.4% of the peak memory the flow conservation model with triangle-based energy graphs used. These speed-ups and lower peak memory usages apply to both the flow conservation and path-based models. The objective values are the same, for both implementations, but there may be some variations in the total energy consumed, which is addressed later in this chapter.

With *smalloo3* the flow conservation and path-based model solved to optimality. The path-based model with SOS2 energy graphs had a total run-time of 30.70 seconds, where the path-based model with SOS3 energy graphs did not find a solution within the solver time-limit of six minutes. The flow conservation model with SOS2 energy graphs had a total runtime of 5.14 seconds, where the flow conservation model with SOS3 energy graphs had a runtime of 194.61 seconds, which is a 3686% increase in runtime.

The flow conservation model and path-based model solved to optimality with *smalloo5* where the SOS3 energy graph counterpart had a relative gap greater than zero. The best solution obtained from all the models with triangle-based energy graphs was the flow conservation model with the shortest path as a warm-start, which yielded an objective value of 14142.6 seconds. The flow conservation model with SOS2 energy graphs had a total runtime of 1.37 seconds and had an optimal solution of 11866.6 seconds. Which is 16.09% lower than the best solution with SOS3 energy graphs. The peak memory usage of all the models with SOS2 energy graphs where also lower than the SOS3 energy graph counterparts.

Dataset *smalloo9* did not solve to optimality with the time-limit of six minutes when using the flow conservation model without warm-start solutions. When solutions from the k-shortest path heuristics were used as a warm-start for the flow conservation model, it had a significant impact. The P-BLK-K1 had a total run-time of 49.792 seconds and the FC-BLK-FK1 had a run-time of 54.0387 which is 103.8307 seconds total; this is a faster run-time than without a warm-start, also without the warm-start the flow conservation model did not solve to optimality.

Table 6 shows the results for *mediumoo2* and *mediumoo4*, with a time-limit of six minutes, SOS2 for the energy graphs, a solar panel attached to the BEV and not accounting for acceleration. For mediumoo2, the shortest path was once again the optimal path. When using the shortest path solution as a warm-start to the flow conservation model the run-time decreased by 24.43%. The flow conservation model had a 86.07% lower run-time than the path-based model that considers all possible paths.

Using warm-starts for *mediumoo2* had a negative impact on the total run-time of the flow conservation model. The path-based model that considers all possible paths did not find a solution within the solver time-limit of six minutes. The shortest path was the optimal path and

Table 6: Results for models with rectangular-based energy graphs and six-minute solver time-limit using *mediumoo2* and *mediumoo4*.

Dataset	Model	Runtime	Peak Memory (KB)	Objective Value	Relative Gap
mediumoo2	FC-BLK-FC	19.727	1140256	1920.7	0
	FC-BLK-FK1	14.9069	1004740	1920.7	3.70071e-08
	FC-BLK-FK2	15.3368	1002704	1920.7	3.70071e-08
	FC-BLK-FK3	14.8211	1003648	1920.7	3.70071e-08
	P-BLK-K1	1.86063	369716	1920.7	*0
	P-BLK-K2	12.4576	938332	1920.7	*0
	P-BLK-K3	17.9588	1403876	1920.7	*0
	P-BLK-P	141.619	2189992	1920.7	0
mediumoo4	FC-BLK-FC	13.9192	802512	53733.1	0
	FC-BLK-FK1	20.8126	832776	53733.1	0
	FC-BLK-FK2	20.1176	841528	53733.1	0
	FC-BLK-FK3	20.0532	830600	53733.1	0
	P-BLK-K1	3.83809	131420	53733.1	*0
	P-BLK-K2	31.5478	315012	53733.1	*0
	P-BLK-K3	360.041	1053992	53733.1	0.377986

the path-based heuristic had a total run-time of 3.84 seconds, which can be verified with the flow conservation model.

4.2.4 *Results of various datasets, with rectangular-based energy graphs accounting for acceleration and a solver time-limit of six minutes*

Table 7 shows the results for *smalloo1*, *smalloo2*, *smalloo3*, *smalloo5* and *smalloo9* with SOS2 energy graphs, accounting for BEV acceleration energy and a solver time-limit of six minutes. The runtime is more than ten times faster than the SOS3 energy graph counterpart in Table 4.

All the variations of models solved to optimality for *smalloo1* with an objective function of 2296.17 seconds. The peak memory usage is also significantly lower than the SOS3 energy graph counterparts.

With *smalloo2* the path-based model with SOS2 energy graphs was 54.57 times faster than the SOS3 energy graph counterpart and a 91.25% decrease in peak memory usage. Similar performance gains are applicable with the other models, for instance models with SOS2 energy graphs are by average 41.99 times faster than the SOS3 energy graph alternative. The objective values and selected routes of *smalloo2* are the same as in Table 4.

For *smalloo3* the flow conservation model had a runtime of 6.08 seconds which is a 85.78% decrease in runtime of the path-based model. The peak memory usage of the path-based model is 149.30% more than the flow conservation model. The path-based heuristic with $k = 1$ obtained the optimal solution, since the shortest path is the optimal path. Using the results of the path-based heuristic with only the shortest path as a warm-start improved the total runtime of the flow conservation model by 6.82%.

The source node of smalloo5 started in Swellendam, Western Cape, South Africa and the destination node ended in Guguletu, Western Cape, South Africa. The flow conservation model was solved to optimality under one second, where the path-based model had a total runtime

Table 7: Results for models with rectangular-based energy graphs which accounts for acceleration and six-minute solver time-limit using *smalloo1*, *smalloo2*, *smalloo3*, *smalloo5* and *smalloo9*.

Dataset	Model	Runtime	Peak Memory (KB)	Objective Value	Relative Gap
smalloo1	FC-BLK-FC	0.298111	113256	2296.17	0
	FC-BLK-FK1	0.200373	71412	2296.17	0
	P-BLK-K1	0.284626	114116	2296.17	*0
	P-BLK-P	0.277422	109124	2296.17	0
smalloo2	FC-BLK-FC	0.515547	167760	1839.95	0
	FC-BLK-FK1	0.283916	83436	1839.95	0
	FC-BLK-FK2	0.315634	81443	1839.95	0
	*P-BLK-K1	0.367422	121020	1877.45	*0
	*P-BLK-K2	0.567422	145320	1839.95	*0
	P-BLK-P	1.09316	220256	1839.95	0
smalloo3	FC-BLK-FC	6.08045	638736	662.058	0
	FC-BLK-FK1	5.21112	619448	662.058	0
	FC-BLK-FK2	5.31727	617600	662.058	0
	FC-BLK-FK3	5.46467	613744	662.058	0
	P-BLK-K1	0.454383	141964	662.058	*0
	P-BLK-K2	1.06796	229064	662.058	*0
	P-BLK-K3	3.2346	332688	662.058	*0
	P-BLK-P	42.7505	1592340	662.058	0
smalloo5	FC-BLK-FC	0.859285	159960	8577.69	0
	FC-BLK-FK1	0.868431	152296	8577.69	0
	FC-BLK-FK2	0.88456	152296	8577.69	0
	FC-BLK-FK3	0.838883	154952	8577.69	0
	P-BLK-K1	0.340315	93580	11833.1	*8.97253e-05
	P-BLK-K2	1.71402	174188	11832.1	*0
	P-BLK-K3	11.2351	268264	11832.1	*0
	P-BLK-P	23.3242	516636	8577.69	0
smalloo9	FC-BLK-FC	61.6446	555520	336743	0
	FC-BLK-FK1	9.6654	291200	336743	0
	FC-BLK-FK2	9.28272	298548	336743	0
	FC-BLK-FK3	364.849	1182728	358310	0.153328
	P-BLK-K1	82.3564	276448	348029	*9.87031e-05
	P-BLK-K2	360.027	1676800	348029	*0.507135
	P-BLK-K3	363.301	5287536	368573	*0.550997

of 23.32 seconds. With SOS3 energy graphs, not a single model solved to optimality, and the path-based model using all paths did to get a single solution. The memory usage of the SOS3 energy graph counterpart also had a significantly higher peak memory usage for all the models. The SOS2 for energy graphs has a clear advantage over the SOS3 energy graphs. The k-shortest path heuristic yielded the same objective value for K_1, K_2 and K_3, which was the shortest path. The average arc distance is 35 km, which is significantly longer than the other *small* datasets. The *small* datasets that solved quickly in most cases chose the maximum speed, with the shortest

route, which is optimal when the battery capacity is large, and the routes are short. It is not the case for the *smalloo5* datasets.

Dataset *smalloo9* has 17 vertices and 36 arcs with the source node in Mutum, Nigeria and the destination node in Bordj Badji Mokhtar, Algeria which is more than 2000 km apart. The flow conservation model obtained the optimal solution in 61.64 seconds, where the warm-start solutions significantly increased the total runtime when K_1 and K_2 were used. When K_3 was used, it had a negative impact on the total run-time and peak memory usage of the flow conservation model, since the heuristic with K_3 did not solve to optimality and had a solution with a worse objective value than for K_1 and K_2. The k-shortest path heuristics had a longer run-time than the flow conservation model. The SOS3 energy graph flow conservation and path-based counterpart did not obtain a single solution for this dataset.

Table 8: Results for models with rectangular-based energy graphs which accounts for acceleration and six-minute solver time-limit using *mediumoo2* and *mediumoo4*.

Dataset	Model	Runtime	Peak Memory (KB)	Objective Value	Relative Gap
mediumoo2	FC-BLK-FC	26.3551	1218260	1900.93	0
	FC-BLK-FK1	13.3378	988536	1900.93	0
	FC-BLK-FK2	13.2547	988584	1900.93	0
	FC-BLK-FK3	13.1571	993020	1900.93	0
	P-BLK-K1	2.12822	350568	1900.93	*0
	P-BLK-K2	14.5493	1011700	1900.93	*0
	P-BLK-K3	32.21	1817032	1900.93	*0
	P-BLK-P	82.3644	2327904	1900.93	0
mediumoo4	FC-BLK-FC	360.131	2650352	52896.5	0.946309
	FC-BLK-FK1	362.422	2337448	48169.6	0.895698
	FC-BLK-FK2	362.373	2323368	48169.6	0.895698
	FC-BLK-FK3	362.368	2311668	48169.6	0.895716
	P-BLK-K1	3.64261	133912	53731.9	*0
	P-BLK-K2	53.9718	893412	53731.9	*0
	P-BLK-K3	360.034	1488564	53731.9	*0.38376

Table 8 shows the results for *mediumoo2* and *mediumoo4* with BEV acceleration accounted for and a six-minute solver time-limit. Dataset *mediumoo2* has 185 vertices and 186 arcs and the edge degree is close to one; there are not many paths to explore between the source and target nodes. The start node is in Aba, Nigeria and the destination node in Umuahia, Nigeria. The road limits were set to 120 km per hour, which is not the actual road limits, and the driving time is estimated at 1900.93 seconds. The flow conservation model solved in 26.3551 seconds which is 68% decrease in total runtime compared to the path-based model. The path-based heuristic when only the shortest path was considered solved in 2.13 seconds and improved the flow conservation model's total runtime by 41.32%; this is likely because the shortest path was the optimal path.

Dataset *mediumoo4* has 30 vertices and 108 arcs where the edge degree of the dataset is more than three. The vehicle is allowed to drive 120 km per hour and uses undocumented roads. The starting node is in Keimoes, Northern Cape and the destination node is in West Coast DC, Western Cape. The models struggled to reduce the relative gap within six minutes. The path-based heuristic with K_1 solved in 3.64 seconds. When the K_1, K_2 and K_3 heuristic solutions were used as

a warm-start for the flow conservation model, the objective value obtained was better and the relative gap lower. However, the actual total runtime is the sum of the heuristic and the flow conservation model; but this still shows that the shortest path heuristic solution may improve solution quality.

The six-minute time-limit was chosen for practicality. The route optimisation should be relatively fast since the route calculation should be able to happen dynamically. When the model use SOS2 for energy graphs, there was a significant total runtime and memory improvement for all models. The models solve relatively fast when the total travel distance of the BEV is short. The total solving time increases when the knots with maximum velocity cannot be chosen, which is when the battery capacity is not enough to travel the shortest route at maximum speed; this is when the arc distances are more than 10 km long. The path-based model performs better when the total number of paths are lower, and overall it seems that the shortest path heuristic benefits the flow conservation model. Not all the datasets solved; more datasets were solved with the rectangular-based energy graph implementation than with the triangle-based energy graph counterpart.

Overall the optimal route is the fastest route when the battery is not depleted when driving at maximum speed, which was the case for many datasets. In the next section, solving the remaining datasets with a solver time-limit of one hour is presented.

4.2.5 *Results of various datasets that were not solved within six minutes*

The tables in this section describe the results where at least one solution within one-hour was found. There is an indication of whether acceleration was accounted for and if the BEV had a solar panel attached. Whenever a model did not find a solution, it will be indicated in the objective value field as either memory-limit or time-limit reached.

Table 9 shows the results of *small004* with a one-hour solver time-limit. In most cases, the time-limit was reached without a solution. None of the models that use the triangle method for energy graphs found a solution within one-hour. The only solution found within time was the shortest path heuristic with the block method for energy graphs.

Table 10 shows the results of *small006* with a one-hour solver time-limit. The models with triangle-based energy graphs did not find a solution within one-hour. For *small006*, the acceleration did not play a significant role in the solutions found. The shortest path heuristic was the only model that did not run out of time or memory.

Table 11 shows the results of *small007*, with a one-hour time-limit. The shortest path heuristic produced a solution after one hour, but did not solve to optimality for any combination of parameters. The flow conservation model was numerically unstable when a solar panel was attached to the vehicle. The model's numerical instability is caused by the chosen value of M.
Table 12 shows the results of *medium001* with a one-hour solver time-limit. The k-shortest path heuristic shows there is no solution for the shortest path, with or without a solar panel attached. The k-shortest path found a solution when using the second shortest path with a solar panel and block-based energy graphs. When the energy used to accelerate the BEV from one speed to another was accounted for, the objective value was 105,132 seconds with a 90.14% relative gap; this is for a heuristic, so the gap does not reflect bounds for the exact models. Similar results were obtained when not accounting for acceleration energy.

Table 13 shows the results of *medium004* with a one-hour solver time-limit. The partial warm-start solutions did not improve the solution quality of the flow conservation model for *medium004*.

Table 9: Results for *smalloo4* with a one-hour solver time-limit.

Solar Panel	Acceleration	Method	Runtime (s)	Peak Memory (KB)	Objective Value(s)	Relative Gap
Yes	No	FC-BLK-FC	>1h	1503792	Time-limit	N/A
		P-BLK-K1	1959.25	923448	113143	*0
		P-BLK-K2	>1h	26630612	117332	*0.838049
		P-BLK-K3	>1h	>32 GB	112527	*0.83774
		FC-BLK-FK1	>1h	1282440	Time-limit	N/A
		FC-BLK-FK2	>1h	1284448	Time-limit	N/A
		FC-BLK-FK3	>1h	1282596	Time-limit	N/A
		P-BLK-P	>1h	23554016	Time-limit	N/A
Yes	Yes	FC-BLK-FC	>1h	1492664	Time-limit	N/A
		P-BLK-K1	1918.13	920740	53766.3	0
		P-BLK-K2	>1h	27542392	117332	*0.838049
		P-BLK-K3	>1h	31996104	112527	*0.83774
		FC-BLK-FK1	>1h	1273068	Time-limit	N/A
		FC-BLK-FK2	>1h	1270012	Time-limit	N/A
		FC-BLK-FK3	>1h	1272100	Time-limit	N/A
		P-BLK-P	>1h	23507760	Time-limit	N/A
No	Yes	FC-BLK-FC	18.2456	885210	No Solution	N/A
		P-BLK-K1	>1h	2437312	No solution	N/A
		P-BLK-K2	>1h	2814336	No solution	N/A
		P-BLK-K3	>1h	3094256	No solution	N/A
		P-BLK-P	>1h	14157321	Time-limit	N/A
Yes	No	FC-TRI-FC	>1h	12978440	Time-limit	N/A
		P-TRI-K1	>1h	4119420	Time-limit	N/A
		P-TRI-K2	>1h	17643660	Time-limit	N/A
		P-TRI-K3	>1h	5148776	Time-limit	N/A
		P-TRI-P	526.58	>32GB	Memory-limit	N/A
Yes	Yes	FC-TRI-FC	>1h	12990720	Time-limit	N/A
		P-TRI-K1	>1h	4170856	Time-limit	N/A
		P-TRI-K2	>1h	17631524	Time-limit	N/A
		P-TRI-K3	>1h	5297588	Time-limit	N/A
		P-TRI-P	531.52	>32GB	Memory-limit	N/A
No	Yes	FC-TRI-FC	19.1769	861284	No solution	N/A
		P-TRI-K1	>1h	3039308	No solution	N/A
		P-TRI-K2	>1h	1747796	No solution	N/A
		P-TRI-K3	>1h	994256	No solution	N/A
		P-TRI-P	>1h	15567220	Time-limit	N/A

The rectangular-based shortest path heuristic had the best known solution of 53766.3 seconds with a runtime of only 6.84 seconds. Similar results were obtained when acceleration was accounted for, with a runtime of 5.6967 seconds. The lack of a solar panel had a great influence on the total runtime of this datasets; it took 2194.34 seconds to obtain a solution of 186,190 seconds with the shortest path heuristic; this indicates the BEV needs to drive more than three times slower

Table 10: Results for *smalloo6* with a one-hour solver time-limit.

Solar Panel	Acceleration	Method	Runtime (s)	Peak Memory (KB)	Objective Value(s)	Relative Gap
Yes	No	FC-BLK-FC	>1h	3057636	Time-limit	N/A
		P-BLK-K1	>1h	23138624	320078	*0.379792
		P-BLK-K2	>1h	>32GB	Time-limit	*0.838049
		P-BLK-K3	>1h	6715296	Time-limit	*0.83774
		FC-BLK-FK1	>1h	19263376	Time-limit	N/A
		P-BLK-P	>1h	5745376	Time-limit	N/A
Yes	Yes	FC-BLK-FC	>1h	3043096	Time-limit	N/A
		P-BLK-K1	>1h	23330460	320078	*0.380518
		P-BLK-K2	>1h	>32GB	Time-limit	N/A
		P-BLK-K3	>1h	6729788	Time-limit	N/A
		FC-BLK-FK1	>1h	19413812	Time-limit	N/A
		P-BLK-P	>1h	5776960	Time-limit	N/A
No	Yes	FC-BLK-FC	>1h	19363936	Time-limit	N/A
		P-BLK-K1	>1h	3457864	Time-limit	N/A
		P-BLK-K2	>1h	3569476	Time-limit	N/A
		P-BLK-K3	>1h	3758172	Time-limit	N/A
		P-BLK-P	>1h	5921664	Time-limit	N/A
Yes	No	FC-TRI-FC	>1h	5503144	Time-limit	N/A
		P-TRI-K1	>1h	22838880	Time-limit	N/A
		P-TRI-K2	>1h	4016772	Time-limit	N/A
		P-TRI-K3	>1h	4433680	Time-limit	N/A
		P-TRI-P	>1h	20771148	Time-limit	N/A
Yes	Yes	FC-TRI-FC	>1h	5491508	Time-limit	N/A
		P-TRI-K1	>1h	22833732	Time-limit	N/A
		P-TRI-K2	>1h	4032464	Time-limit	N/A
		P-TRI-K3	>1h	4445120	Time-limit	N/A
		P-TRI-P	>1h	20539232	Time-limit	N/A
No	Yes	FC-TRI-FC	>1h	26982620	Time-limit	N/A
		P-TRI-K1	>1h	8452132	Time-limit	N/A
		P-TRI-K2	>1h	7990440	Time-limit	N/A
		P-TRI-K3	>1h	5367672	Time-limit	N/A
		P-TRI-P	>1h	20189384	Time-limit	N/A

on the shortest path when a solar panel was not attached. The lack of a solar panel attached to the BEV causes the battery State of Charge (SoC) to decrease for any speed thus energy graphs on each segment that can be solved with less effort. When the two shortest paths are considered, a solution was found within 303.09 seconds, with an objective value of 95,519.4 seconds.

Table 11: Results for *smalloo7* with a one-hour solver time-limit.

Solar Panel	Acceleration	Method	Runtime (s)	Peak Memory (KB)	Objective Value(s)	Relative Gap
Yes	No	FC-BLK-FC	9.07	259476	**Unstable**	N/A
		P-BLK-K1	>1h	23138624	380856	*0.066016
		P-BLK-K2	>1h	4188932	Time-limit	N/A
		P-BLK-K3	>1h	910920	Time-limit	N/A
		FC-BLK-FK1	16.16	383604	**Unstable**	N/A
		P-BLK-P	>1h	3956784	Time-limit	N/A
Yes	Yes	FC-BLK-FC	8.90	261364	**Unstable**	N/A
		P-BLK-K1	>1h	3403036	380856	*0.064989
		P-BLK-K2	>1h	4280536	Time-limit	N/A
		P-BLK-K3	>1h	914580	Time-limit	N/A
		FC-BLK-FK1	15.5886	381996	**Unstable**	N/A
		P-BLK-P	>1h	3969828	Time-limit	N/A
No	Yes	FC-BLK-FC	50.51	472236	**Unstable**	N/A
		P-BLK-K1	>1h	3529012	401220	*0.123865
		P-BLK-K2	>1h	3652008	Time-limit	N/A
		P-BLK-K3	>1h	14187760	Time-limit	N/A
		FC-BLK-FK1	11.50	285156	**Unstable**	N/A
		P-BLK-P	>1h	3727012	Time-limit	N/A
Yes	No	FC-TRI-FC	>1h	3981152	Time-limit	N/A
		P-TRI-K1	>1h	4403620	383440	*0.345274
		P-TRI-K2	>1h	11434160	Time-limit	N/A
		P-TRI-K3	>1h	3956532	Time-limit	N/A
		FC-TRI-FK1	>1h	12369712	Time-limit	N/A
		P-TRI-P	>1h	6498428	Time-limit	N/A
Yes	Yes	FC-TRI-FC	>1h	3980748	Time-limit	N/A
		P-TRI-K1	>1h	4542096	383331	*0.304528
		P-TRI-K2	>1h	11397444	Time-limit	N/A
		P-TRI-K3	>1h	15125976	Time-limit	N/A
		P-TRI-P	>1h	6663872	Time-limit	N/A
No	Yes	FC-TRI-FC	>1h	4814852	411214	0.589439
		P-TRI-K1	>1h	1878868	403207	*0.351031
		P-TRI-K2	>1h	7454228	Time-limit	N/A
		P-TRI-K3	>1h	15086716	Time-limit	N/A
		FC-TRI-FK1	>1h	4545808	Time-limit	N/A
		P-TRI-P	>1h	15984116	Time-limit	N/A

Table 12: Results for *mediumoo1* with a one-hour solver time-limit.

Solar Panel	Acceleration	Method	Runtime (s)	Peak Memory (KB)	Objective Value(s)	Relative Gap
Yes	No	FC-BLK-FC	2234.46	6499352	**Unstable**	N/A
		P-BLK-K1	102.608	1006520	No solution	N/A
		P-BLK-K2	3603.17	17394028	106300	*0.901591
		P-BLK-K3	>1h	3294556	Time-limit	N/A
		P-BLK-P	1835.11	>32 GB	Memory-limit	N/A
Yes	Yes	FC-BLK-FC	2242.49	6551740	**Unstable**	N/A
		P-BLK-K1	112.43	1011636	No solution	N/A
		P-BLK-K2	3613.87	17413536	106132	*0.901435
		P-BLK-K3	>1h	3280248	Time-limit	N/A
		FC-BLK-FK2	2357.18	6124694	**Unstable**	N/A
		P-BLK-P	1794.99	>32 GB	Memory-limit	N/A
No	Yes	FC-BLK-FC	2133.98	6623682	**Unstable**	N/A
		P-BLK-K1	105.50	1051544	No solution	N/A
		P-BLK-K2	>1h	6469484	Time-limit	N/A
		P-BLK-K3	>1h	5470016	Time-limit	N/A
		P-BLK-P	1723.45	>32GB	Memory-limit	N/A
Yes	No	FC-TRI-FC	>1h	29358748	Time-limit	N/A
		P-TRI-K1	>1h	5977832	Time-limit	N/A
		P-TRI-K2	>1h	14378712	Time-limit	N/A
		P-TRI-K3	>1h	15529968	Time-limit	N/A
		P-TRI-P	>1h	23330916	Time-limit	N/A
Yes	Yes	FC-TRI-FC	>1h	29565304	Time-limit	N/A
		P-TRI-K1	>1h	5950536	Time-limit	N/A
		P-TRI-K2	>1h	14495352	Time-limit	N/A
		P-TRI-K3	>1h	15559272	Time-limit	N/A
		P-TRI-P	>1h	23306112	Time-limit	N/A
No	Yes	FC-TRI-FC	>1h	27073060	Time-limit	N/A
		P-TRI-K1	>1h	5990056	Time-limit	N/A
		P-TRI-K2	>1h	12857988	Time-limit	N/A
		P-TRI-K3	>1h	16176372	Time-limit	N/A
		P-TRI-P	>1h	24180696	Time-limit	N/A

Table 13: Results for *mediumoo4* with a one-hour solver time-limit.

