

UTRECHT UNIVERSITY



Utrecht
University

FACULTY OF SCIENCE

Routing electric cargo bikes: a hybrid solution approach

COMPUTING SCIENCE MSc THESIS

Author:
S.C. HESSELMANS

Supervisor:
Dr. ir. J.M. van den AKKER

Second supervisor:
Dr. J.A. HOOGEVEEN

July 2022

Abstract

Cities are getting more and more congested and environmental impact is becoming a larger point of interest for both customers and companies. Because of this more delivery companies are choosing to use cargo bikes instead of vans for the last-mile delivery of packages. This brings an interesting challenge as most vehicle routing problem solutions are created with vans in mind.

We study the vehicle routing problem for electric cargo bikes. Cargo bicycles differ from vans since they have less power and lower weight. Because of this, the difference in travel speed caused by the weight of the load and also by the slope of the road is significant for cargo bicycles, whereas this is negligible for vans. Hence, load-dependent travel times have to be taken into account when we want to find a good routing solution. Specifically, we consider the vehicle routing problem with load-dependent travel times for cargo bicycles (or VRPLTT) as proposed in an earlier paper by Fontaine. This problem is a variant of the capacitated vehicle routing problem with time windows using a single depot.

Our contribution is threefold. Firstly, we present a hybrid solution method based on combining column generation and local search. Such hybrid methods have previously been applied successfully on VRP models. We show that we can improve the solutions by applying this method. Secondly, we show how to introduce a constant wind in the VRPLTT model and present computational results showing different routing decisions are made when incorporating this wind. Lastly, in the original model, travel times are deterministic. This might not be a correct representation of the real world as travel times are not deterministic, especially in the city. Therefore, we show how to include stochastic travel times in our solution algorithm for VRPLTT and present computational results based on real-life instances.

Contents

1	Introduction	4
2	Literature	4
2.1	Hybrid approach to the VRP	5
2.2	Load dependency	6
3	Problem description	7
3.1	The vehicle routing problem with load-dependent travel times	7
3.1.1	Model representation	7
3.1.2	Calculating travel times	8
3.1.3	MIP formulation	9
3.2	The vehicle routing problem with load-dependent travel time under constant wind	10
4	Simulated annealing with ILP optimization	11
4.1	Simulated annealing	11
4.1.1	Neighborhoods	12
4.1.2	Initial solution	13
4.1.3	Solution acceptance	13
4.1.4	Selecting routes for the second stage	14
4.2	ILP optimization	14
5	Experiments	15
5.1	Data	15
5.2	Experimental setup	15
5.2.1	Parameter configuration	15
5.2.2	Experiments	16
5.3	Results VRPLTT	16
5.4	Results wind	18
6	Conclusion	20
6.1	Future research	20
A	Selecting the right neighborhoods	24
A.1	Configuration results	26
A.2	The final configuration	29
B	Stochastic travel times	30
B.1	Literature	30
B.2	Changes to the model	32
B.3	Gamma distribution	32
B.3.1	Calculating arrival times without waiting	32
B.3.2	Calculating arrival times with waiting	34
B.4	Normal approximation	39
B.4.1	Calculating arrival times without waiting	39
B.4.2	Calculating arrival times with waiting	41
B.5	Experiments	44
B.5.1	Distribution creation	44
B.5.2	Methodology	44
B.6	Results	45
B.6.1	Without waiting	45
B.6.2	With waiting	47

B.7 Conclusion	48
B