Solar Panel	Acceleration	Method	Runtime (s)	Peak Memory (KB)	Objective Value(s)	Relative Gap
Yes	No	FC-BLK-FC	>1h	3756516	53937.1	0.735347
		P-BLK-K1	6.84055	133132	53766.3	0
		P-BLK-K2	69.715	683120	53766.3	8.5673e-05
		P-BLK-K3	2928.3	1826240	53766.3	9.84602e-05
		FC-BLK-FK1	>1h	5069148	53937.1	0.900043
		FC-BLK-FK2	>1h	5072292	53937.1	0.900043
		FC-BLK-FK3	>1h	5073824	53937.1	0.900043
		P-BLK-P	727.22	>32 GB	Memory-limit	N/A
Yes	Yes	FC-BLK-FC	>1h	3722552	53937.1	0.735856
		P-BLK-K1	5.69672	133692	53766.3	0
		P-BLK-K2	69.1894	683184	53766.3	8.5673e-05
		P-BLK-K3	2941.6	1823988	53766.3	9.84602e-05
		FC-BLK-FK1	>1h	5064472	53937.1	0.900043
		FC-BLK-FK2	>1h	5072236	53937.1	0.900043
		FC-BLK-FK3	>1h	5064992	53937.1	0.900043
		P-BLK-P	566.43	>32 GB	Memory-limit	N/A
No	Yes	FC-BLK-FC	>1h	14768345	95519.4	0.985464
		P-BLK-K1	2194.34	1224236	186190	0
		P-BLK-K2	303.088	1707808	95519.4	9.99997E-05
		P-BLK-K3	>1h	3314088	95519.4	9.63038E-05
		FC-BLK-FK1	>1h	13562572	95519.4	0.932579
		FC-BLK-FK2	>1h	4879304	95519.4	0.96289
		FC-BLK-FK3	>1h	5326787	95519.4	0.946327
		P-BLK-P	588.46	>32GB	Memory-limit	N/A
Yes	No	FC-TRI-FC	>1h	10410272	Time-limit	N/A
		P-TRI-K1	>1h	3196936	64547.7	0.44275
		P-TRI-K2	>1h	27207720	54352.9	0.516356
		P-TRI-K3	>1h	20923684	Time-limit	N/A
		FC-TRI-FK1	>1h	15783636	Time-limit	N/A
		FC-TRI-FK2	>1h	13243696	Time-limit	N/A
		P-TRI-P	665.82	>32GB	Memory-limit	N/A
Yes	Yes	FC-TRI-FC	>1h	10349320	Time-limit	N/A
		P-TRI-K1	>1h	3200056	64547.7	0.44275
		P-TRI-K2	>1h	27323144	54352.9	0.517501
		P-TRI-K3	>1h	20877380	Time-limit	N/A
		FC-TRI-FK1	>1h	15607096	Time-limit	N/A
		FC-TRI-FK2	>1h	13459352	Time-limit	N/A
		P-TRI-P	501.38	>32GB	Memory-limit	N/A
No	Yes	FC-TRI-FC	>1h	15931244	Time-limit	N/A
		P-TRI-K1	>1h	231620	Time-limit	0.662145
		P-TRI-K2	>1h	117547	Time-limit	0.237979
		P-TRI-K3	>1h	20877380	Time-limit	N/A
		FC-TRI-FK1	>1h	15276896	Time-limit	N/A
		FC-TRI-FK2	>1h	10289156	Time-limit	N/A
		P-TRI-P	863.63	>32GB	Time-limit	N/A

4.3 SUMMARY

Overall, the k-shortest path heuristic with the block method for approximating energy has the potential to be used in real-world scenarios. The flow conservation model is more likely to become numerically unstable compared to the path-based model, especially when the model accounts for energy used when accelerating. When using the triangle method for creating energy graphs, models struggled to find feasible solutions within a reasonable time. For most datasets, a solution was found as the shortest path or the second shortest path. For a route that was only between close cities, or within a city, the datasets solved in under three minutes. The models failed to find feasible solutions when the arc length was substantial (>50 km). One reason is that the total battery usage over that time is more than the battery capacity, since the velocity on the segment needs to be constant. Another problem is that the slope is not captured in detail with an arc length of more than 50 km. The weather may change significantly in 50 km, and overall, this granularity plays a significant role in the solution quality and whether solutions are found. An apparent method to improve the solution quality is to increase the dataset's granularity, but as the number of arcs increases, so does the computation time. When solving for all possible paths, the path-based model does not always result in a feasible solution, and when the average edge degree is large (more than 3), the computation time and memory usage to generate all possible paths are high. Although the flow conservation model has potential, the use of multiple Big M constraints may create numerical instability. By choosing M to be twice the battery capacity, correct results were provided, but there were some cases where numerical instability was an issue. Using the path-based heuristics as a warm-start to the flow conservation model sometimes positively affected the solution quality. The path-based heuristic can also quickly eliminate short infeasible paths, which allows a smaller problem to be solved with the path-based model.

The flow conservation model is a better formulation when considering runtime and peak memory performance since there is no need to define each path explicitly. The path-based model can more easily be extended to include extra features such as acceleration. In the next chapter, the path-based model is adapted to solve the race strategy problem of the Sasol Solar Challenge.

5

CASE STUDY

5.1 OVERVIEW

The Sasol Solar Challenge is a biennial competition where multiple teams from around the world design and build solar-powered vehicles to travel across South Africa over eight days. The routes are usually between Pretoria and Cape Town. The data associated with routes include altitude, speed limit and weather variations. There are multiple loops on each route; for instance, the participants can take each loop more than once. The objective is to travel the greatest possible distance with a solar-powered vehicle.

Most of the focus in the past was on mechanical and electrical design, while participants gave little attention to race strategy. The race strategy is formulated as an optimisation problem by expanding the general models discussed in Chapter 3. The objective is clear; we need to maximise distance travelled, accounting for the regulations of the challenges, speed limits on the roads and weather predictions over eight days.

The problem is similar to the World Solar Challenge but the objective and rules are slightly different. With the World Solar Challenge the participants need to travel a fixed distance. The objective of the World Solar Challenge is to minimise the total race time. Most of the apparent racing strategies apply to both challenges, such as balancing power resources and power consumption.

The granularity and availability of data need to be considered since it plays a significant role in the computation-time and memory-usage of the implemented models. The modelling approach is elaborated on in Section 5.4. Weather data used in this optimisation runs were obtained using `solarpy` for clear sky radiation and `Weatherbit` for weather forecasts.

5.2 RELATED WORK

Merino et al. [76] proposed an energy management system for the World Solar Challenge that consists of three stages. The first stage plans for long-term, i.e. several days, the second stage plans for the current day, and the last stage applies a continuous-time optimal control algorithm to the problem. The authors mention that their proposed algorithms give an advantage over less straightforward methods that keep constant speed along the way.

Betancur et al. [77] uses two heuristics to address the World Solar Challenge. They favour the evolutionary algorithm because the exhaustive search is computationally expensive and the *Big Bang-Big Crunch* [78] converges slower.

Shimizu et al. [79] introduced a cruising strategy support system, which consists of three elements, supervision support, cruising simulation and speed optimising control.

Howlett [80, 81] used dynamic programming (DP) to maximise the expected distance travelled on the remaining days of the World Solar Challenge. A Markov process with continuous state-space was used to model the solar radiation. The model performed reasonably, but they did not take acceleration from one velocity to another into account. The Solar World Challenge held in

Australia has relatively flat gradients and is predominantly sunny; this may indicate that the race strategy is less important than the vehicle's mechanical and electric soundness. The approach of Howlett closely relates to the case study in this chapter, but with the exception being that there are no optional loops in the World Solar Challenge.

The masters thesis of Scheidegger [82] also approaches the Solar World Challenge as a DP problem and accounts for the most significant vehicle and environment factors. The author implements the energy management optimisation for a solar-powered vehicle as a deterministic and stochastic problem and defines precise mathematical formulations for the problem. The drawback to the models implemented by Scheidegger is the lack of accounting energy when accelerating.

Guerrero and Daurte-Mermoud [76] provided a detailed mathematical description of the vehicle and a descriptive mathematical model for weekly-, daily- and continuous planning. The authors implemented the models in MATLAB® using the free Gauss Pseudospectral Optimisation Software (GPOPS) 4.1. The models do not account for acceleration and use discrete time intervals. The authors provide simulation results that show the reliability of their models.

The most significant work related to this case study is Oosthuizen et al. [72]. The authors implement two algorithms, one for single day optimisation using high-resolution data and the second for multiple-day optimisation but with lower resolution data. The single-day algorithm uses a sequential quadratic programming solver. It optimises the energy stored by controlling the vehicle speed where the multiday algorithm optimises for total distance travelled. They approached the multiday method as a dynamic program that estimates the number of extra loops the solar-powered vehicle needs to take each day. The algorithms were tested during the 2018 Sasol Solar Challenge, with promising results.

5.3 RESEARCH PROBLEM

There are currently various race strategies documented for the World Solar Challenge, but limited race strategy algorithms and models exist in the literature for the Sasol Solar Challenge. This thesis's approach is slightly different than [72], since the method choice is a Mixed Integer Linear Programming (MILP) model, which assumes determinism. The approach allows the model implementation to heavily depend on the weather predictions and vehicle simulations discussed in section 3.3.1 (more accurate predictions produces more accurate optimisation results with the same model). This model's philosophy is, given the most accurate available data, what decision should be made?

- Given the available weather data for the remaining race days, vehicle and road parameters, what is the best possible speed to drive the vehicle?
- When there are loops available on the route, how many loops should the vehicle drive?
- When is it advantageous to stay in an area to charge the battery?

5.4 RESEARCH METHODOLOGY AND CONTRIBUTIONS

The model described in this chapter extends on the path-based model described in Chapter 3, since the main advantage of the path-based model is the ability to limit the number of paths. The

Sasol Solar Challenge has a single predefined path with the ability to drive additional loops. These loops can easily be incorporated as different paths in the path-based model. The Battery Electric Vehicle (BEV) simulation, weather simulation and data gathering methods described in Chapter 3 apply to this chapter. The Sasol Solar Challenge demands an efficient solar-powered vehicle with a thought-through racing strategy that takes road profiles, predicted weather conditions and the battery state into account.

The objective of this case study is describe usable mathematical models, simulations, and algorithms that result in a practical race strategy for the Sasol Solar Challenge. The main objective of this chapter is to develop an optimisation model (MILP) that addresses the race strategy of the Sasol Solar Challenge and implement a simulation for the vehicle on the road that can be used in the optimisation model. Additionally, gather or simulate weather data for race conditions.

5.5 MODEL

The objective of the Sasol Solar Challenge is to maximise the total distance travelled of the vehicle over the span on multiple days. The complexity of the problem arises with the decision of how many loops the vehicle should drive on each day, and at what speed; these decisions affect future decisions. The path-based model described in Chapter 3 allows multiple routes to be defined, paths are the default route with enumerated loops, e.g. the first path is the route without a loop, the second path is the route with one loop, the third path is a route with two loops, etc. The path-based model can easily be adjusted to accommodate problems similar to the Sasol Solar Challenge. The objective is to maximise the total distance travelled, that is

$$\sum_{d\in\mathcal{D}}\sum_{\rho\in\mathcal{P}(d)} z_\rho\zeta_\rho, \tag{5.1}$$

subject to

$$\sum_{\rho\in\mathcal{P}(d)}\zeta_\rho = 1, \qquad \forall d\in\mathcal{D}, \tag{5.2}$$

$$\tau_{(|\mathcal{S}(\rho)|-1)} + t_{(|\mathcal{S}(\rho)|-1)} + M(\zeta_\rho - 1) \leq T_d\zeta_\rho, \qquad \forall\rho\in\mathcal{P}(d), d\in\mathcal{D}, \tag{5.3}$$

$$\tau_{(s-1)} + t_{(s-1)} = t_s, \qquad \forall s\in\mathcal{S}(\rho)\backslash\{0\}, \rho\in\mathcal{P}(d), d\in\mathcal{D}, \tag{5.4}$$

$$v_s = \sum_{x\in\mathcal{X}}\alpha_{xs}v_{xs}, \qquad \forall s\in\mathcal{S}(\rho), \rho\in\mathcal{P}(d), d\in\mathcal{D}, \tag{5.5}$$

$$\tau_s = \sum_{x\in\mathcal{X}}\alpha_{xs}\tau_{xs}, \qquad \forall s\in\mathcal{S}(\rho), \rho\in\mathcal{P}(d), d\in\mathcal{D}, \tag{5.6}$$

$$\alpha_{xs} \leq h_{(x-1)s} + h_{xs}, \qquad \forall x\in\mathcal{X}, s\in\mathcal{S}(\rho), \rho\in\mathcal{P}(d), d\in\mathcal{D}, \tag{5.7}$$

$$\sum_{x\in\mathcal{X}\backslash\{X\}} h_{xs} = \zeta_\rho, \qquad \forall s\in\mathcal{S}(\rho), \rho\in\mathcal{P}(d), d\in\mathcal{D}, \tag{5.8}$$

$$\sum_{x\in\mathcal{X}}\alpha_{xs} = \zeta_\rho, \qquad \forall s\in\mathcal{S}(\rho), \rho\in\mathcal{P}(d), d\in\mathcal{D}, \tag{5.9}$$

$$t_s \leq \sum_{y \in \mathcal{Y} \setminus \{Y\}} \kappa_{ys} t_{(y+1)s}, \qquad \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}(d), d \in \mathcal{D}, \tag{5.10}$$

$$t_s \geq \sum_{y \in \mathcal{Y} \setminus \{Y\}} \kappa_{ys} t_{ys}, \qquad \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}(d), d \in \mathcal{D}, \tag{5.11}$$

$$\sum_{y \in \mathcal{Y} \setminus \{Y\}} \kappa_{ys} = \zeta_\rho, \qquad \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}(d), d \in \mathcal{D}, \tag{5.12}$$

$$e_{s\rho d} \leq \sum_{x \in \mathcal{X}} \alpha_{xs} e_s(x, y) + M(1 - \kappa_{ys}), \qquad \forall y \in \mathcal{Y}, \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}(d), d \in \mathcal{D}, \tag{5.13}$$

$$e_{s\rho d} \geq \sum_{x \in \mathcal{X}} \alpha_{xs} e_s(x, y) - M(1 - \kappa_{ys}), \qquad \forall y \in \mathcal{Y}, \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}(d), d \in \mathcal{D}, \tag{5.14}$$

$$E_{s\rho d} = e_{s\rho d} + E_{(s-1)\rho d}, \qquad \forall s \in \mathcal{S}(\rho) \setminus \{0\}, \rho \in \mathcal{P}(d), d \in \mathcal{D}, \tag{5.15}$$

$$E_{s\rho d} = e_{s\rho d} + \epsilon_H, \qquad \forall s \in \mathcal{S}(\rho) \bigcap \{0\}, \rho \in \mathcal{P}(0), \tag{5.16}$$

$$E_{s\rho d} = e_{s\rho d} + E_{(|\mathcal{S}(r)|-1)(r)(d-1)}, \qquad \forall s \in \mathcal{S}(\rho) \bigcap \{0\}, r \in \mathcal{P}(d-1), \rho \in \mathcal{P}(d), d \in \mathcal{D} \setminus \{0\}, \tag{5.17}$$

$$-(1 - \zeta_\rho)M + \epsilon_L \leq E_{s\rho d} \leq \epsilon_H + (1 - \zeta_\rho)M, \qquad \forall s \in \mathcal{S}(\rho), \rho \in \mathcal{P}(d), d \in \mathcal{D}. \tag{5.18}$$

$\mathcal{D}$ is a set that contains the remaining race days. $\mathcal{P}(d)$ is a set that contains the different routes of day d. ζ_ρ is a binary decision variable used to select the route of the day. z_ρ is the total distance of route ρ.

$\mathcal{S}(\rho)$ is a set containing the segments on route ρ. t_s is the start time on segment s. τ_s is the duration it takes the solar car travelling velocity v_s on segment s. τ_s, t_s and v_s are decision variables.

The sets $\mathcal{X}$ and $\mathcal{Y}$ contains indices for the sample vertices used for linear piecewise approximation of the non linear functions, such as velocity, duration and energy consumption. α_{xs}, κ_{ys} and h_{xs} are Special Ordered Set (SOS) variables. α_{xs} and κ_{ys} are continuous variables and h_{xs} is a binary decision variable. M is a large number used for BigM constraints.

Continuous decision variable $e_{s\rho d}$ is the energy consumed on segment s and $E_{s\rho d}$ is a continuous decision variable that represents the total energy at the beginning of segment s. Constant value ϵ_H is the battery capacity and ϵ_L is the predefined minimum energy the battery needs to prevent damage.

Constraint set (5.2) ensures that one route is selected per day. Constraint set (5.3) only selects a route that can be completed within the fixed time-limit of the day T_d. M allows the constraints to be satisfied when a route is not selected. Constraint set (5.4) forces the starting time to be the sum of the previous segment start time and duration.

The duration τ_s is a function of the velocity and the segment distance, linear piecewise approximation of the function,

$$\tau_s(v_s) = \frac{d_s}{v_s} \qquad \forall s \in S \tag{5.19}$$

is used. Constraint set (5.5), (5.6), (5.7), (5.8), (5.9) allow linear piecewise approximation of function (5.19).

An alternative method to implement the multi-dimensional linear piecewise approximation is, to replace (5.10) - (5.14) with (5.20) - (5.25).

$$\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}}\beta_{xys}=1, \qquad \forall s\in\mathcal{S}(\rho),\rho\in\mathcal{P}(d),d\in\mathcal{D} \tag{5.20}$$

$$v_s=\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}}\beta_{xys}v_{xs}, \qquad \forall s\in\mathcal{S}(\rho),\rho\in\mathcal{P}(d),d\in\mathcal{D} \tag{5.21}$$

$$t_s=\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}}\beta_{xys}t_{ys}, \qquad \forall s\in\mathcal{S}(\rho),\rho\in\mathcal{P}(d),d\in\mathcal{D} \tag{5.22}$$

$$e_{s\rho d}=\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}}\beta_{xys}e_{xys}, \qquad \forall s\in\mathcal{S}(\rho),\rho\in\mathcal{P}(d),d\in\mathcal{D} \tag{5.23}$$

$$\sum_{x\in\mathcal{X}\setminus\{X\}}\sum_{y\in\mathcal{Y}\setminus\{Y\}}\kappa^{(u)}_{xys}+\kappa^{(l)}_{xys}=1, \qquad \forall s\in\mathcal{S}(\rho),\rho\in\mathcal{P}(d),d\in\mathcal{D} \tag{5.24}$$

$$\begin{aligned}\beta_{xys}\le\kappa^{(u)}_{xys}+\kappa^{(l)}_{xys}+\kappa^{(u)}_{x(y-1)s}+\kappa^{(l)}_{(x-1)ys}+\kappa^{(u)}_{(x-1)(y-1)s}+\kappa^{(l)}_{(x-1)(y-1)s}\\ \forall x\in\mathcal{X}\setminus\{0\},y\in\mathcal{Y}\setminus\{0\},s\in\mathcal{S}(\rho),\rho\in\mathcal{P}(d),d\in\mathcal{D}\end{aligned} \tag{5.25}$$

Constraint sets (5.20) - (5.25) is described as the *Triangle method* in [51] where (v_{xs},t_{ys}) are sampling coordinates for $x\in\mathcal{X},y\in\mathcal{Y}$ on segment s and e_{xyd} is the function evaluated at each breakpoint (v_{xs},t_{ys}). $\beta_{xys}\in[0,1]$ is a continuous variable (one per breakpoint), used for computing the convex combinations for the three-dimensional space. β_{xys} should be defined as a Special Ordered Set of Type 3 (SOS3) but current MILP solvers do not have such functionality, thus we associate $\kappa^{(u)}_{xys},\kappa^{(l)}_{xys}$ with the upper and lower triangle in the rectangle, thus requiring constraint sets (5.24) and (5.25). Other formulations of the *Triangle method* were also introduced by [83–86].

5.5.1 *Panel tilt optimisation*

For the 2018 Sasol Solar Challenge, the North-West University solar car's design decision was to allow the attached solar panel to tilt around an axis parallel to the vehicle's direction. Figure 23 shows a mechanical drawing of a solar car with a tilt-able panel. On the left of Figure 23, the top view of a solar car is shown as a supplement to the front view on the right. The front view on the right shows the maximum allowed solar panel tilt θ_{tilt}, where the tilt can be between -30° and 30°. The solar panel on the vehicle can only rotate around the vector in which the vehicle is moving. With the rotation axis limitation, the solar panel does not always need to be tilted. The energy for rotating the solar panel from -30° and 30° is assumed to be 90 J. The optimisation model does not have a memory of the previous segment's panel position; this may not reflect the exact energy usage of the panel tilt, but 90 J is considered neglectable, especially for large segments and a battery capacity of 18 MJ. At the start of each segment, the panel is assumed to be at 0°, but in reality, this depends on the previous segment. The panel tilt is not a variable in the optimisation model, but there is a local optimisation for each segment to determine the optimal consumed power given a certain speed, and starting 0°.

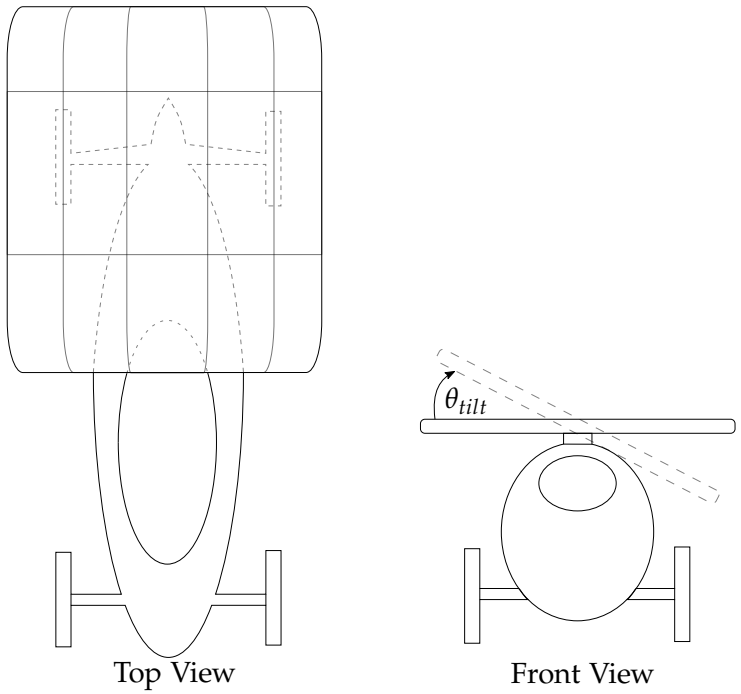


Figure 23: Simplified mechanical drawing of a BEV with a tilt-able solar panel.

5.5.2 *Drive-train efficiency*

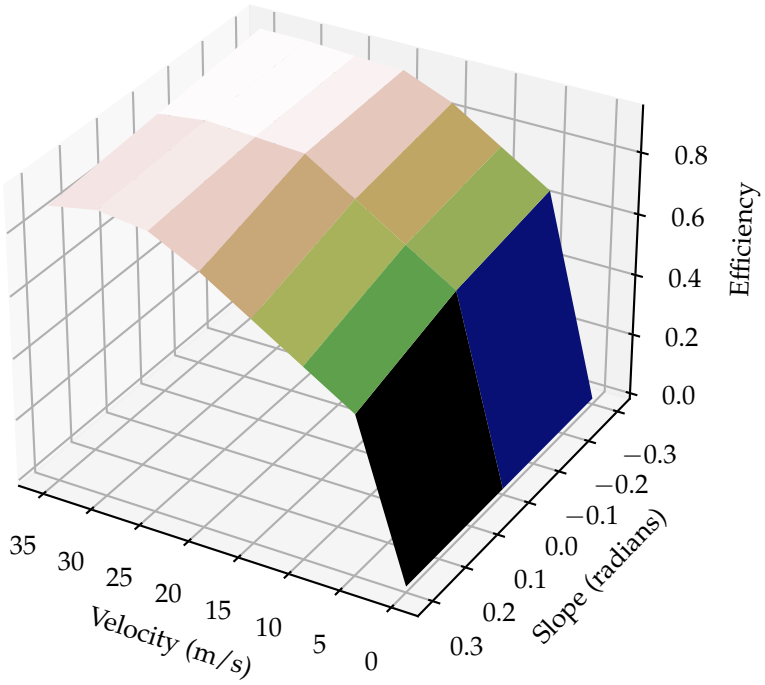


Figure 24: Drive-train efficiency given a slope and velocity.

The drive-train efficiency varies depending on the load the motor is currently under as-well as the current speed. In most of the literature studies, the vehicle simulation assumes a fixed drive-train efficiency. Figure 24 shows the drive-train efficiency graph, given a velocity and slope on a segment. The slope is varied between -0.3 radians and 0.3 radians, where -0.3 is a downhill slope and 0.3 is an uphill slope. Road slope are unlikely to be larger than 0.3 radians (around 17.2°). Overall the drive-train efficiency declines as the vehicle speed declines. As the slope increases, the load on the motor is more and the efficiency also declines. The drive-train efficiency values can usually be estimated with the help of the motor data-sheet and load measurements on different slopes accounting for vehicle mass.

5.6 RESULTS

To demonstrate the generality of the model, the short track at Red Star Raceway (26° 4′ 30.4716″ S, 28° 45′ 18.6444″ E) shown in Figure 25 is used for a single day optimisation. Starting and ending on vertex 2186 with one lap is approximately 1.5 km. The short-term optimisation uses the graph in Figure 25 and includes local tilt optimisations. For the single-day and multi-day optimisation, segments are aggregated. Multiple laps around the track are described as a single segment; for the single day optimisation, nine laps are aggregated to a single segment.

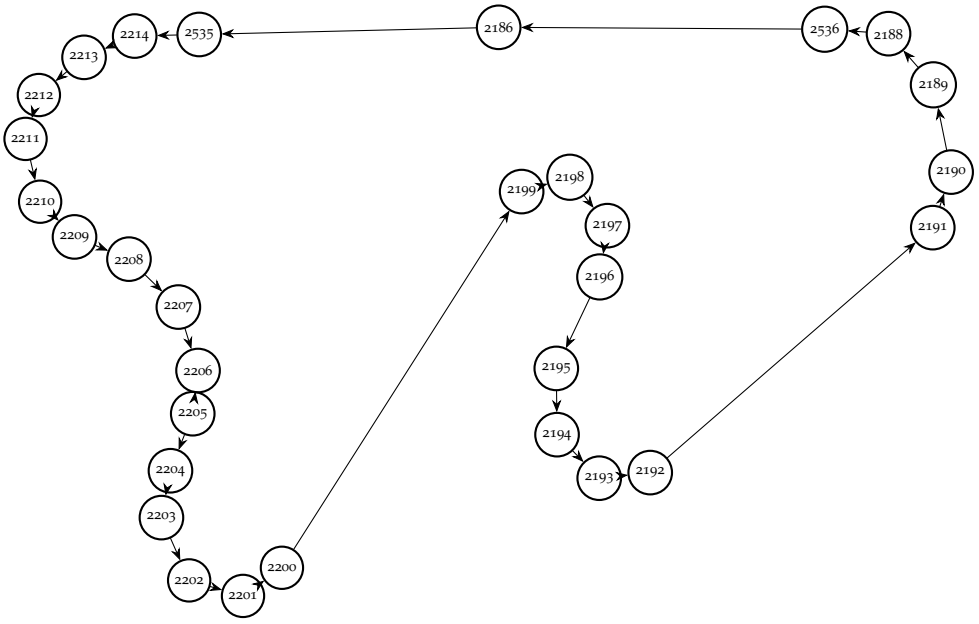


Figure 25: Short track input graph.

5.6.1 *Single day optimisation for the Red Star Raceway*

For the simulation aspect of the case study, a clear sky is assumed. The clear sky data retrieved was in three-minute intervals, where the values between the intervals were linearly interpolated. The race's start time in the case study is 8:00 AM (SAT) and ends on 5:00 PM (SAT). Figure 53, 54, 55 and 56 in Appendix D show the resulting energy graphs for the optimisation run. The green dot on the graphs represents the energy usage calculated by the optimisation model, and the red dot shows the energy usage calculated by the simulation. If only one dot is shown, the dots are on the same location. Each graph shows the associated segment's solution;

this includes the velocity, start time and energy consumed on the segment. Each segment cannot start before the previous segment. A simple visual inspection can verify this. The start time of each segment increases as the vehicle moves to the next segment. The first and last segments are short, less than 20 m, and their energy graphs show that the energy consumed is much lower than the long segment counterparts. Segment 46 to 67 are long segments that describe the aggregated nine laps around the track, each segment with a distance of 13.4 km. The start time on the energy graph is displayed in seconds, where 0 is 8:00 AM (SAT). As the time-of-day progress, the effect of solar irradiation is prominent for low velocities (under $10\,\mathrm{m\,s^{-1}}$). As the velocities increase, the vehicle's energy consumed is more significant than the energy gained from solar panels.

The maximum number of loops for the dataset can be adjusted in the preprocessing steps. When the search space is narrowed, e.g. only consider 190 to 300 laps rather than 1 to 500 laps, the optimisation run-time can be reduced significantly. Insights into the problem, such as the maximum number of laps achievable when considering vehicle limits, can reduce the search space.

s	d_s(m)	η_M	$\mathcal{E}_{scl}$%	$\mathcal{E}_{act}$%	e_{Model}(J)	e_{Sim} (J)	v_s(m s^{-1})	t_s(s)	τ_s(s)
45	16.09	0.89	-0.00	0.00	-3007.12	-3007.12	19.44	0.00	0.83
46	13388.52	0.72	0.00	0.01	-1377521.22	-1377443.22	11.11	0.83	1204.97
47	13388.52	0.64	1.20	22.74	-1166936.97	-950741.68	7.05	1205.79	1977.65
48	13388.52	0.67	3.27	86.91	-1264337.59	-676441.61	8.33	3183.44	1606.62
49	13388.52	0.61	0.95	168.08	-272812.08	-101766.38	5.56	4790.07	2409.93
50	13388.52	0.77	-0.01	0.17	-932338.22	-933943.80	13.66	7200.00	983.50
51	13388.52	0.78	0.20	3.95	-950763.85	-914650.10	13.89	8183.50	963.97
52	13388.52	0.89	0.28	3.92	-1322407.66	-1272471.68	19.44	9147.47	688.55
53	13388.52	0.78	0.51	10.66	-950763.85	-859149.57	13.89	9836.03	963.97
54	13388.52	0.78	-0.00	0.00	-836740.14	-836740.14	13.89	10800.00	963.97
55	13388.52	0.89	0.11	1.60	-1225482.40	-1206210.72	19.21	11763.97	698.26
56	13388.52	0.78	0.16	3.67	-836740.14	-807109.80	13.89	12462.23	963.97
57	13388.52	0.78	0.24	5.33	-836740.14	-794410.16	13.89	13426.20	963.97
58	13388.52	0.72	0.32	11.05	-583198.88	-525162.58	11.11	14390.18	1204.97
59	13388.52	0.61	-0.09	3.36	485959.49	470169.55	5.68	15595.14	2375.06
60	13388.52	0.61	-0.33	12.89	515082.29	456267.32	5.56	17970.20	2409.93
61	13388.52	0.61	-0.64	33.56	455153.46	340780.39	5.56	20380.13	2409.93
62	13388.52	0.61	-0.51	56.54	256397.64	163788.21	5.56	22790.07	2409.93
63	13388.52	0.67	-0.02	0.80	-540319.24	-544685.50	8.56	25200.00	1573.30
64	13388.52	0.67	-0.62	17.66	-517986.38	-629053.96	8.33	26773.30	1606.62
65	13388.52	0.67	-1.30	31.04	-517986.38	-751134.99	8.33	28379.92	1606.62
66	13388.52	0.72	-0.41	6.93	-987282.05	-1060829.34	11.11	29986.54	1204.97
67	13388.52	0.72	-0.83	13.18	-987282.05	-1137211.51	11.11	31191.51	1204.97
68	19.58	0.61	-0.00	25.38	-1946.51	-2608.61	5.56	32396.48	3.52

Table 14: Case study results for the single day optimisation run. Segments labelled 46 - 67 are nine laps of the Red Star Raceway, with d_s the distance travelled on each segment in meter. The drive-train efficiency of the vehicle associated with the segment and speed v_s is given by η_M. The energy consumed, calculated by the simulation is e_{Sim} where as the energy consumed according to the optimisation model is e_{Model}.

Table 14 shows more details for the single day optimisation, this includes, the durations, start times energy used and energy errors. Segment 45 and 68 are dummy segments for the start and end, where segment 46 to 67 are nine laps of the Red Star Raceway. There are two different energy

errors displayed in Table 14, one is scaled to the battery and the other is relative to the simulated values. The scaled energy error $\mathcal{E}_{scl}$ is given by,

$$\mathcal{E}_{scl} = \frac{e_{Sim} - e_{Model}}{\epsilon_H} \times 100\%, \tag{5.26}$$

this approximates the overall influence on the State of Charge (SoC). The actual energy error $\mathcal{E}_{act}$ % is given by,

$$\mathcal{E}_{act} = \frac{|e_{Sim} - e_{Model}|}{|e_{Sim}|} \times 100\%, \tag{5.27}$$

and is the error between the simulated and optimised values. The actual energy error was as high as 168.08%, but it will not have such a large effect on the SoC because the consumed energy is relatively low. Segment 48 had an scaled energy error of 3.27%, which is large relative to the other segments and can be seen in Figure 53d where the simulated energy is much higher than the optimisation energy. The scaled energy needs to be used in conjunction with the actual energy error for race strategy decision making.

Table 14 can be used to verify that the start times and duration adhere to the model constraint,

$$\tau_{(s-1)} + t_{(s-1)} = t_s, \quad \forall s \in \mathcal{S}(\rho), \quad \forall \rho \in \mathcal{P}(d), \quad \forall d \in \mathcal{D}. \tag{5.28}$$

The drive-train efficiency η_M used in the simulation is also displayed in Figure 24. The drive-train efficiency was above 0.6 because the simulated vehicle drove at speeds above $20\,\mathrm{km\,h^{-1}}$ on all segments. The velocity of the vehicle varied between $20\,\mathrm{km\,h^{-1}}$ and $70\,\mathrm{km\,h^{-1}}$. In most cases where the vehicle speed is $20\,\mathrm{km\,h^{-1}}$, energy was gained from the sun. The accumulated scaled error is 2.48%, the simulated SoC will be 2.48% more than in the optimisation run, this can vary and caution should be taken when the accumulated scaled energy is negative.

5.6.2 *Tilt optimisation per segment*

Five laps of the Red Star Raceway were used to generate the results for the tilt optimisation; the main reason for this is the vehicle's start time increases as more energy is available due to the solar panel tilt. The distance the vehicle travels is the same with and without the solar panel tilt accounted for because of the small distances. This dataset is only used to demonstrate the effect a solar panel with the ability to tilt has. Figure 26 shows the sum of the energy consumed and gained by the vehicle on each segment (since both optimisation runs result in the same segments in this case). As time goes on, each segment's start time with solar panel tilt accounted for diverges from the segments where tilt was zero. Figure 26 shows there is an advantage to tilt the solar panel. The solar panel tilt flattens the maximum amplitude of the total energy gained and consumed. As time passes and the earth rotates, the solar panel tilt's effect is less as the start times near midday; this is clear from the increase in energy gained from the solar panel.

Figure 27 shows how the sun angle on the panel changes when the solar panel is rotated around the axis of the segment. Figure 27 shows that the start time on each segment of the vehicle is before the same segment when there is no tilt allowed as a result when the tilt is zero for both optimisation runs on the same segment, the sun angle is lower when tilt is allowed. The start time on the first segment is early in the morning, and the clear sky irradiation usually peaks midday. As time passes, the sun angle on the panel will increase on a clear sky until midday. The variations on the sun angle on the panel are due to the nature of the dataset; it is around a track, so the vehicle makes a full rotation around the normal vector of a zero tilted solar panel.

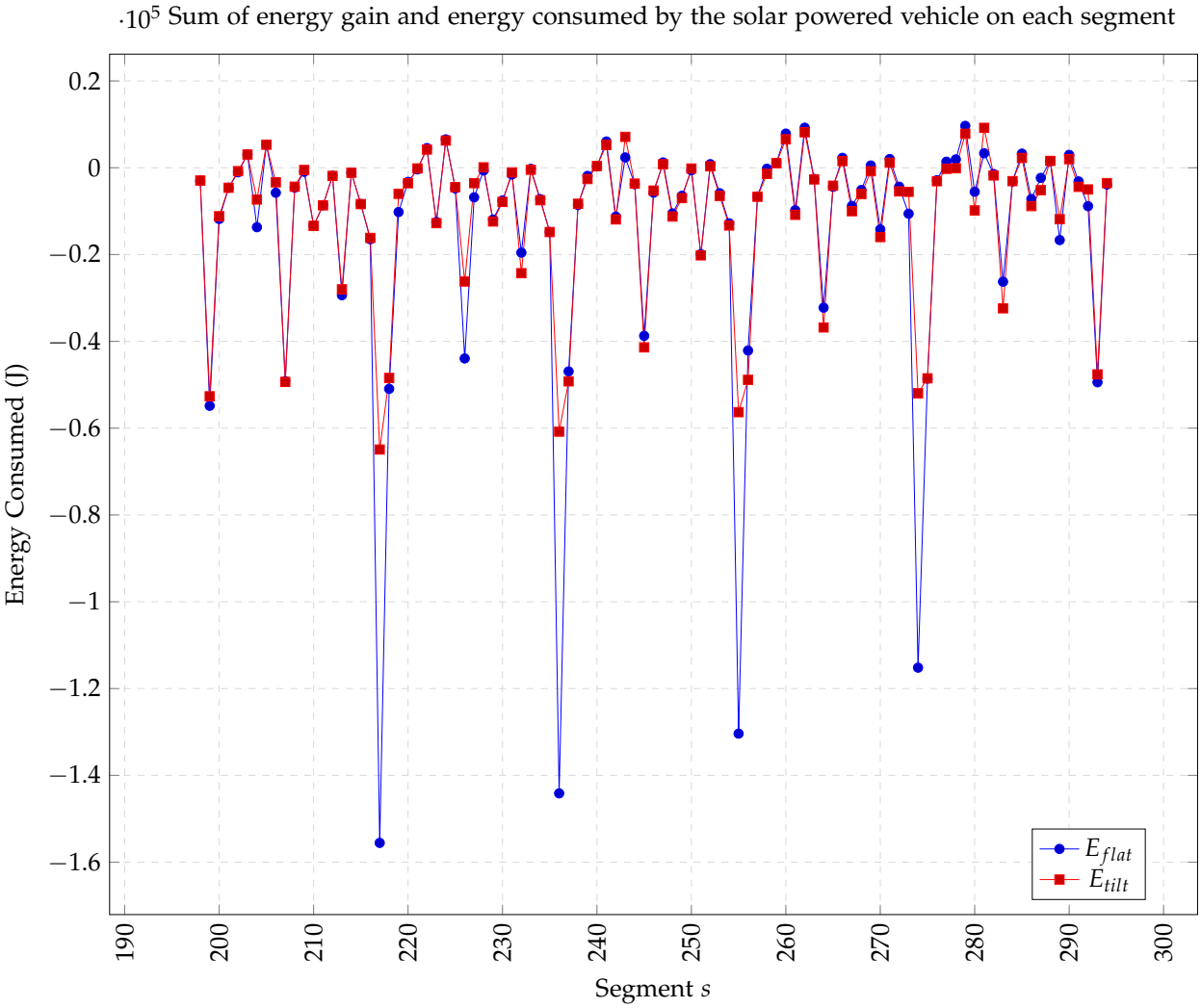


Figure 26: Energy consumed on segment s, with and without solar panel tilt.

5.6.3 *Multi-day optimisation across South Africa*

The dataset starts in Pretoria on the first day and ends in Cape Town on day nine. The dataset is fictitious but very similar to the route of the 2018 Sasol Solar Challenge. The segment lengths of the dataset over multiple days are much longer than for the single day optimisation. At the start of each day, the race drive begins at 8:00 AM (SAT), and the vehicle must be at the starting location of the next day on or before 5:00 PM (SAT). With the current model, the solar-powered vehicle must drive without the intervention of loading it on a trailer. Throughout the simulation, it is assumed that the vehicle does not break down at any time. On the first day, the vehicle starts in Pretoria and drives to Kroonstad, and it drives to a different location each day until the vehicle reaches Cape Town. Figure 28 shows a crude representation of the Sasol Solar Challenge route in 2018. The input graph to each day is provided in Appendix C, and arcs going from and to a vertex that does not follow an obvious path indicates where the vehicle may drive multiple times, i.e. a loop. The weather data used for the simulations begin at 8:00 AM (SAT), 4 February 2021, and ends on 5:00 PM (SAT), 13 February 2021.

Table 15 shows the results for the multi-day optimisation run, with each row corresponding to a road segment. The solver time-limit was set to one hour. Multiple solutions were found within

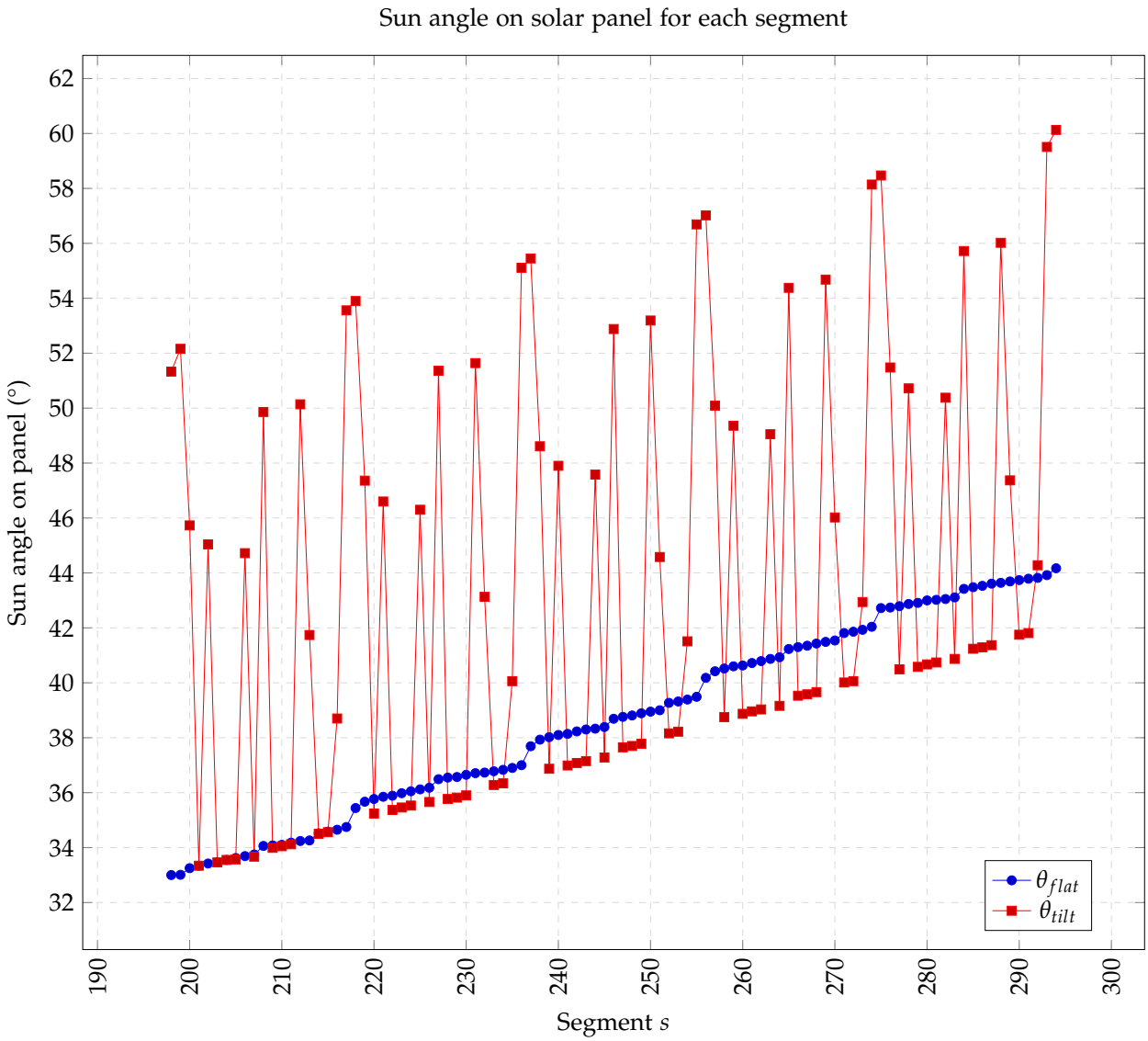


Figure 27: Sun angle on the vehicle for segment s, with and without solar panel tilt.

one hour but still had a sizeable relative gap of 108.4%, from which the best objective value among solutions found was 1851.4 km. Most of the scaled energy error (as defined in Equation 5.6.1) has an absolute energy error lower than 5% except for where velocities were under $5\,\mathrm{m\,s^{-1}}$. The actual energy error shows there is a significant error for velocities under $10\,\mathrm{m\,s^{-1}}$; this error is caused by the form of the energy graphs shown in Appendix E, as the velocity increase, the surface flattens. The start time t_s is the seconds passed since the start of the race day, from 8:00 AM (SAT). The end time needs to be equal to the start time plus the duration; this verifies that the time constraints are correct. The drive-train efficiency is as low as 0.3 when the vehicle is driving slowly. The energy consumed after each segment is given by e_{model}, and always needs to be between 20% and 100% of the battery energy capacity (18 MJ). On day one of the race, the vehicle should drive one loop. On day two to five, the vehicle should only drive the route and skip the loops. On day six and eight, the vehicle should drive one loop, and on day seven and nine, the vehicle should only drive the route.

Figure 28: Sasol Solar Challenge 2018 route [87].

Figure 29 shows the results for the multi-day dataset[1] but for different times of the year. The solver time-limit was set to one-hour for all the datasets in table 15, and was solved with IBM ILOG CPLEX 12.8, with 64 GB memory and AMD RYZEN™ 3900XT@3.8GHz processor. The start time for each dataset is on the second day of each month in 2021. The race starts at 8:00 AM (SAT) and ends at 5:00 PM (SAT) on the same day. The time of year, the solar-powered vehicle is driving through South Africa affects the total distance the vehicle can drive. During the summer[2] the solver found a solution much easier and with a travelling distance of more than 1800 km within the one hour time-limit. There is an apparent increase in travel distance from the start of spring[3] up until the end of spring. There is a declination in the total distance the vehicle can drive in autumn[4]. During winter[5] the vehicle drives the shortest distance, only 857 km. The vehicle's driving range correlates with the irradiation cycle during a year, e.g. lower irradiation during winter and higher irradiation during summer. Since there was a one-hour time-limit, some of the

1 The dataset starts in Pretoria on the first day and ends in Cape Town on day nine, it is the dataset used in Table 15.
2 Summer is between 1 December to 28/29 February.
3 Spring is between 1 September and 30 November.
4 Autumn is between 1 March and 31 May.
5 Winter is between 1 June and 31 August

Table 15: Results for multi-day optimisation across South Africa.

d	s	t_s (s)	τ_s (s)	**End** (s)	v_s(m s^{-1})	$\mathcal{E}_{scl}$ %	$\mathcal{E}_{act}$ %	η_M	E_{Model}(J)
1	0	0.00	4679.77	4679.77	11.11	0.00	-0.00	0.72	12264156.93
1	1	4679.77	6556.74	11236.52	11.11	2.46	-16.31	0.72	9104967.60
1	2	11236.52	571.74	11808.26	27.78	0.03	-0.21	0.92	6481514.38
1	3	14400.00	2891.84	17291.84	5.56	0.00	0.00	0.61	7304077.76
1	4	17291.84	10219.41	27511.25	5.56	-1.35	-11.63	0.61	9641246.63
1	5	27511.25	4888.75	32400.00	10.44	-0.70	4.91	0.71	7220742.45
2	104	0.00	14912.31	14912.31	3.98	-31.39	-116.30	0.47	17728537.32
2	105	14912.31	5876.04	20788.35	8.33	-0.06	1.83	0.67	17147088.31
2	106	21600.00	6509.42	28109.42	9.53	-1.64	14.84	0.69	15454180.45
2	107	28109.42	4290.58	32400.00	8.33	-3.05	32.83	0.67	14331112.16
3	158	0.00	10800.00	10800.00	6.74	-4.96	367.12	0.63	14980627.88
3	159	10800.00	6948.80	17748.80	5.56	1.13	17.41	0.61	15943045.27
3	160	18000.00	3600.00	21600.00	7.92	-0.66	17.63	0.66	15385003.93
3	161	21600.00	5609.17	27209.17	5.56	-1.17	-11.30	0.61	17465159.90
4	217	0.00	4341.65	4341.65	9.61	-0.28	1.31	0.69	13695178.79
4	218	4341.65	5154.47	9496.12	11.11	1.09	-5.39	0.72	9844326.64
4	219	9496.12	12293.11	21789.23	2.78	2.08	5.79	0.31	15939854.46
4	220	21789.23	7010.77	28800.00	5.56	-0.16	1.75	0.61	14310648.80
4	221	28800.00	3600.00	32400.00	15.71	0.04	-0.51	0.81	13014385.48
5	222	0.00	3391.15	3391.15	22.22	0.00	-0.00	0.90	3673899.76
5	223	3391.15	17749.75	21140.90	3.20	-46.17	-148.69	0.37	17574453.57
5	224	21140.90	7659.10	28800.00	8.33	-1.94	-453.13	0.67	18000000.00
5	225	28800.00	3600.00	32400.00	6.66	-0.32	8.29	0.63	17357198.72
6	230	0.00	9316.27	9316.27	5.19	-5.65	80.28	0.60	17107490.77
6	231	9316.27	2391.32	11707.59	13.89	1.01	-7.60	0.78	14523784.10
6	232	11707.59	1840.52	13548.11	11.11	0.10	-15.99	0.72	14395251.81
6	233	13548.11	1189.61	14737.71	16.67	0.21	-1.89	0.82	12369927.63
6	234	14737.71	3262.29	18000.00	5.56	0.02	0.61	0.61	12934403.22
6	235	18000.00	7736.23	25736.23	11.11	0.00	-0.00	0.72	8970617.29
6	236	25736.23	6663.77	32400.00	11.98	-4.35	13.13	0.74	3793685.00
7	345	0.00	17083.68	17083.68	2.42	-37.98	-147.30	0.31	15271745.75
7	346	17083.68	3303.22	20386.90	11.11	0.04	-0.68	0.72	14077139.58
7	347	21600.00	4963.82	26563.82	13.89	0.00	-0.00	0.78	9218258.46
7	348	26563.82	2979.75	29543.57	12.14	-2.42	15.24	0.74	6798444.12
7	349	29543.57	2856.43	32400.00	13.89	-0.54	2.93	0.78	3600000.00
8	350	0.00	17500.97	17500.97	2.27	-41.04	-172.34	0.29	15273635.39
8	351	18000.00	7797.42	25797.42	5.56	-0.00	-0.00	0.61	16984973.82
8	352	28800.00	1721.21	30521.21	25.00	0.00	-0.00	0.91	10580440.47
8	353	30521.21	1878.79	32400.00	24.83	-0.79	1.98	0.91	3600000.00
9	392	0.00	18000.00	18000.00	1.77	-56.49	-249.32	0.23	17847573.11
9	393	18000.00	13241.54	31241.54	2.69	2.14	107.42	0.30	17821036.11
9	394	31241.54	756.34	31997.88	30.56	-0.45	2.17	0.92	14130603.24
9	395	31997.88	402.12	32400.00	33.33	-0.21	1.10	0.90	10824302.19

datasets did not solve optimally and causes some variation in the total distance the vehicle can drive, but the overall trend is clear.

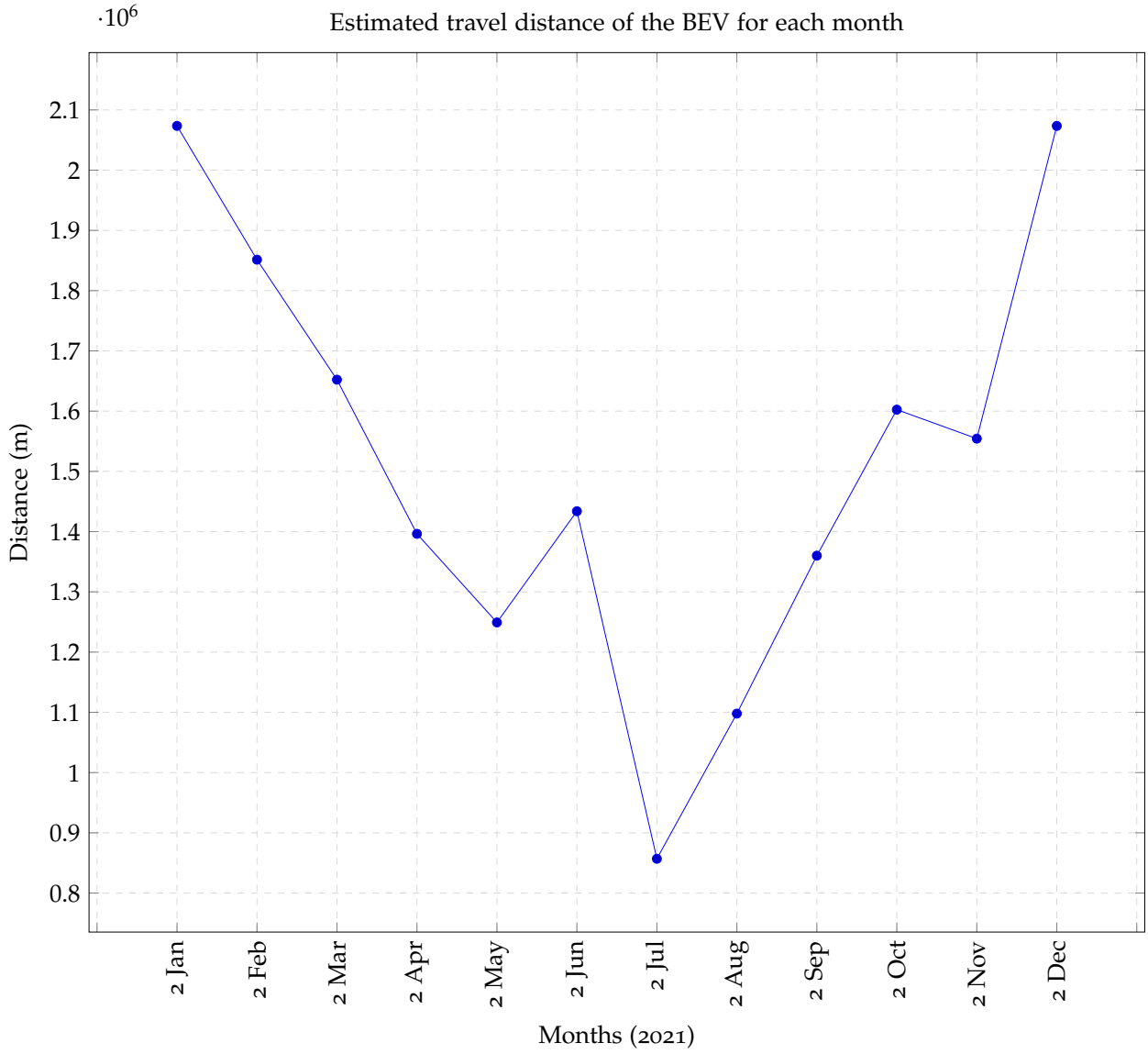


Figure 29: Multi-day results for every month in 2021.

5.7 SUMMARY

Having a solar panel that can tilt can be beneficial, especially in the mornings and late day, but care should be taken when implementing this. Practically the vehicle can flip with wind from the side when the vehicle is lightweight, and the solar panel is tilted so that it intercepts the wind. The optimisation model is practical for single-day optimisation but may not solve to optimality for multi-day optimisation. Warm-starting the model with the solution without loops had a significant impact on the computation time, along with making the linear piecewise constraints depend on the route chosen, e.g. the velocity and duration is only a convex combination when the route was selected. The energy graphs show it requires more points at low speeds to minimise the actual energy error. The optimisation model depends heavily on a solar-powered vehicle simulation, and improvements to the simulation will generate more practical energy graphs.

The optimisation can indicate how many loops should be taken each day, each route in the model corresponds to the number of loops that can be driven. Weather data used in this optimisation run were obtained using `solarpy` for clear sky radiation and `Weatherbit` for weather forecasts.

Overall the model can assist in the decision making regarding the race strategy for Sasol Solar Challenge, or similar events.

6

CONCLUSION AND FUTURE WORK

6.1 SUMMARY OF CONTRIBUTIONS

This study focused on model implementations to optimise route planning for Battery Electric Vehicles (BEVs). Simulations and optimisation models were described from first principles, but little detail was given on the effect various factors had on the performance of the BEV. It was more important to describe a generalised model that accounts for most factors. Two different models were proposed in the thesis: the path-based and flow conservation models. The flow conservation model describes the model with fewer binary variables at the expense of more BigM constraints, whereas the path-based model requires binary variables for all possible paths. The path-based model has the benefit of being used as a heuristic since the number of paths can be limited; the paths are determined by a k-shortest path algorithm in this thesis which is a generalised form of Dijkstra's shortest path algorithm. An algorithm that shows the parallel nature of the path-based model is given, but there is no real computational benefit when the number of paths is more than the parallel processors. A smaller formulation, such as the flow conservation model, is more beneficial.

Two different methods for modelling the energy graphs for each segment are used, a rectangular method and a triangle method. The triangle method divides the three-dimensional surface into triangles. Each triangle has a selection variable associated with it. Any point on the triangle can be written as a convex combination of the associated points of the triangle. The triangle method is described by Special Ordered Set of Type 3 (SOS3). At the time of writing, Mixed Integer Linear Programming (MILP) solvers did not have any speed-ups when using SOS3. The rectangular method is described by multiple Special Ordered Set of Type 2 (SOS2). The three-dimensional graph is divided into different rectangles, where inclusion constraints define the widths.

The generalised path-model can be applied to similar problems, such as the Sasol Solar Challenge with slight modification for the original path-based model. A case study for the Sasol Solar Challenge is provided, with specific North-West University vehicle features, such as solar panel tilt. The path-based model is adapted to specify multiple days, with the total energy remaining in the battery carrying over to the next day, with a starting and ending time for each day. After finding a solution, the number of loops and velocity on each segment can be provided as a race strategy.

The models can be used to improve the design of a solar vehicle by testing the effect of various vehicle parameters on different terrains and weather conditions. There are some notable observations while solving the different datasets; more details are given in the next section.

6.2 IMPORTANT OBSERVATIONS

One of the most notable observations was that the runtime and solution quality of datasets varies with different weather conditions when the BEV had a solar panel attached. When it is summer in the general vicinity captured by the dataset, the model solved faster compared to when it is

winter; this is likely because of lower overall irradiation over a day. With higher overall irradiation, the BEV can travel at higher speeds during day time.

When the road segments (or arcs in the case of the flow conservation model) are longer than 50 km, solutions that might exist are overlooked because of a constant velocity throughout the segment. Using finer segment granularity with some datasets showed improved solution accuracy. Initially, the datasets were solved with twelve velocity steps, but some datasets were infeasible in such conditions because of the coarse granularity at low speeds. Increasing the velocity steps resulted in longer runtimes, but at least it produced a solution. Another issue with long segments is the risk of increasing the energy consumed's error on those segments. There are two ways to compare the model validity and the energy used on a segment; one way is to compute error on the expected results of the model, and the other is to compare it to real-world data. Within this thesis, the error was computed for the model's expected results, which is a good starting point when real-world data is unavailable. As the time of day change, irradiation on the segment changes; this is not accounted for in the models directly. The mid longitude and latitude of a segment, with the start time, is used to determine the irradiation. This approach requires short segments to improve accuracy compared to granular simulations that track the segment's longitude and latitude as the vehicle progresses. The energy error compared to the expected model output is acceptable for both the triangle-and block-based energy graphs, usually and average energy error under 5%.

The electronics, such as the vehicle's motor controller and lights, were not accounted for; additional constant energy leaches can be defined within the simulation model but were left out. The model supports charging stations that allow various charge functions. In some datasets, charging stations were included; this was especially useful when the BEV did not have a solar panel attached.

6.3 FUTURE WORK

The thesis shows some useful tools for route planning, but as mentioned, the weather conditions can influence the overall runtime significantly. The more applicable use-case for such an implementation is when used as a decision support tool for design decisions. Practical trade-off studies can be done by using this tool as a benchmark with different terrains, vehicle characteristics and weather conditions.

The input data is modelled as graphs, making it trivial to transform the implementation into a solution for various applications, such as route planning for gliders. Gliders also have multiple routes that can be followed when considering weather conditions, and thermal columns can be modelled in a similar way as charging stations. Future work includes simulating and applying these model implementations to gliders. Optimisation models described in the thesis should also be able to assist with the design decisions of gliders.

The model described in Chapter 5 is a variation of the general path-based model in Chapter 3; future work includes testing the model implementation against real-world results. The BEV simulation and characteristics will be refined in the next Sasol Solar Challenge to improve result accuracy.

BIBLIOGRAPHY

[1] "Department: Energy: Republic of south africa." [Online]. Available: http://www.energy.gov.za/files/media/Energy_Balances.html

[2] J. Conti, P. Holtberg, J. Diefenderfer, A. LaRose, J. T. Turnure, and L. Westfall, "International energy outlook 2016 with projections to 2040," USDOE Energy Information Administration (EIA), Washington, DC, United States, Tech. Rep., 2016.

[3] S. Pelletier, O. Jabali, G. Laporte, and M. Veneroni, "Battery degradation and behaviour for electric vehicles: Review and numerical analyses of several models," *Transportation Research Part B: Methodological*, vol. 103, pp. 158–187, 2017.

[4] E. A. Grunditz and T. Thiringer, "Performance analysis of current bevs based on a comprehensive review of specifications," *IEEE Transactions on Transportation Electrification*, vol. 2, no. 3, pp. 270–289, 2016.

[5] B. A. Davis and M. A. Figliozzi, "A methodology to evaluate the competitiveness of electric delivery trucks," *Transportation Research Part E: Logistics and Transportation Review*, vol. 49, no. 1, pp. 8–23, 2013.

[6] J. Van Duin, L. A. Tavasszy, and H. Quak, "Towards e (lectric)-urban freight: first promising steps in the electric vehicle revolution," 2013.

[7] M. Schiffer, S. Stütz, and G. Walther, "Are ecvs breaking even?-competitiveness of electric commercial vehicles in medium-duty logistics networks," *Technical Report Working Paper*, no. OM-02/2016, 2016.

[8] C. Kanumilli, A. Singh, A. Ganesh, and M. Srinivas, "Plug in electric solar vehicle," in *2016 Biennial International Conference on Power and Energy Systems: Towards Sustainable Energy (PESTSE)*. IEEE, 2016, pp. 1–4.

[9] K. C. Prajapati, R. Patel, and R. Sagar, "Hybrid vehicle: A study on technology," *International Journal of Engineering Research & Technology (IJERT)*, vol. 3, no. 12, p. 8, 2014.

[10] A. Artmeier, J. Haselmayr, M. Leucker, and M. Sachenbacher, "The shortest path problem revisited: Optimal routing for electric vehicles," in *Annual conference on artificial intelligence*. Springer, 2010, pp. 309–316.

[11] J. Eisner, S. Funke, and S. Storandt, "Optimal route planning for electric vehicles in large networks," in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[12] S. Mehar, S. M. Senouci, and G. Rémy, "Ev-planning: Electric vehicle itinerary planning," in *2013 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, vol. 1. IEEE, 2013, pp. 1–5.

[13] U. F. Siddiqi, Y. Shiraishi, and S. M. Sait, "Multi-constrained route optimization for electric vehicles (evs) using particle swarm optimization (pso)," in *2011 11th International Conference on Intelligent Systems Design and Applications*. IEEE, 2011, pp. 391–396.

[14] T. M. Sweda and D. Klabjan, "Finding minimum-cost paths for electric vehicles," in *2012 IEEE International Electric Vehicle Conference*. IEEE, 2012, pp. 1–4.

[15] M. Baum, J. Dibbelt, T. Pajor, and D. Wagner, "Energy-optimal routes for electric vehicles," in *Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems*, 2013, pp. 54–63.

[16] P. Toth and D. Vigo, *The vehicle routing problem*. Society for Industrial and Applied Mathematics, 2002.

[17] S. Erdoğan and E. Miller-Hooks, "A green vehicle routing problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 48, no. 1, pp. 100–114, 2012.

[18] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231–247, 1992.

[19] G. Laporte, Y. Nobert, and M. Desrochers, "Optimal routing under capacity and distance restrictions," *Operations research*, vol. 33, no. 5, pp. 1050–1073, 1985.

[20] A. Froger, J. E. Mendoza, O. Jabali, and G. Laporte, "Improved formulations and algorithmic components for the electric vehicle routing problem with nonlinear charging functions," *Computers & Operations Research*, vol. 104, pp. 256–294, 2019.

[21] T. Korkmaz and M. Krunz, "Multi-constrained optimal path selection," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 2. IEEE, 2001, pp. 834–843.

[22] R. Bellman, "On a routing problem," *Quarterly of applied mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[23] L. R. Ford Jr, "Network flow theory," Rand Corp Santa Monica Ca, Tech. Rep., 1956.

[24] D. B. Johnson, "Efficient algorithms for shortest paths in sparse networks," *Journal of the ACM (JACM)*, vol. 24, no. 1, pp. 1–13, 1977.

[25] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[26] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck, "Customizable route planning," in *International Symposium on Experimental Algorithms*. Springer, 2011, pp. 376–387.

[27] Y. Zhang, Q. Ma, X. Zhang, M. Gao, P. Yang, H. He, X. Hu, and B. Aliya, "Integrated route and charging planning for electric vehicles considering nonlinear charging functions," in *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*. IEEE, 2018, pp. 660–665.

[28] R. Chen, X. Liu, L. Miao, and P. Yang, "Electric vehicle tour planning considering range anxiety," *Sustainability*, vol. 12, no. 9, p. 3685, 2020.

[29] K. Sörensen and F. Glover, "Metaheuristics," *Encyclopedia of operations research and management science*, vol. 62, pp. 960–970, 2013.

[30] A. S. Fraser, "Simulation of genetic systems by automatic digital computers i. introduction," *Australian Journal of Biological Sciences*, vol. 10, no. 4, pp. 484–491, 1957.

[31] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[32] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search," in *Handbook of metaheuristics*. Springer, 2003, pp. 320–353.

[33] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, vol. 2. IEEE, 1999, pp. 1470–1477.

[34] J. Kennedy, "Particle swarm optimization," *Encyclopedia of machine learning*, pp. 760–766, 2010.

[35] J. Hartmanis and R. E. Stearns, "On the computational complexity of algorithms," *Transactions of the American Mathematical Society*, vol. 117, pp. 285–306, 1965.

[36] A. M. Turing, "Computing machinery and intelligence," in *Parsing the Turing Test*. Springer, 2009, pp. 23–65.

[37] R. E. Ladner, "On the structure of polynomial time reducibility," *Journal of the ACM (JACM)*, vol. 22, no. 1, pp. 155–171, 1975.

[38] D. E. Knuth, "Big omicron and big omega and big theta," *ACM Sigact News*, vol. 8, no. 2, pp. 18–24, 1976.

[39] ——, *The art of computer programming*. Pearson Education, 1997, vol. 3.

[40] R. E. Tarjan, "Amortized computational complexity," *SIAM Journal on Algebraic Discrete Methods*, vol. 6, no. 2, pp. 306–318, 1985.

[41] G. Dantzig, *Linear programming and extensions*. Princeton university press, 2016.

[42] D. Avis and V. Chvátal, "Notes on bland's pivoting rule," in *Polyhedral Combinatorics*. Springer, 1978, pp. 24–34.

[43] T. Terlaky, "A convergent criss-cross method," *Optimization*, vol. 16, no. 5, pp. 683–690, 1985.

[44] K. Fukuda and T. Terlaky, "Criss-cross methods: A fresh view on pivot algorithms," *Mathematical Programming*, vol. 79, no. 1-3, pp. 369–395, 1997.

[45] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[46] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[47] R. Gomory, "An algorithm for the mixed integer problem," RAND CORP SANTA MONICA CA, Tech. Rep., 1960.

[48] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," in *50 Years of Integer Programming 1958-2008*. Springer, 2010, pp. 105–132.

[49] J. E. Mitchell, "Branch-and-cut algorithms for combinatorial optimization problems," *Handbook of applied optimization*, vol. 1, pp. 65–77, 2002.

[50] C. Kwon, *Julia Programming for Operations Research*. Changhyun Kwon, 2019.

[51] C. D'Ambrosio, A. Lodi, and S. Martello, "Piecewise linear approximation of functions of two variables in milp models," *Operations Research Letters*, vol. 38, no. 1, pp. 39 – 46, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167637709001072

[52] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[53] J. L. Gross and J. Yellen, *Handbook of graph theory*. CRC press, 2004.

[54] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *Journal of the ACM (JACM)*, vol. 34, no. 3, pp. 596–615, 1987.

[55] G. S. Brodal, R. Fagerberg, U. Meyer, and N. Zeh, "Cache-oblivious data structures and algorithms for undirected breadth-first search and shortest paths," in *Scandinavian Workshop on Algorithm Theory*. Springer, 2004, pp. 480–492.

[56] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[57] S. S. Skiena, *The algorithm design manual*. Springer Science & Business Media, 1998, vol. 1.

[58] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, "The vehicle routing problem: State of the art classification and review," *Computers & Industrial Engineering*, vol. 99, pp. 300–313, 2016.

[59] D. Koller, P. Lindstrom, W. Ribarsky, L. F. Hodges, N. Faust, and G. Turner, "Virtual gis: A real-time 3d geographic information system," in *Proceedings Visualization'95*. IEEE, 1995, pp. 94–100.

[60] K.-T. Chang, "Geographic information system," *International Encyclopedia of Geography: People, the Earth, Environment and Technology*, pp. 1–10, 2016.

[61] T. D. Gillespie, *Fundamentals of vehicle dynamics*. Society of automotive engineers Warrendale, PA, 1992, vol. 400.

[62] J. Ruiz-Arias, D. Pozo-Vazquez, V. Lara Fanego, F. Santos-Alamillos, and J. Pescador, "A high-resolution topographic correction method for clear-sky solar irradiance derived with a numerical weather prediction model," *Journal of Applied Meteorology and Climatology*, vol. 50, pp. 2460–2472, 08 2011.

[63] P. Loutzenhiser, H. Manz, C. Felsmann, P. Strachan, T. Frank, and G. Maxwell, "Empirical validation of models to compute solar irradiance on inclined surfaces for building energy simulation," *Solar Energy*, vol. 81, no. 2, pp. 254–267, 2007.

[64] P. Ineichen, R. Perez, and R. Seals, "The importance of correct albedo determination for adequately modeling energy received by tilted surfaces," *Solar Energy*, vol. 39, no. 4, pp. 301 – 305, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038092X87800166

[65] T. Muneer, "Solar radiation and daylight models for energy efficient design of buildings. architectural press, oxford, uk." *Solar radiation and daylight models for energy efficient design of buildings. Architectural Press, Oxford, UK.*, 1997.

[66] D. Reindl, W. Beckman, and J. Duffie, "Evaluation of hourly tilted surface radiation models," *Solar Energy*, vol. 45, no. 1, pp. 9 – 17, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0038092X9090061G

[67] T. Klucher, "Evaluation of models to predict insolation on tilted surfaces," *Solar Energy*, vol. 23, no. 2, pp. 111 – 114, 1979. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0038092X79901105

[68] J. Davies and J. Hay, "Calculation of the solar radiation incident on an inclined surface," in *Proc. First Canadian Solar Radiation Data Workshop*, pp. 59–72.

[69] R. Perez, P. Ineichen, R. Seals, J. Michalsky, and R. Stewart, "Modeling daylight availability and irradiance components from direct and global irradiance," *Solar Energy*, vol. 44, no. 5, pp. 271 – 289, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0038092X9090055H

[70] B. Y. Liu and R. C. Jordan, "The interrelationship and characteristic distribution of direct, diffuse and total solar radiation," *Solar Energy*, vol. 4, no. 3, pp. 1 – 19, 1960. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0038092X60900621

[71] J. A. Duffie and W. A. Beckman, *Solar engineering of thermal processes*. John Wiley & Sons, 2013.

[72] C. Oosthuizen, B. Van Wyk, Y. Hamam, D. Desai, Y. Alayli, and R. Lot, "Solar electric vehicle energy optimization for the sasol solar challenge 2018," *IEEE Access*, vol. 7, pp. 175 143–175 158, 2019.

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

[74] D. Eppstein, "Finding the k shortest paths," *SIAM Journal on computing*, vol. 28, no. 2, pp. 652–673, 1998.

[75] M. Kerrisk, *The Linux programming interface: a Linux and UNIX system programming handbook*. No Starch Press, 2010.

[76] E. Guerrero Merino and M. A. Duarte-Mermoud, "Online energy management for a solar car using pseudospectral methods for optimal control," *Optimal Control Applications and Methods*, vol. 37, no. 3, pp. 537–555, 2016.

[77] E. Betancur, G. Osorio-Gómez, and J. C. Rivera, "Heuristic optimization for the energy management and race strategy of a solar car," *Sustainability*, vol. 9, no. 10, 2017. [Online]. Available: http://www.mdpi.com/2071-1050/9/10/1576

[78] O. K. Erol and I. Eksin, "A new optimization method: big bang–big crunch," *Advances in Engineering Software*, vol. 37, no. 2, pp. 106–111, 2006.

[79] Y. Shimizu, Y. Komatsu, M. Torii, and M. Takamuro, "Solar car cruising strategy and its supporting system," *JSAE review*, vol. 19, no. 2, pp. 143–149, 1998.

[80] P. Howlett, "A markov model for the stochastic optimal control of a solar powered car," in *Optimization Methods and Applications*. Springer, 2001, pp. 331–341.

[81] J. Boland, V. Gaitsgory, P. Howlett, and P. Pudney, "Stochastic optimal control of a solar car," in *Optimization and related topics*. Springer, 2001, pp. 71–81.

[82] A. Scheidegger and T. Liebling, "Energy management optimization for a solar vehicle," *Masters, Ecole Polytechnique Federale De Lausanne*, 2006.

[83] J. P. Vielma and G. L. Nemhauser, "Modeling disjunctive constraints with a logarithmic number of binary variables and constraints," *Mathematical Programming*, vol. 128, no. 1-2, pp. 49–72, 2011.

[84] J. Lee and D. Wilson, "Polyhedral methods for piecewise-linear functions i: the lambda method," *Discrete applied mathematics*, vol. 108, no. 3, pp. 269–285, 2001.

[85] R. G. Jeroslow and J. K. Lowe, "Modelling with integer variables," in *Mathematical Programming at Oberwolfach II*. Springer, 1984, pp. 167–184.

[86] J. P. Vielma, S. Ahmed, and G. Nemhauser, "Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions," *Operations research*, vol. 58, no. 2, pp. 303–315, 2010.

[87] C. Oosthuizen, B. Wyk, Y. Hamam, D. Desai, and Y. Alayli, "The use of gridded model output statistics (gmos) in energy forecasting of a solar car," *Energies*, vol. 13, p. 1984, 04 2020.

Appendices

A

PRACTICAL RESULTS

Table 16: Results for initial practical models.

Model	Runtime	Objective Value	Velocity Steps	%error	Relative Gap
FC-TRI	0.150743	59808.3	2	21.5209	0
FC-TRI	0.351878	39205.2	4	-5.03955	0
FC-TRI	0.818357	38514.4	8	-3.59834	0
FC-TRI	2.60214	38085.4	16	-1.33566	0
FC-TRI	6.5506	37934	32	-0.14608	7.0434e-05
FC-TRI	42.0892	37919.9	64	-0.0633541	0
P-TRI	0.1595	59808.3	2	21.5209	0
P-TRI	0.387745	39205.2	4	-5.03955	0
P-TRI	0.922885	38514.4	8	-3.59834	0
P-TRI	2.61813	38085.4	16	-1.33566	0
P-TRI	12.8757	37931.3	32	-0.16019	0
P-TRI	59.4802	37919.9	64	-0.0633541	0
FC-BLK	0.068187	59981.7	2	20.0669	0
FC-BLK	0.058246	39228.2	4	-4.91845	0
FC-BLK	0.103338	38542.4	8	-3.4687	0
FC-BLK	0.194955	38106.3	16	-1.24784	3.99844e-05
FC-BLK	0.16796	37934	32	-0.14608	0
FC-BLK	0.241658	37919.9	64	-0.0633541	0
P-BLK	0.05259	59981.7	2	20.0669	0
P-BLK	0.06235	39228.2	4	-4.91845	0
P-BLK	0.100923	38542.4	8	-3.4687	0
P-BLK	0.168376	38106.3	16	-1.24784	3.99844e-05
P-BLK	0.201486	37934	32	-0.14608	0
P-BLK	0.283281	37919.9	64	-0.0633541	0

B

VISUALISATIONS OF DATASETS USED FOR GENERAL MODELS

Visualisation of datasets used for general models are provided here. These datasets are not to scale. The complete datasets as `JSON` graphs can be found at BEVDatasets[1].

1 The above link is click-able. For completeness the text is also provided: https://github.com/baggins800/bevroutes

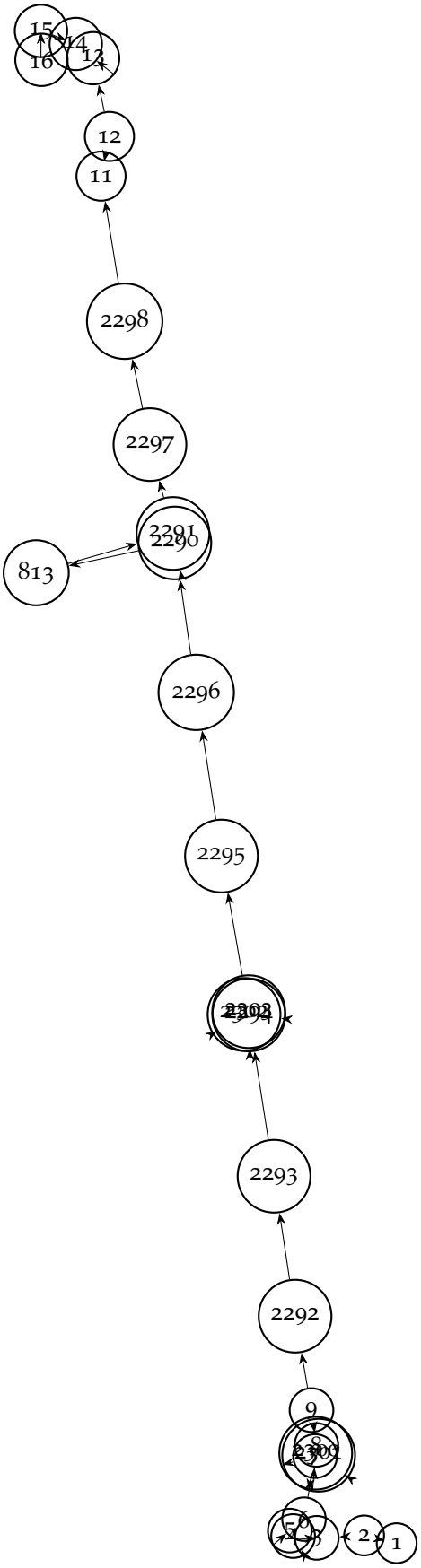


Figure 30: *small001* visualisation.

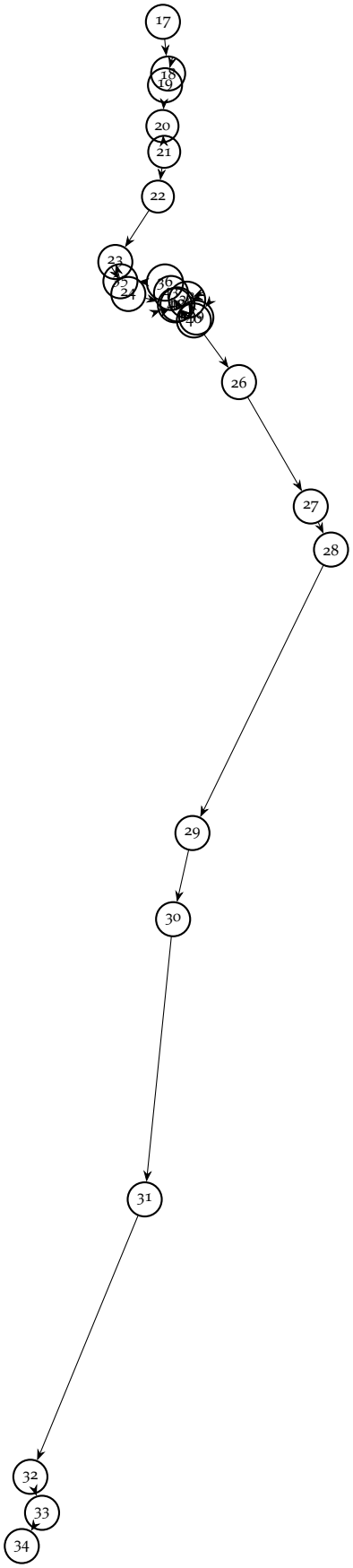


Figure 31: *smalloo2* visualisation.

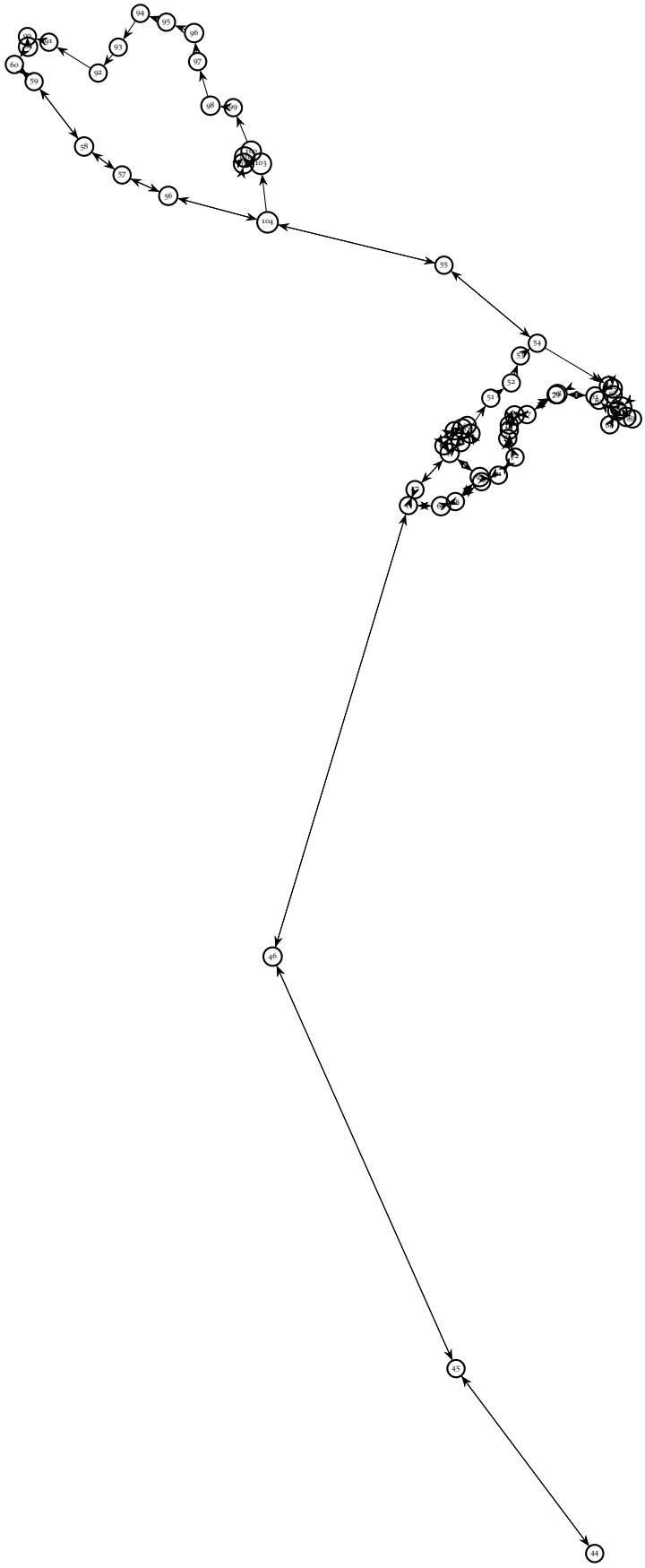


Figure 32: *small003* visualisation.

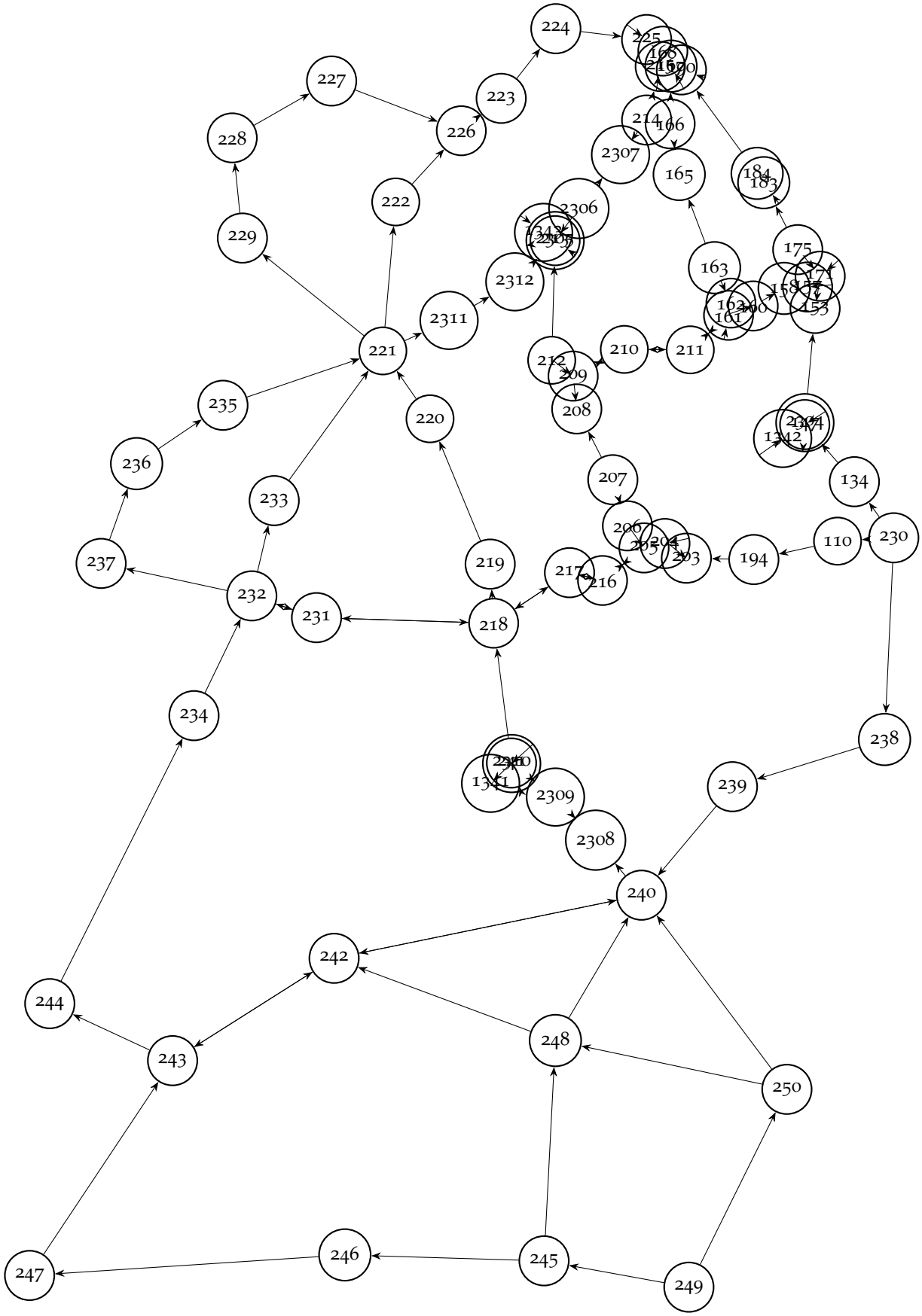


Figure 33: *smalloo4* visualisation.

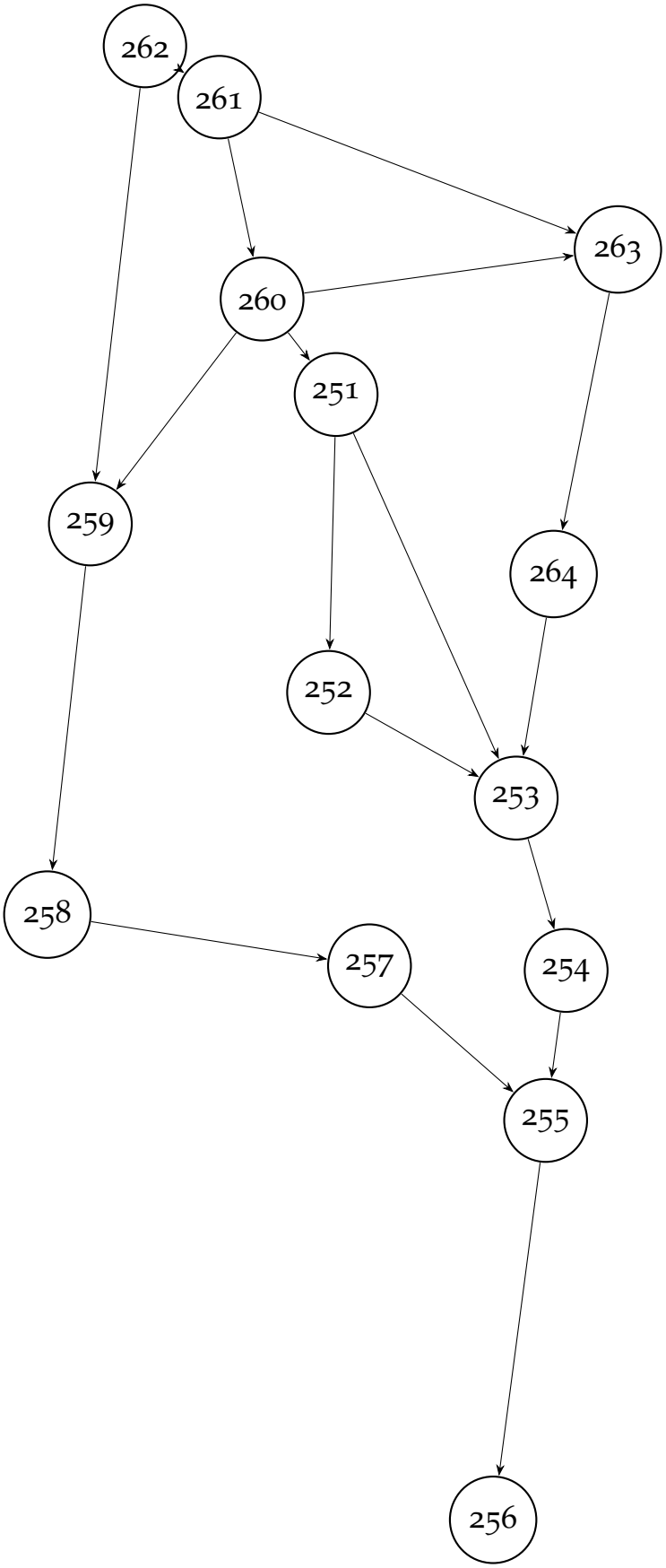


Figure 34: *small005* visualisation.

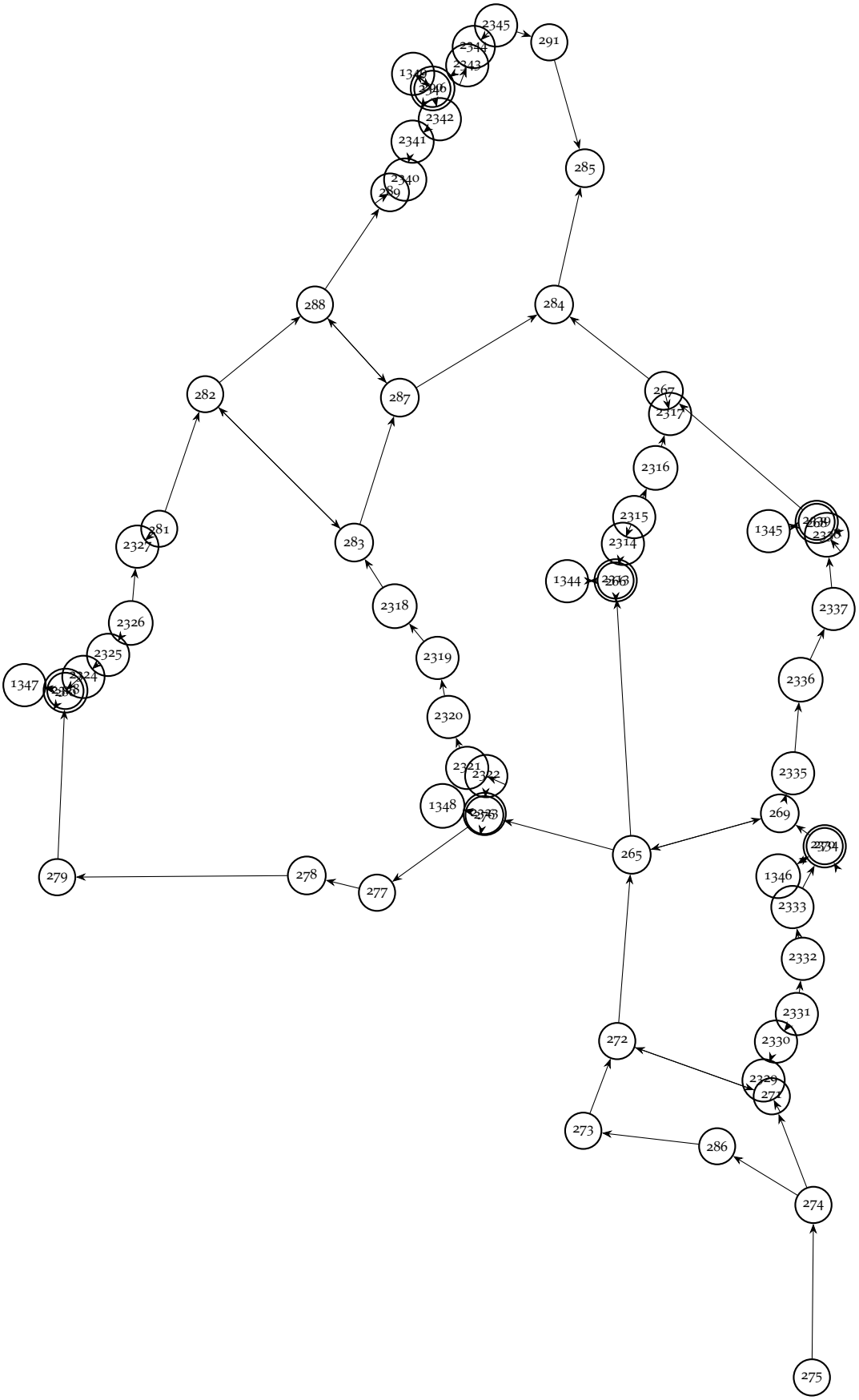


Figure 35: *smalloo6* visualisation.

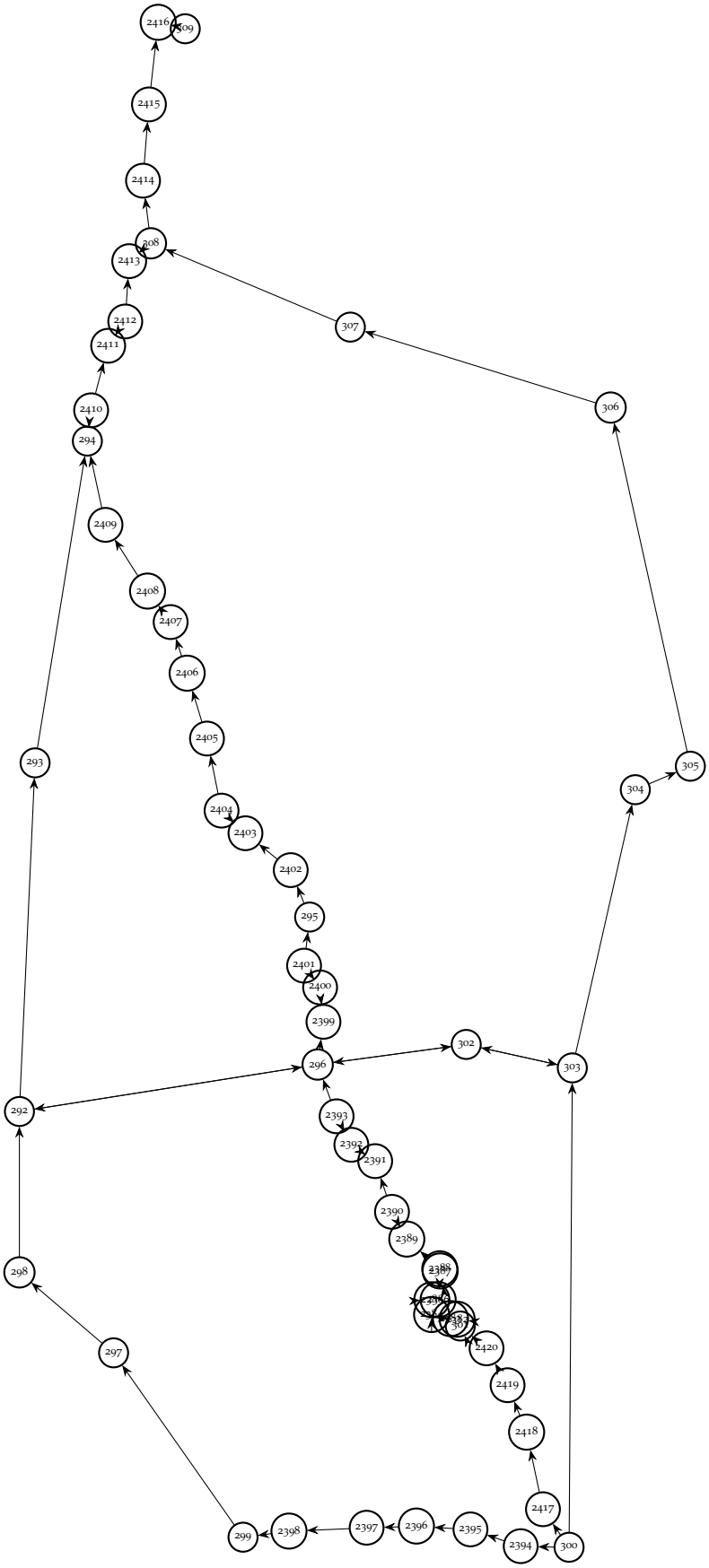


Figure 36: *small007* visualisation.

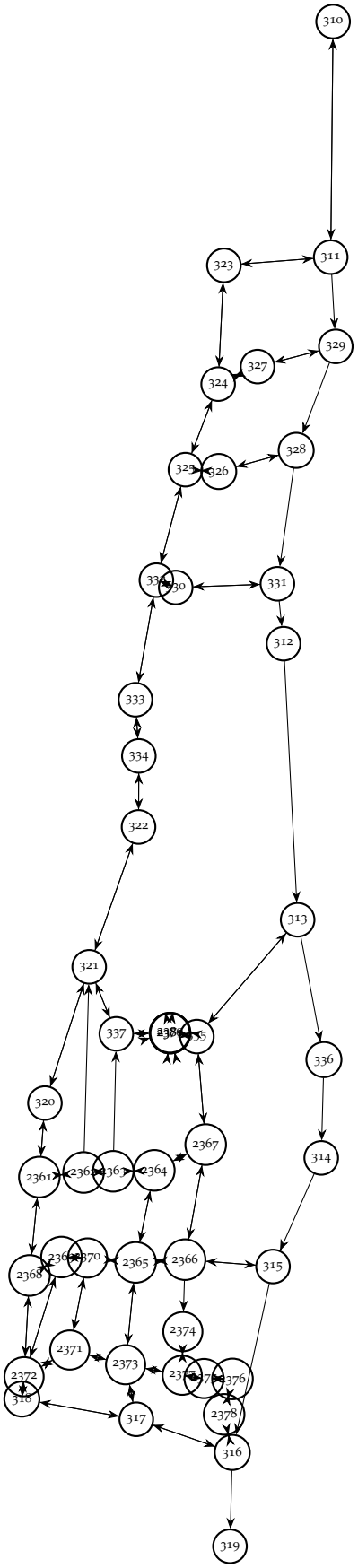


Figure 37: *smalloo8* visualisation.

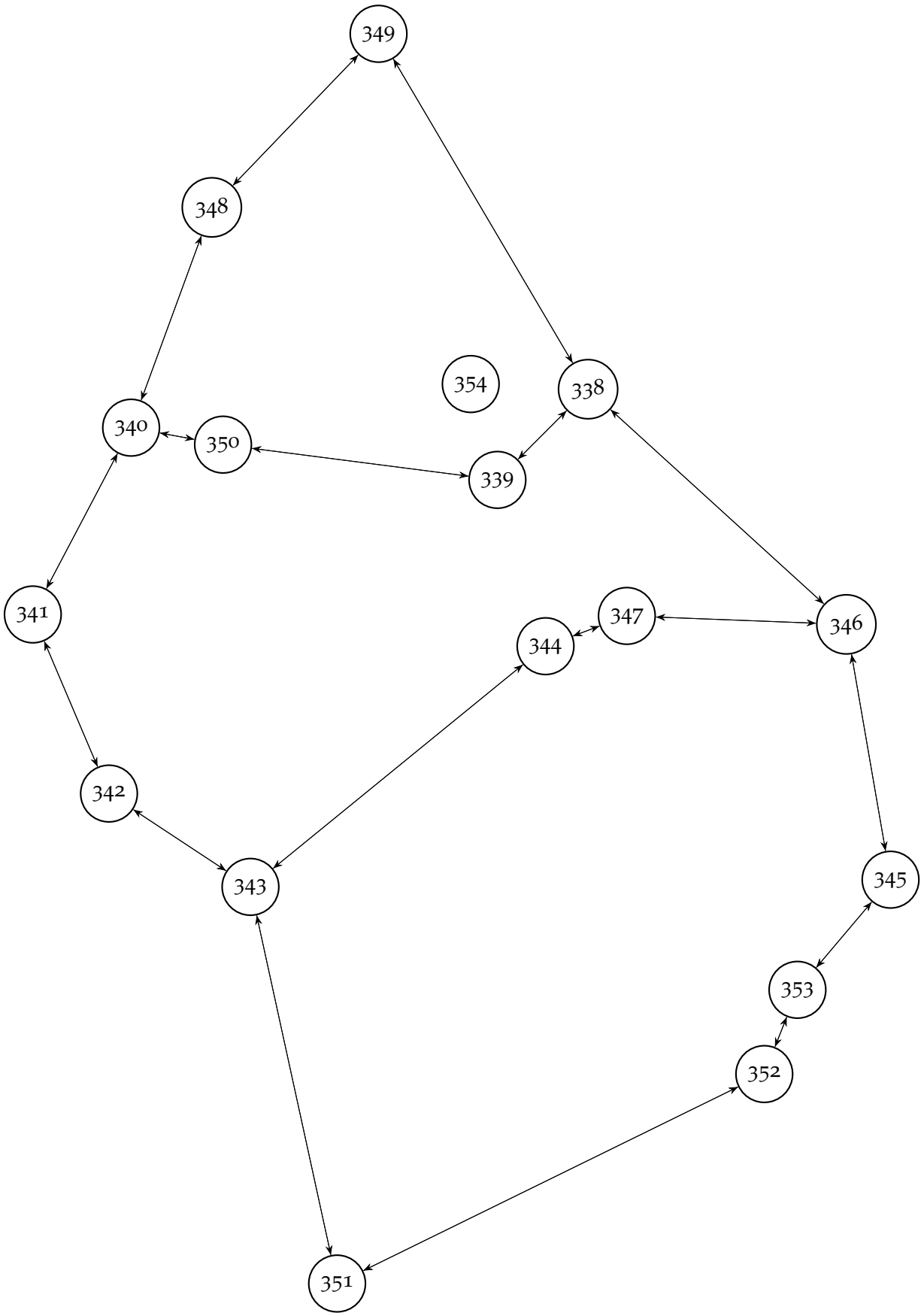


Figure 38: *smalloo9* visualisation.

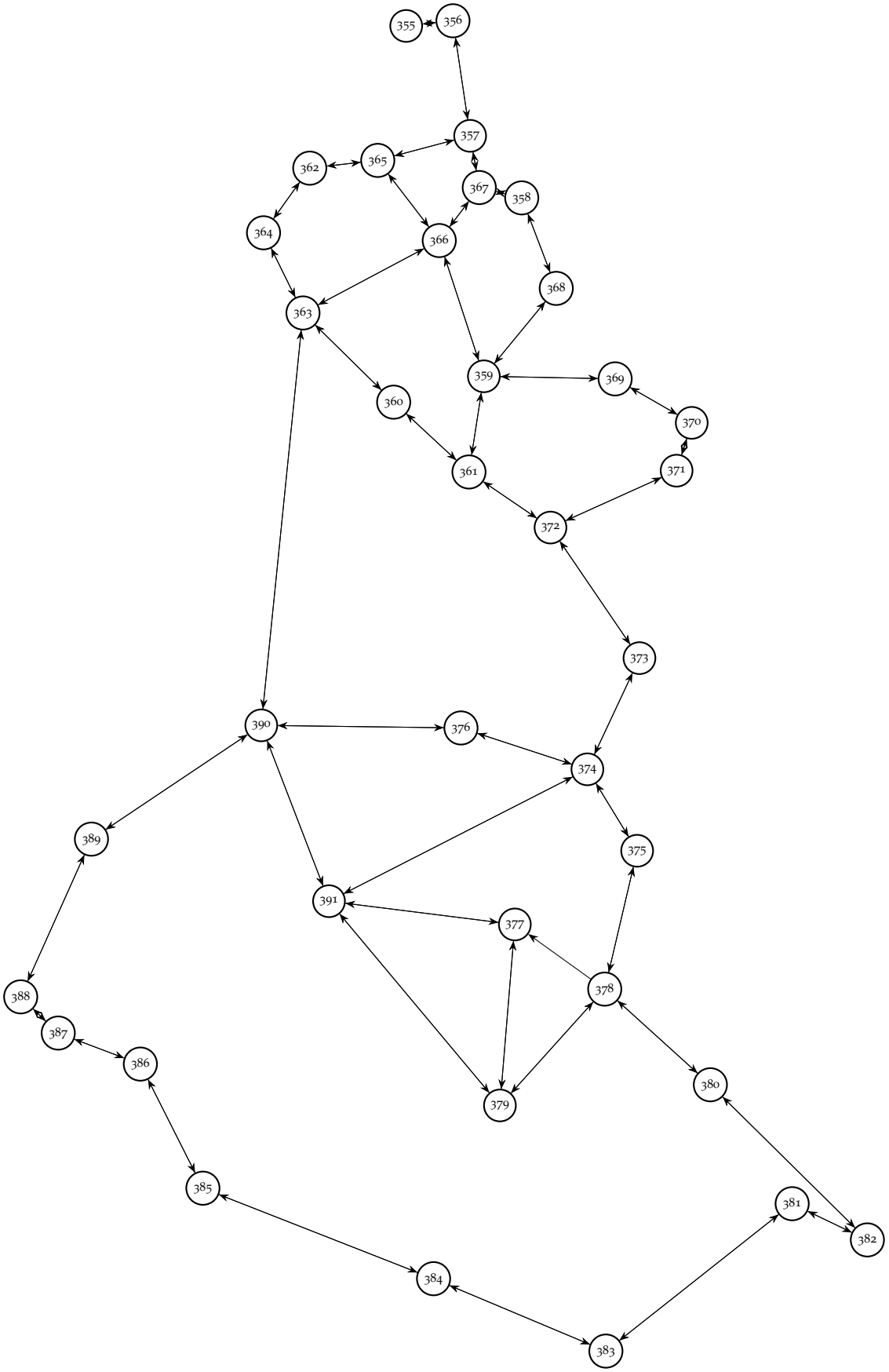


Figure 39: *smallo10* visualisation.

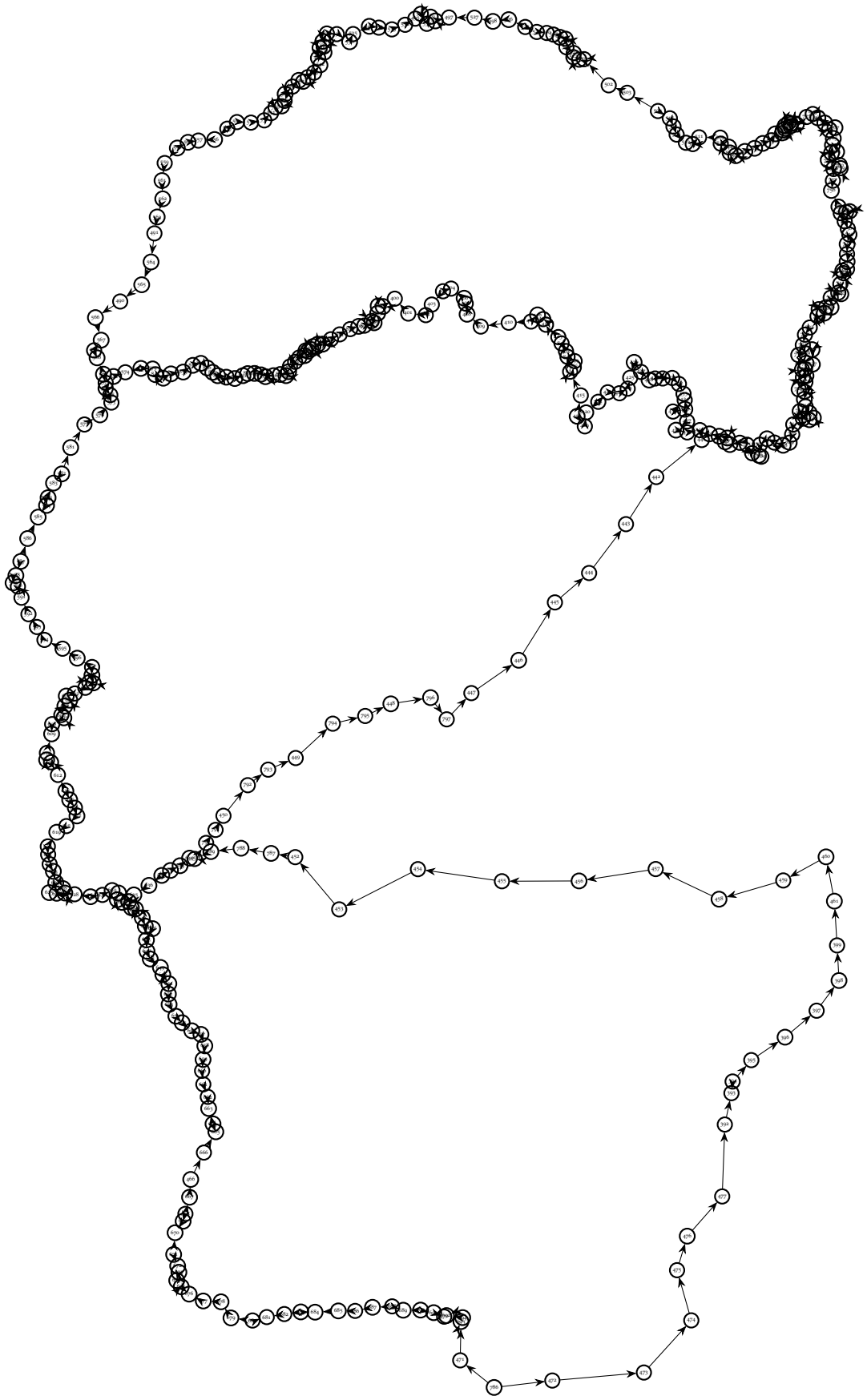


Figure 40: *mediumoo1* visualisation.

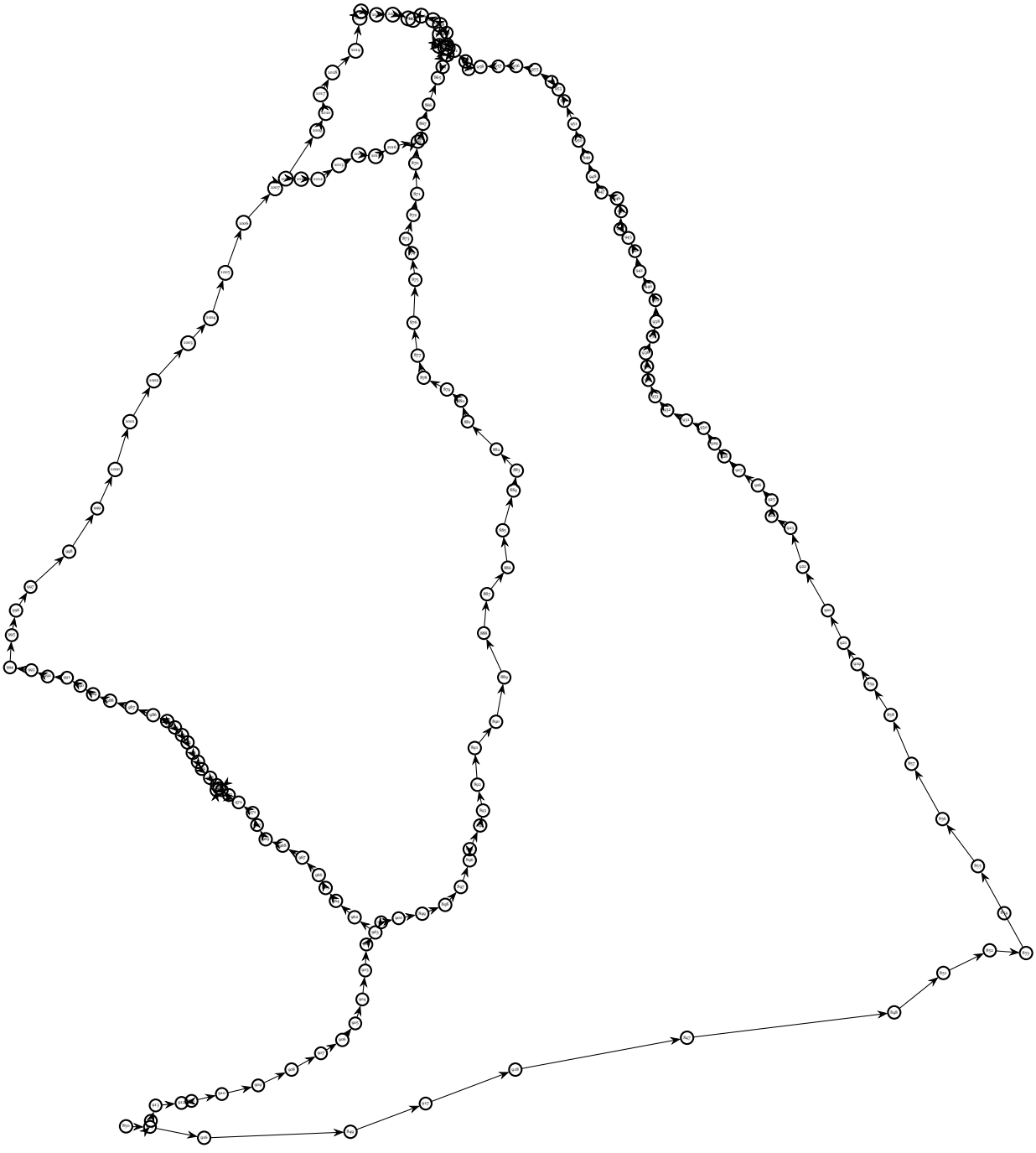


Figure 41: *mediumoo2* visualisation.

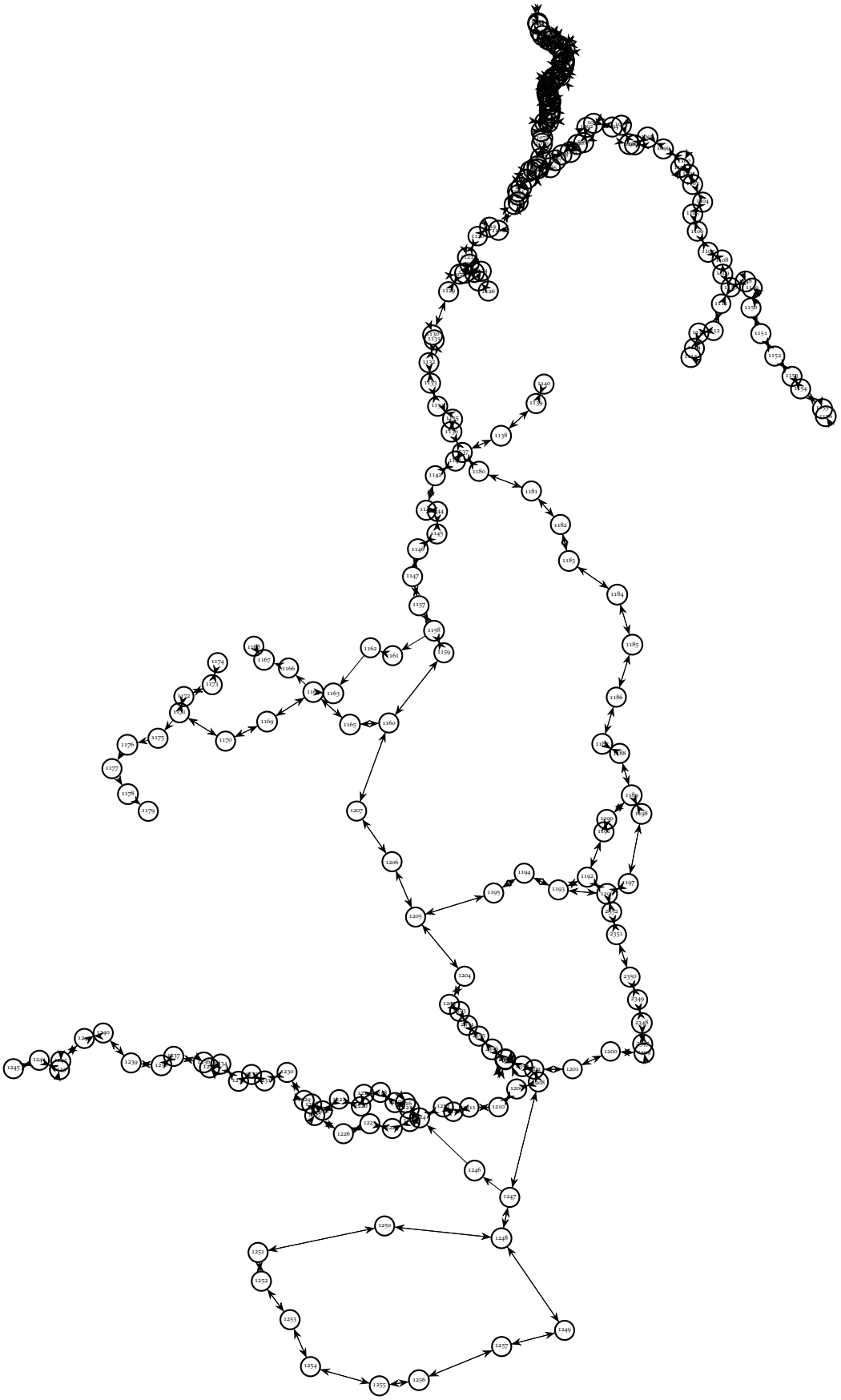


Figure 42: *mediumoo3* visualisation.

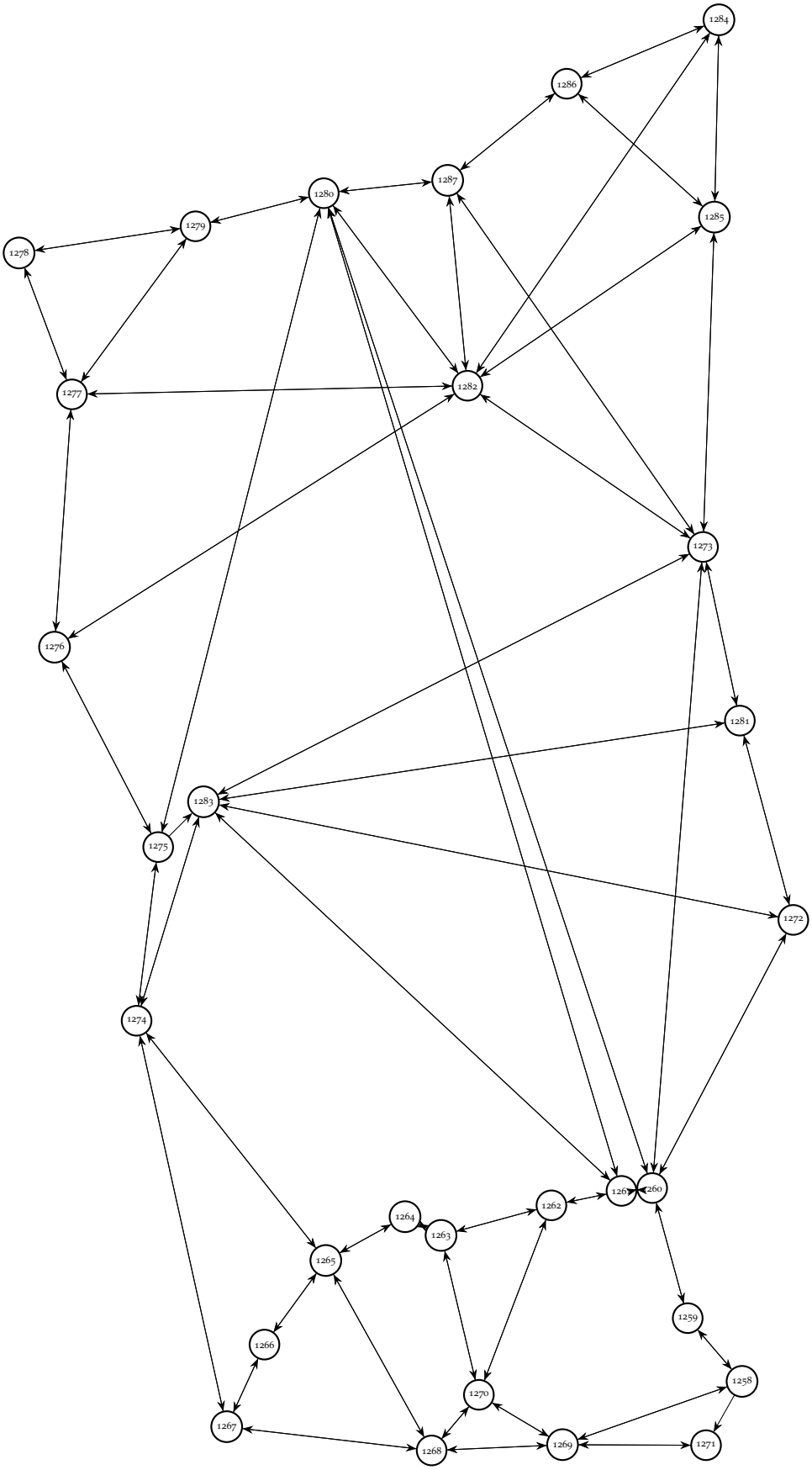


Figure 43: *medium004* visualisation.

C

VISUALISATIONS OF DATASETS USED FOR MULTI-DAY OPTIMISATION

Visualisation of datasets used for multi-day optimisation are provided here. These datasets are not to scale. The complete datasets as `JSON` graphs can be found at Multi-day optimisation datasets[1].

1 The above link is click-able. For completeness the text is also provided: https://github.com/baggins800/mdo

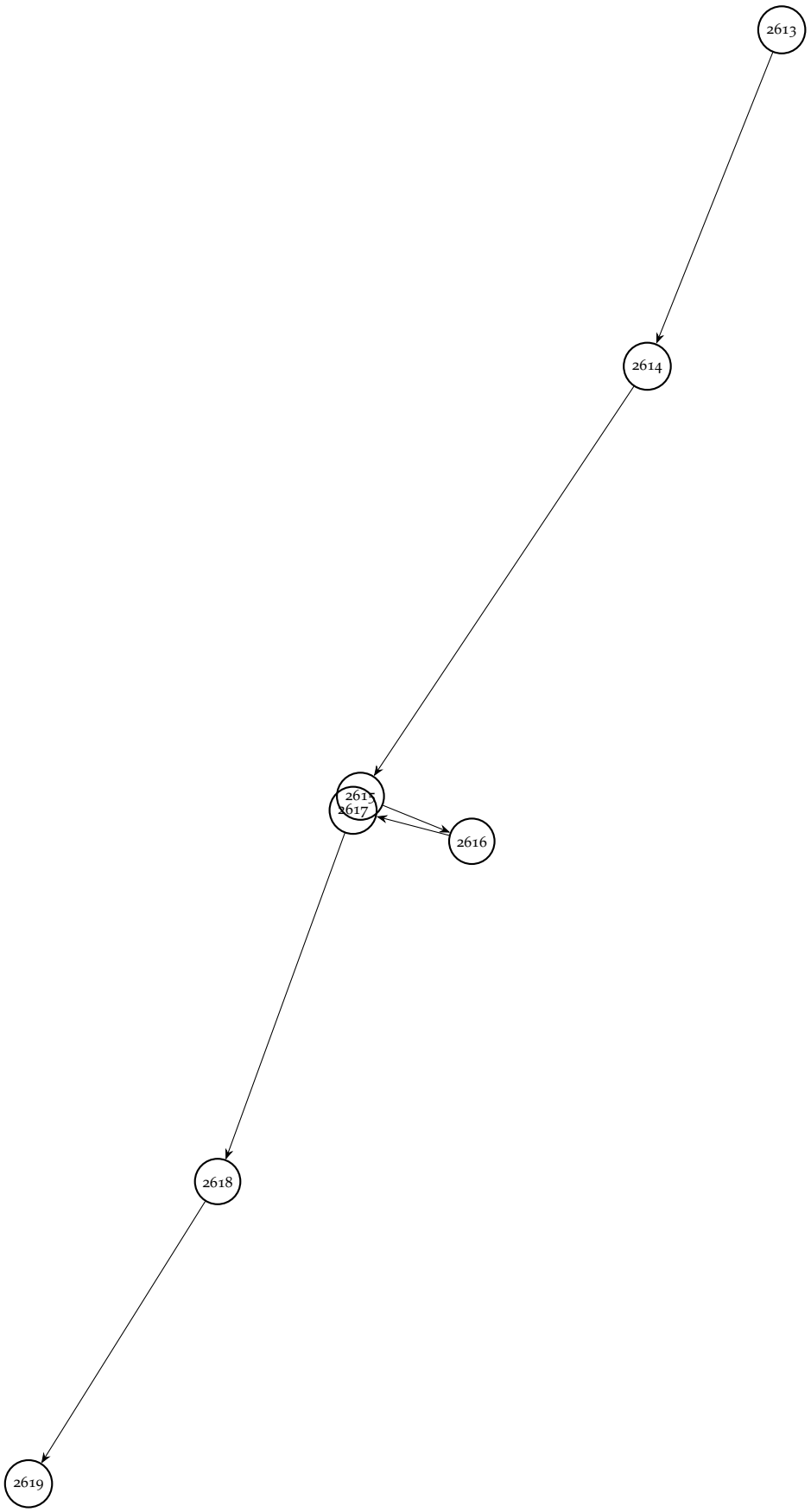


Figure 44: Day 1 graph visualisation.

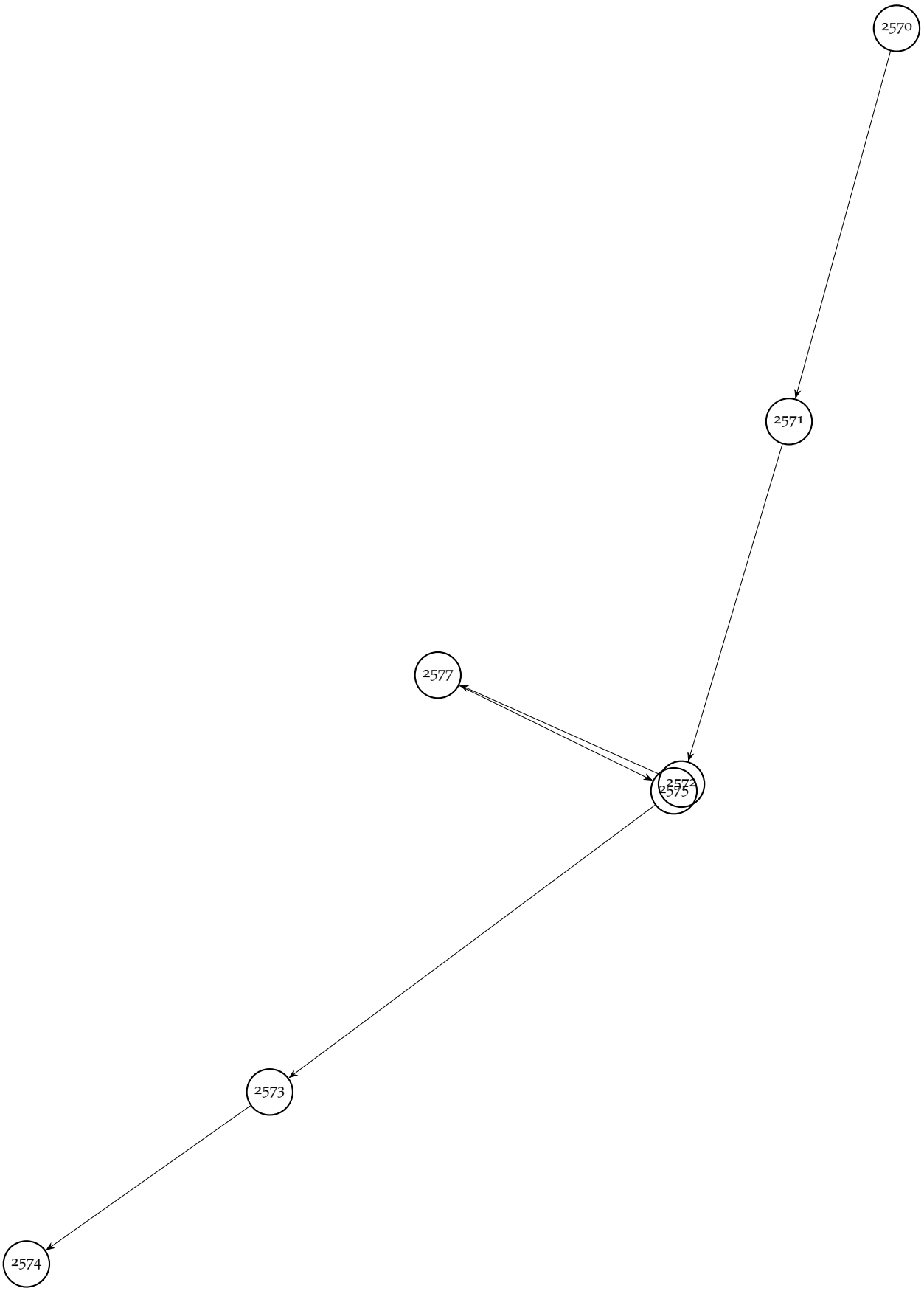


Figure 45: Day 2 graph visualisation.

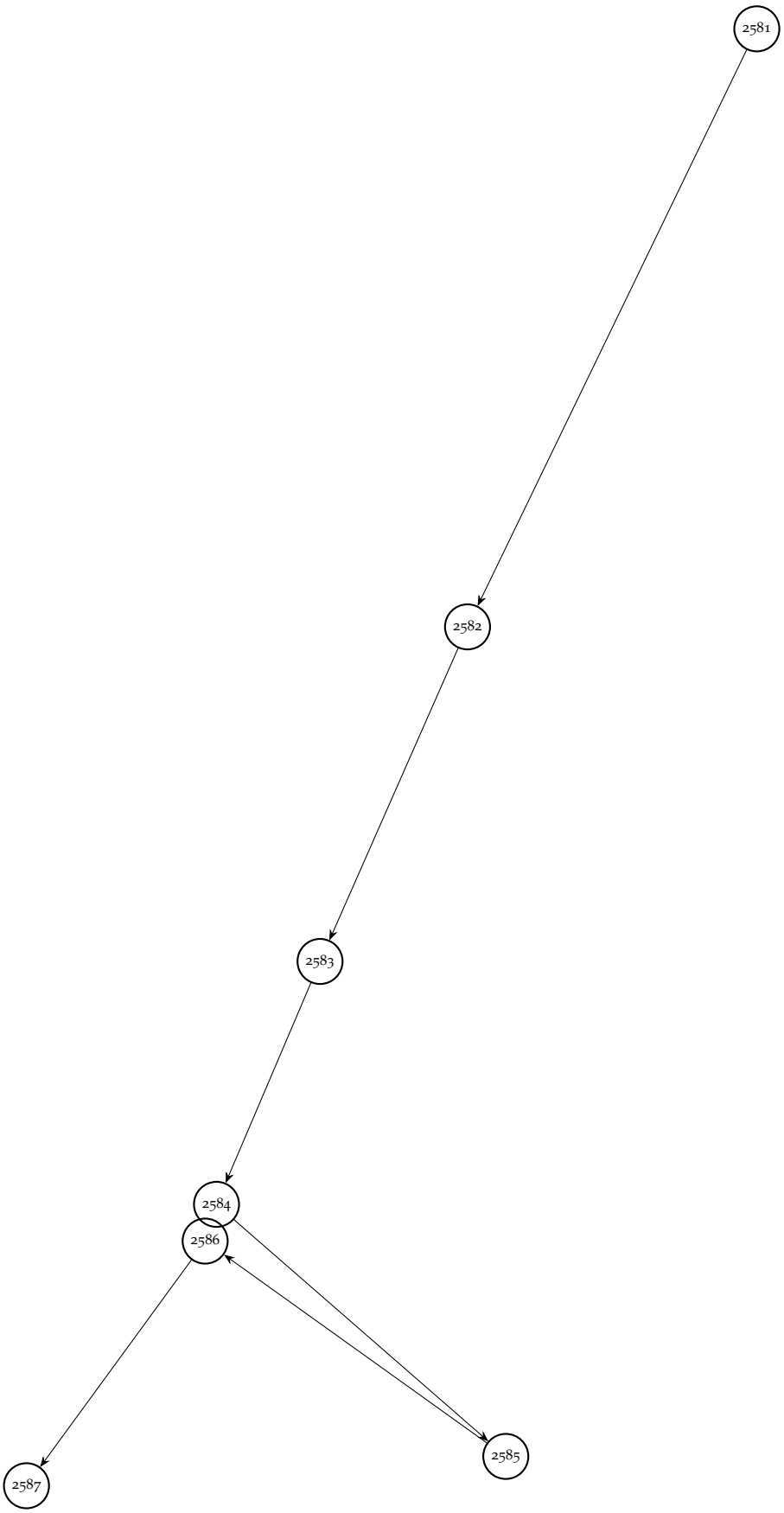


Figure 46: Day 3 graph visualisation.

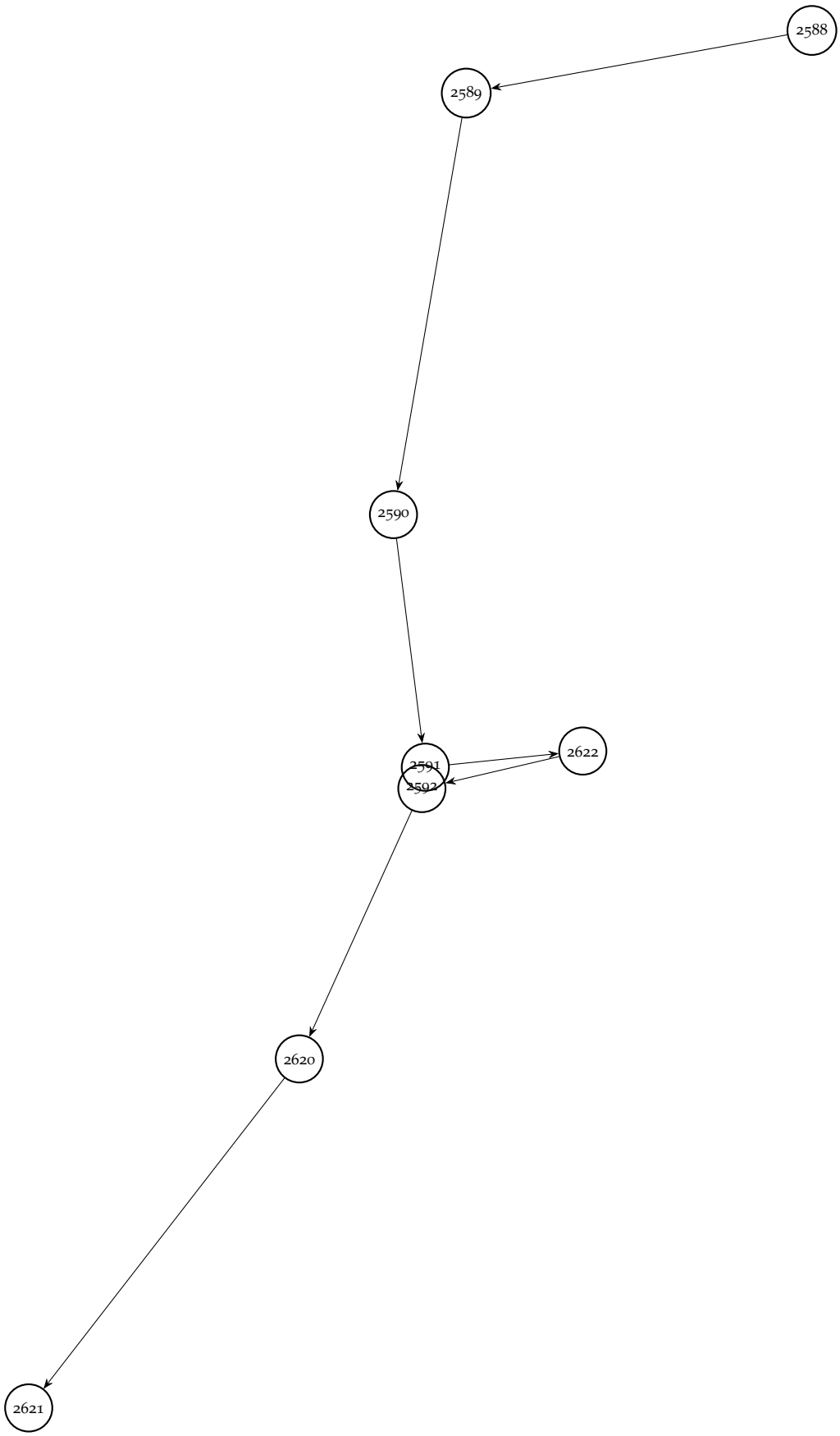


Figure 47: Day 4 graph visualisation.

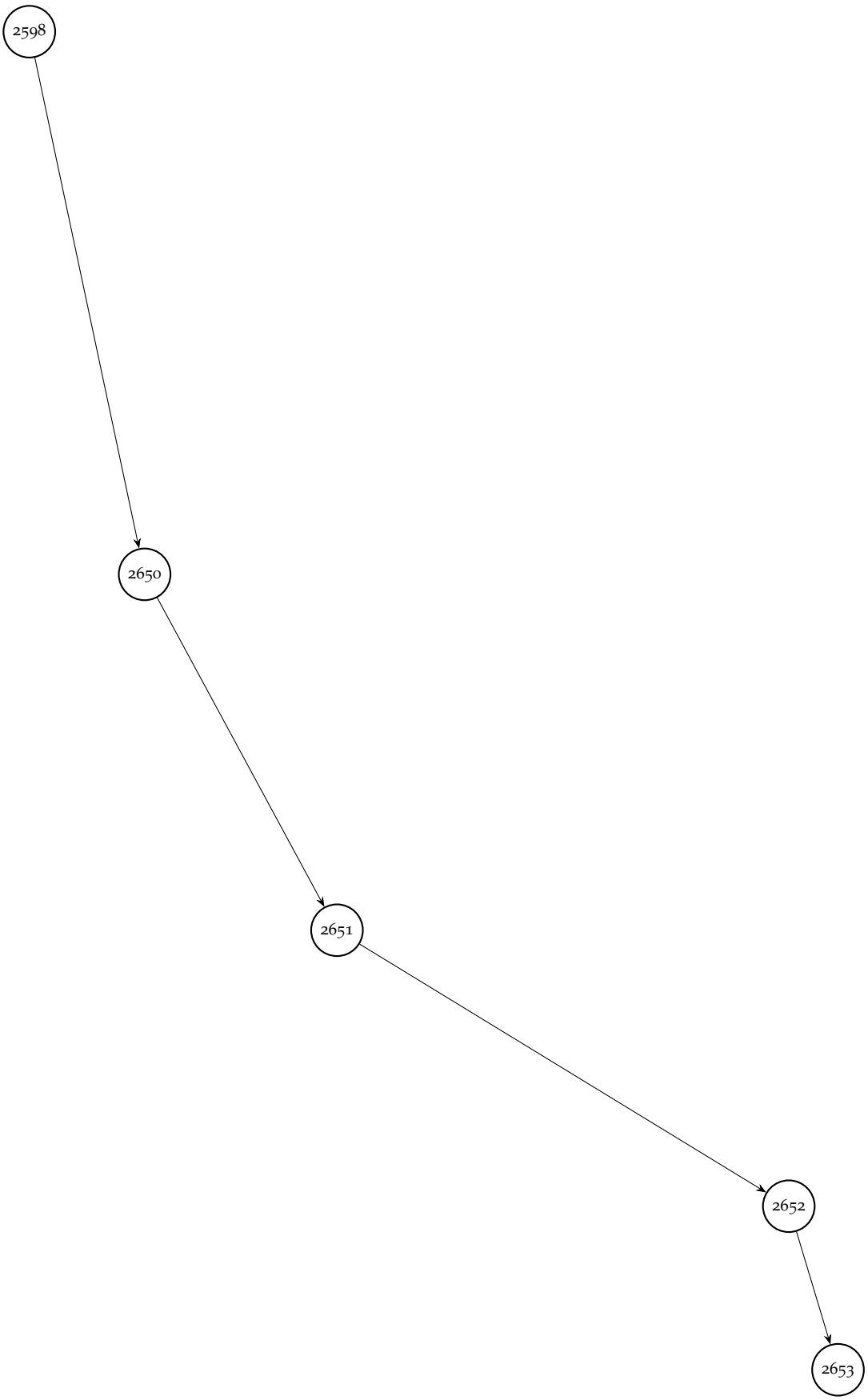


Figure 48: Day 5 graph visualisation.

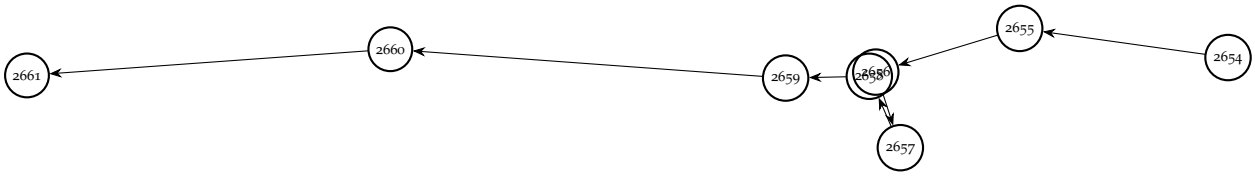


Figure 49: Day 6 graph visualisation.

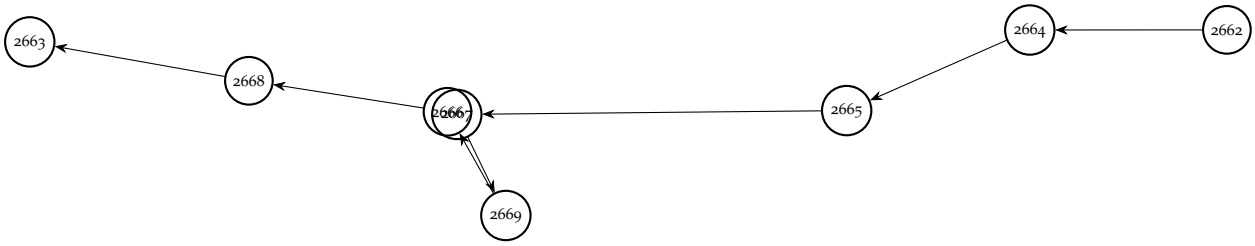


Figure 50: Day 7 graph visualisation.

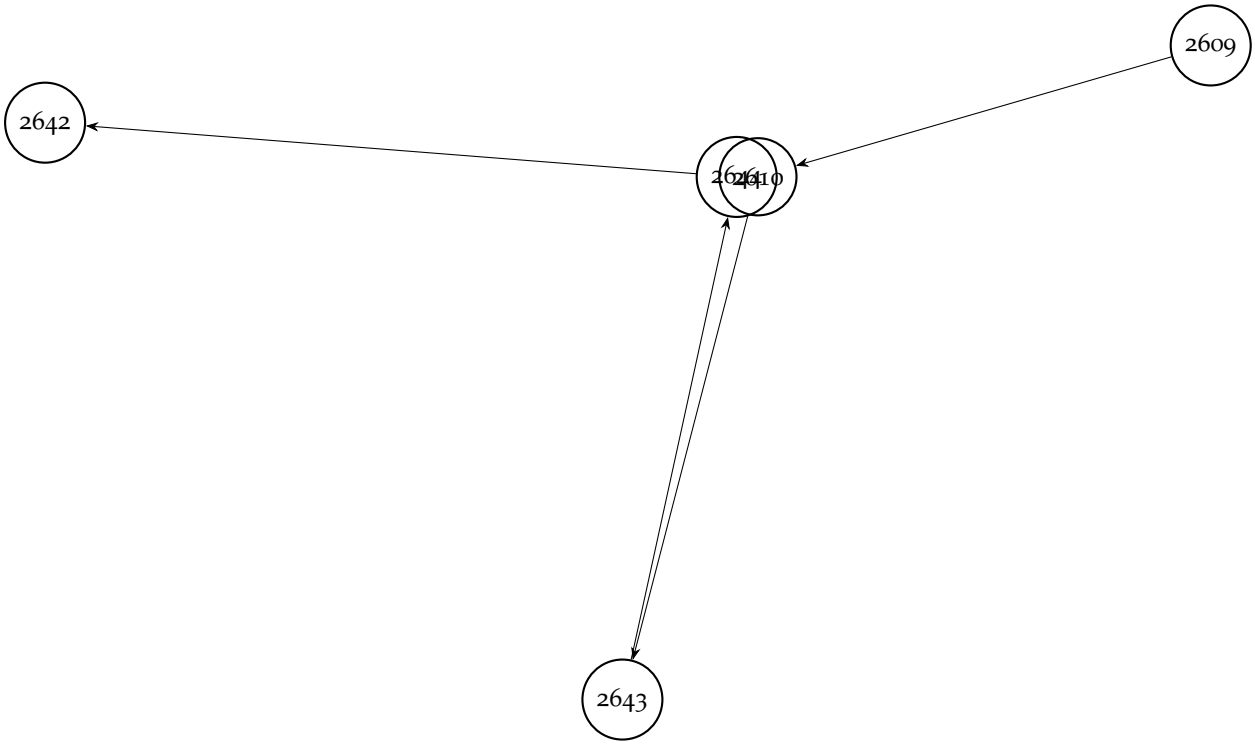


Figure 51: Day 8 graph visualisation.

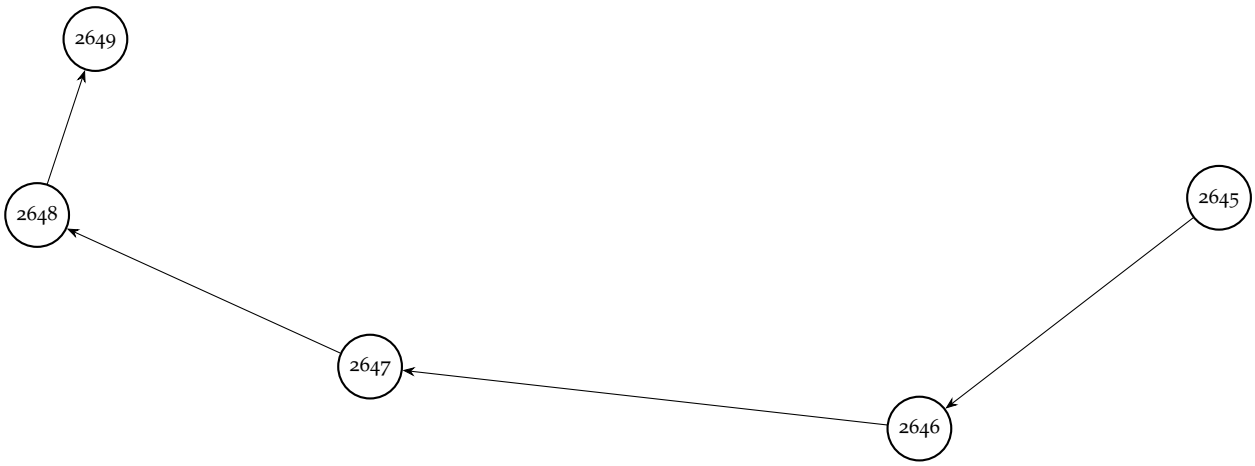
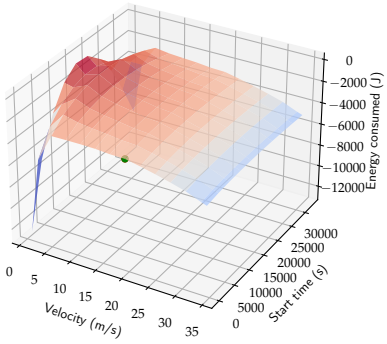
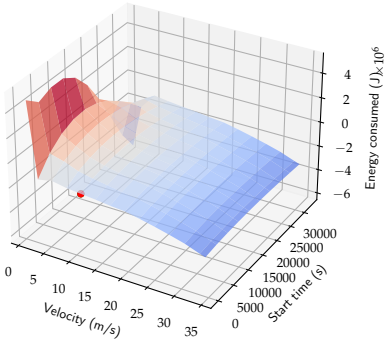


Figure 52: Day 9 graph visualisation.

(a) Energy graph for segment 45.

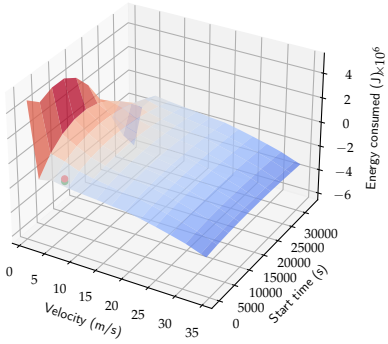
(b) Energy graph for segment 46.

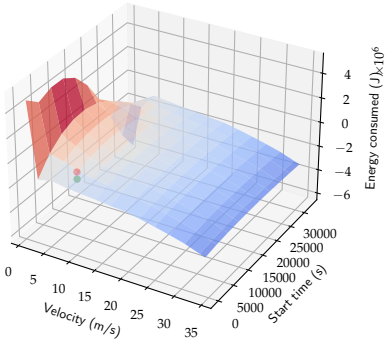




(c) Energy graph for segment 47.

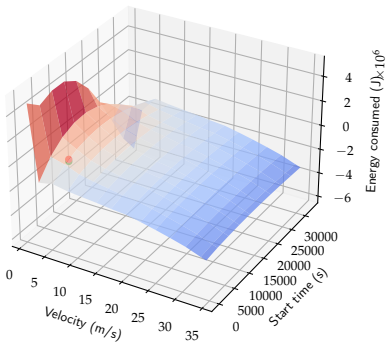
(d) Energy graph for segment 48.





(e) Energy graph for segment 49.

(f) Energy graph for segment 50.



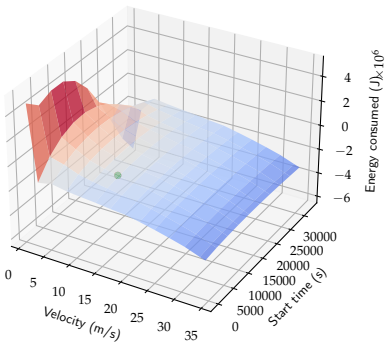
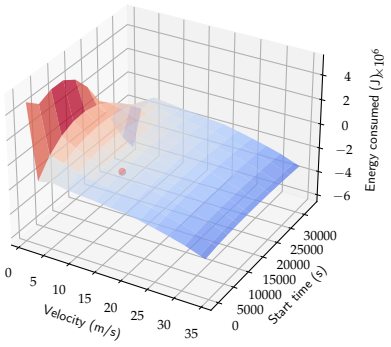
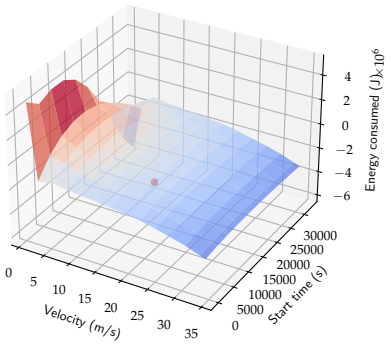


Figure 53: Energy graphs for segment 45 to 50.

(a) Energy graph for segment 51.

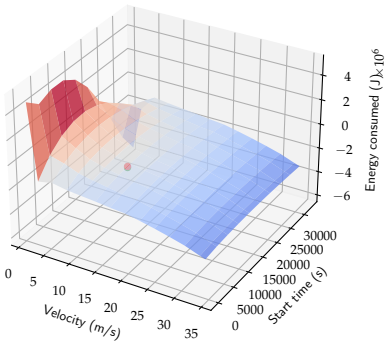
(b) Energy graph for segment 52.

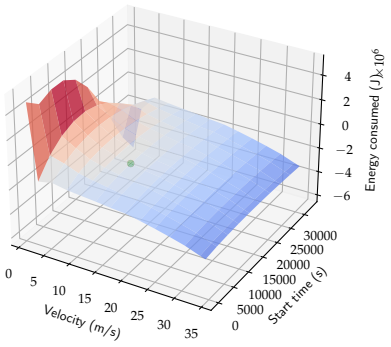




(c) Energy graph for segment 53.

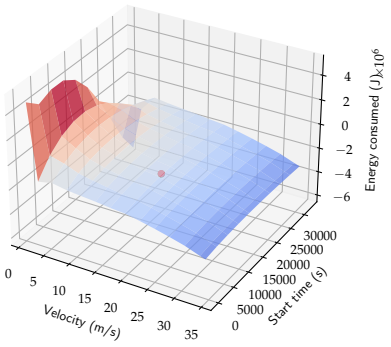
(d) Energy graph for segment 54.





(e) Energy graph for segment 55.

(f) Energy graph for segment 56.



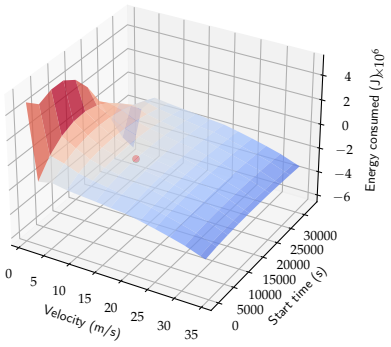
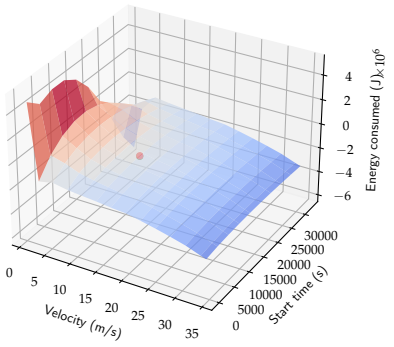
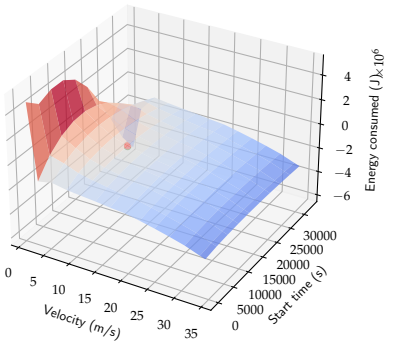


Figure 54: Energy graphs for segment 51 to 56.

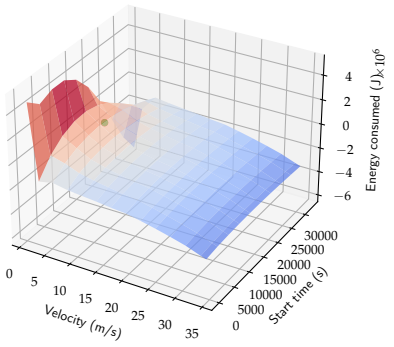
(a) Energy graph for segment 57.



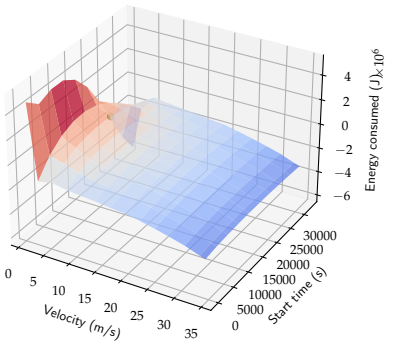
(b) Energy graph for segment 58.



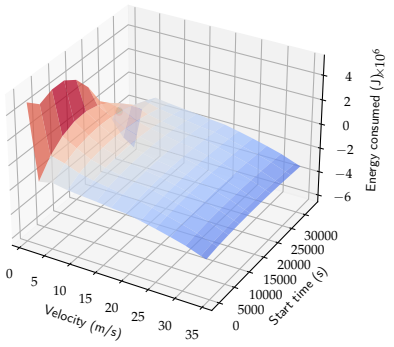
(c) Energy graph for segment 59.



(d) Energy graph for segment 60.



(e) Energy graph for segment 61.



(f) Energy graph for segment 62.

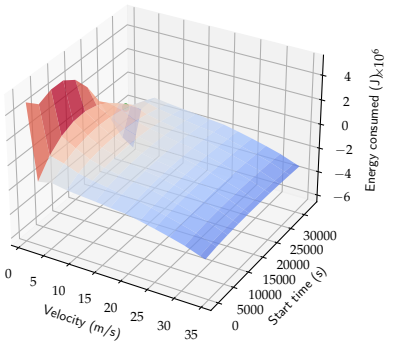
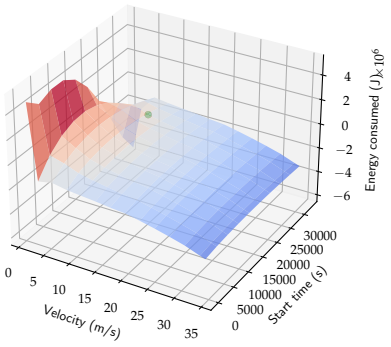
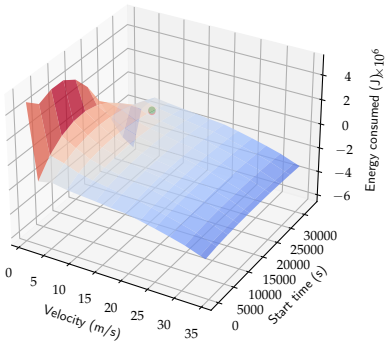


Figure 55: Energy graphs for segment 57 to 62.

(a) Energy graph for segment 63.

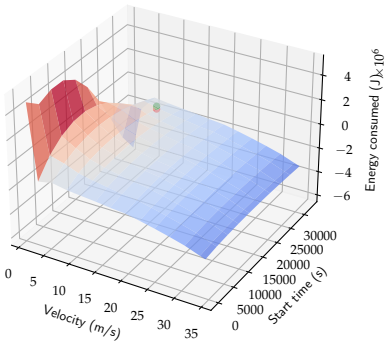
(b) Energy graph for segment 64.

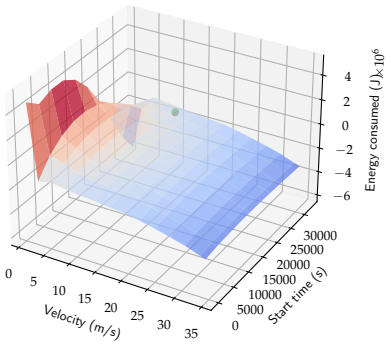




(c) Energy graph for segment 65.

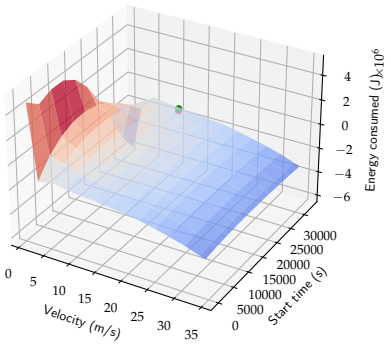
(d) Energy graph for segment 66.





(e) Energy graph for segment 67.

(f) Energy graph for segment 68.



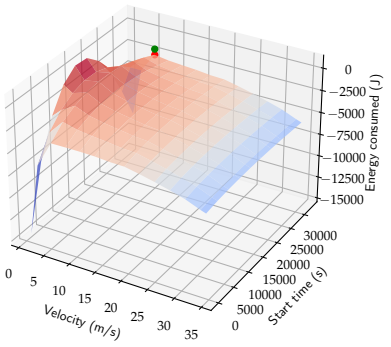
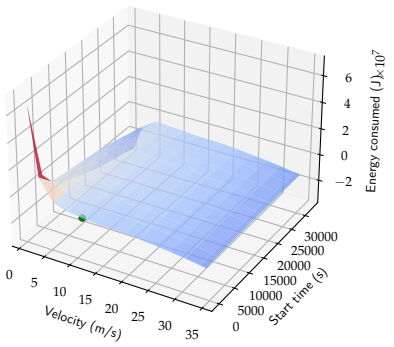
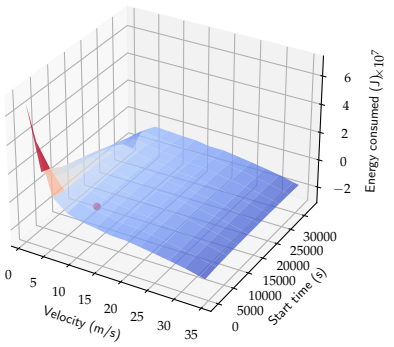


Figure 56: Energy graphs for segment 63 to 68.

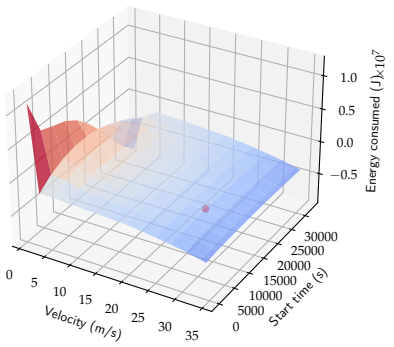
(a) Energy graph for segment 0.



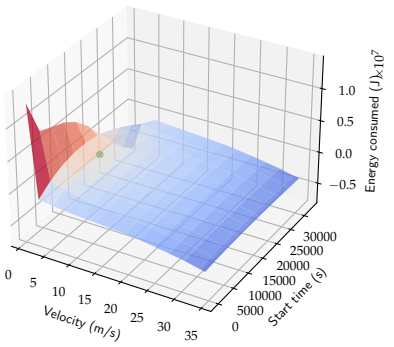
(b) Energy graph for segment 1.



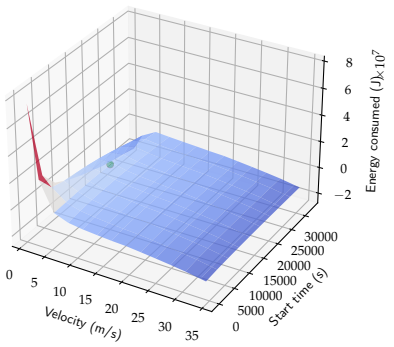
(c) Energy graph for segment 2.



(d) Energy graph for segment 3.



(e) Energy graph for segment 4.



(f) Energy graph for segment 5.

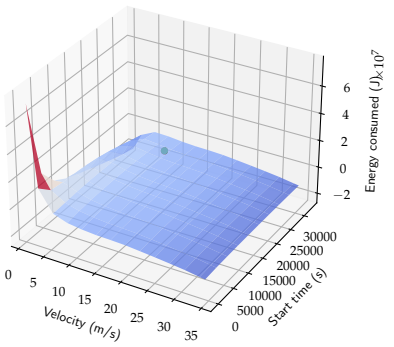
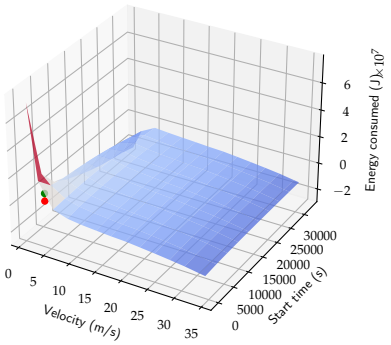
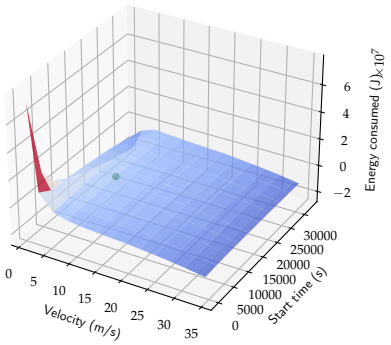


Figure 57: Energy graphs for segment 0 to 5 on day 1.

(a) Energy graph for segment 104.

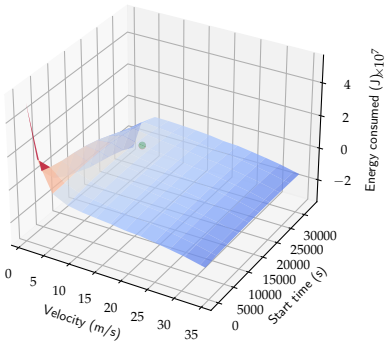
(b) Energy graph for segment 105.





(c) Energy graph for segment 106.

(d) Energy graph for segment 107.



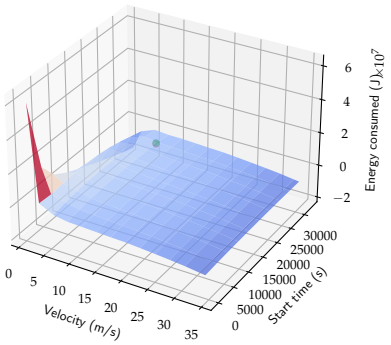
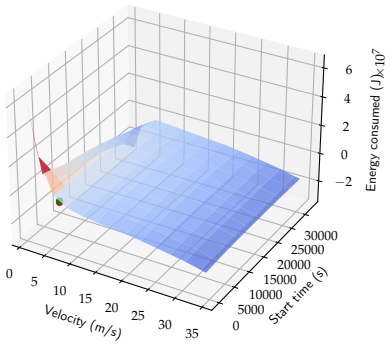
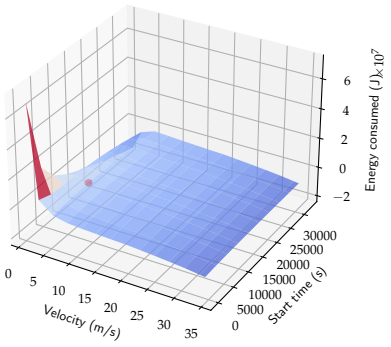


Figure 58: Energy graphs for segment 104 to 107 on day 2.

(a) Energy graph for segment 158.

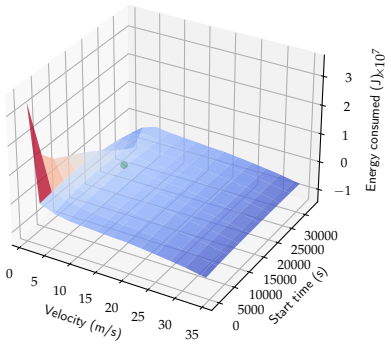
(b) Energy graph for segment 159.





(c) Energy graph for segment 160.

(d) Energy graph for segment 161.



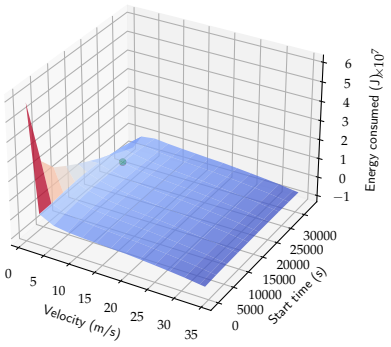
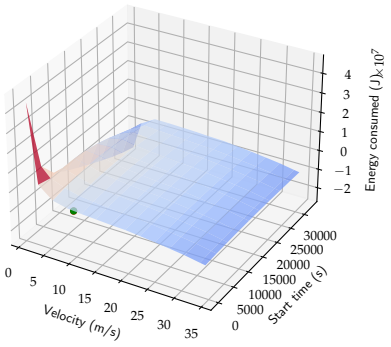
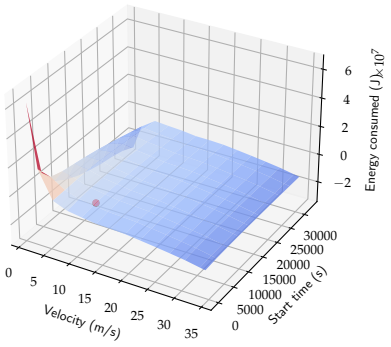


Figure 59: Energy graphs for segment 158 to 161 on day 3.

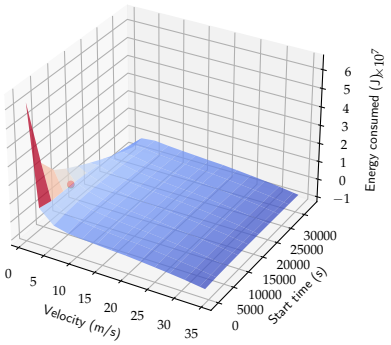
(a) Energy graph for segment 217.



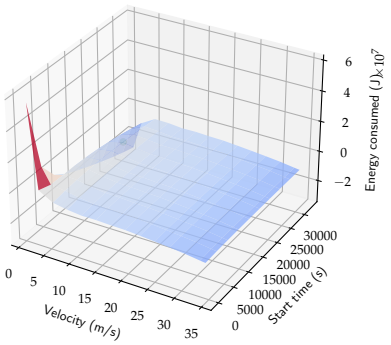
(b) Energy graph for segment 218.



(c) Energy graph for segment 219.



(d) Energy graph for segment 220.



(e) Energy graph for segment 221.

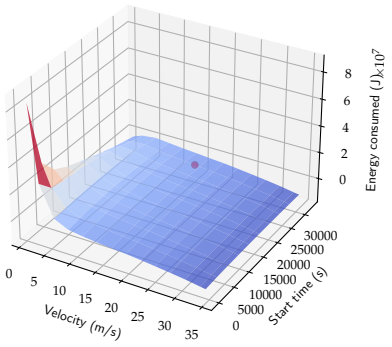
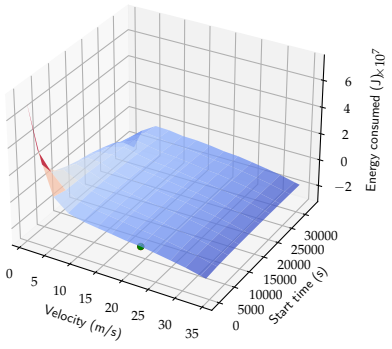
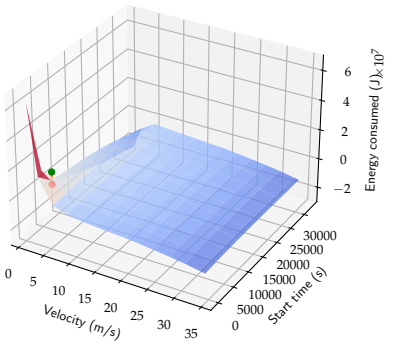


Figure 60: Energy graphs for segment 217 to 221 on day 4.

(a) Energy graph for segment 222.

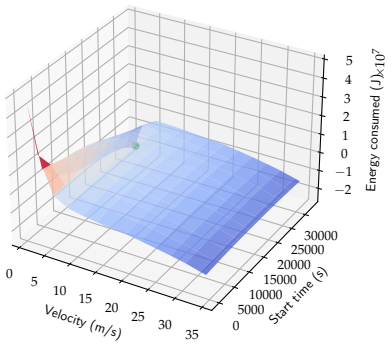
(b) Energy graph for segment 223.





(c) Energy graph for segment 224.

(d) Energy graph for segment 225.



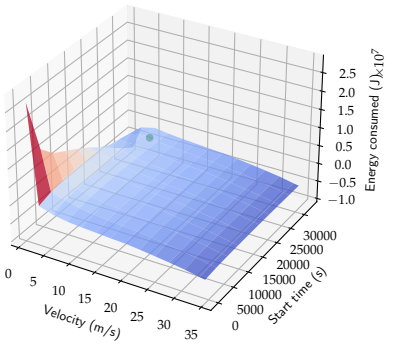
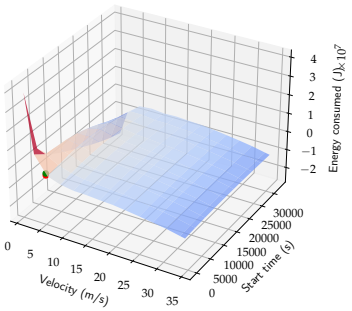
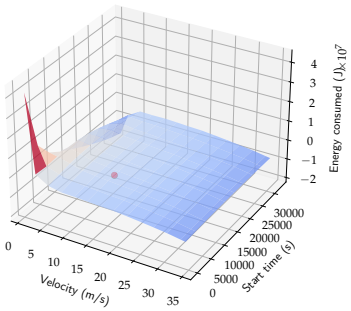


Figure 61: Energy graphs for segment 222 to 225 on day 5.

(a) Energy graph for segment 230.

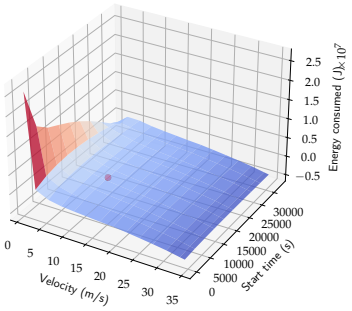
(b) Energy graph for segment 231.

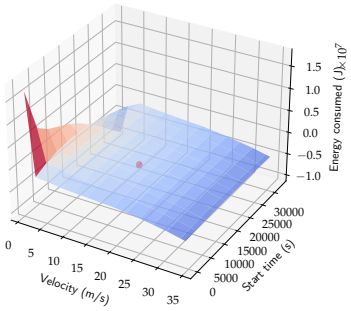




(c) Energy graph for segment 232.

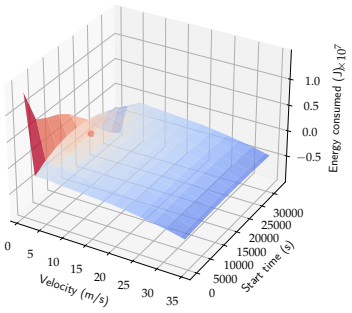
(d) Energy graph for segment 233.

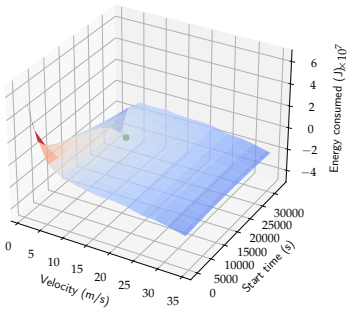




(e) Energy graph for segment 234.

(f) Energy graph for segment 235.





(g) Energy graph for segment 236.

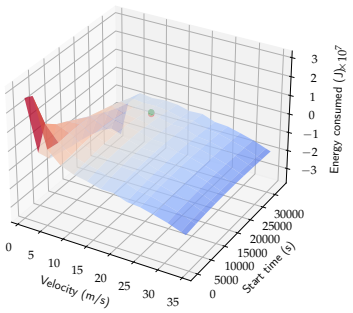
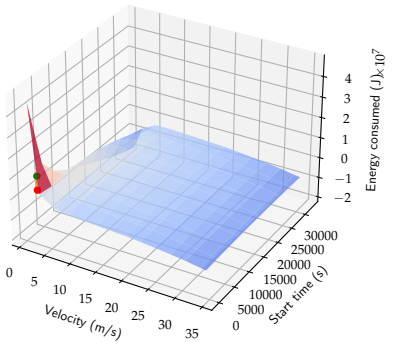
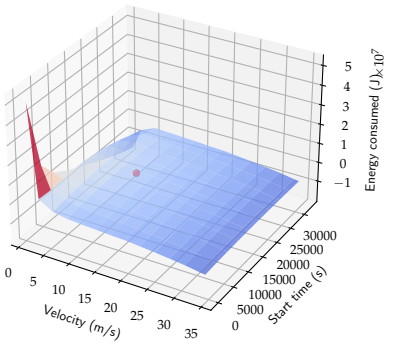


Figure 62: Energy graphs for segment 230 to 236 on day 6.

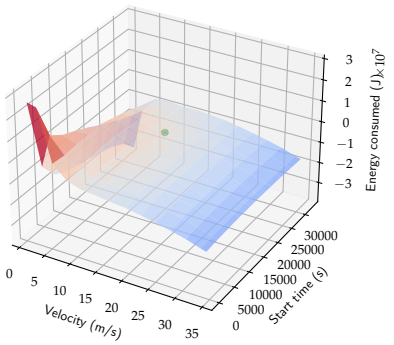
(a) Energy graph for segment 345.



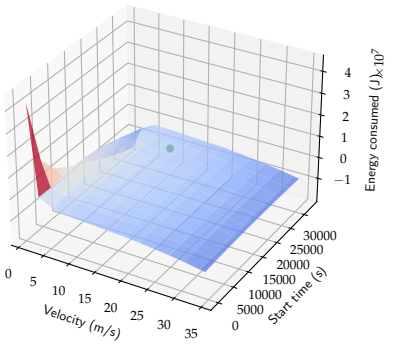
(b) Energy graph for segment 346.



(c) Energy graph for segment 347.



(d) Energy graph for segment 348.



(e) Energy graph for segment 349.

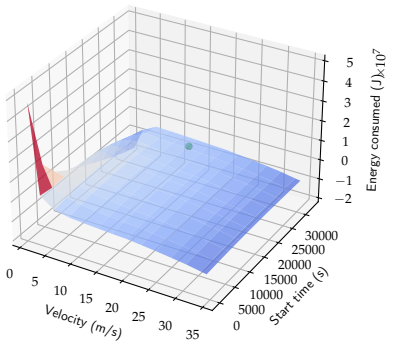
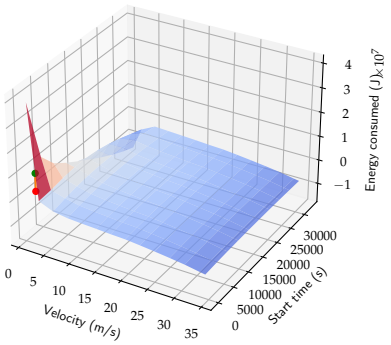
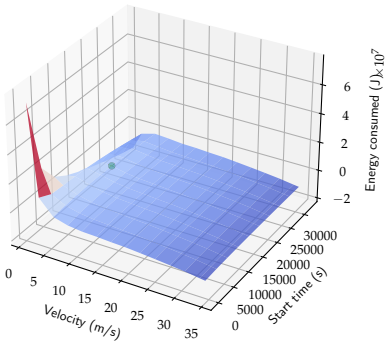


Figure 63: Energy graphs for segment 345 to 349 on day 7.

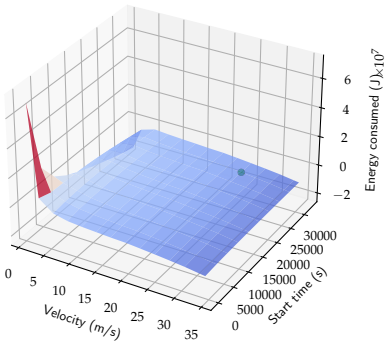
(a) Energy graph for segment 350.



(b) Energy graph for segment 351.



(c) Energy graph for segment 352.



(d) Energy graph for segment 353.

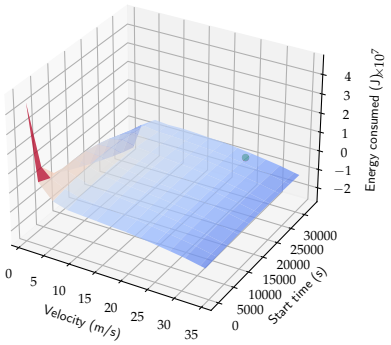
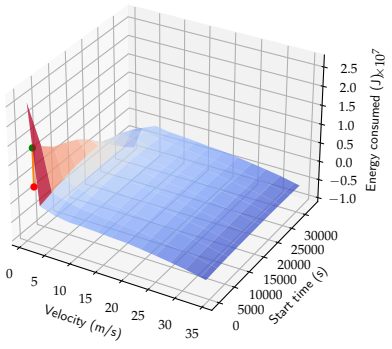
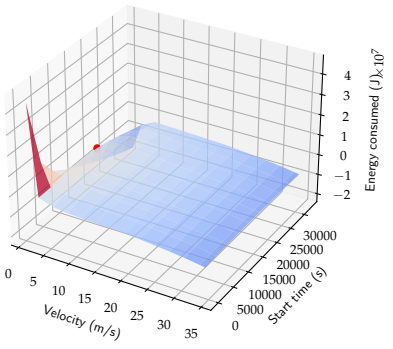


Figure 64: Energy graphs for segment 350 to 353 on day 8.

(a) Energy graph for segment 392.

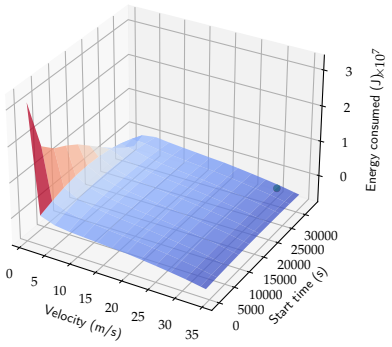
(b) Energy graph for segment 393.





(c) Energy graph for segment 394.

(d) Energy graph for segment 395.



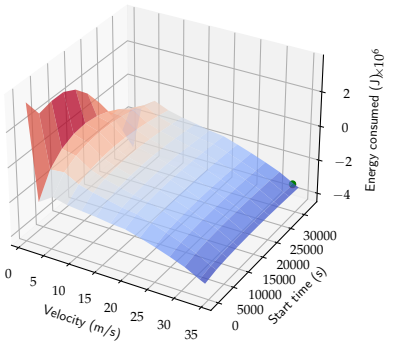


Figure 65: Energy graphs for segment 392 to 395 on day 9.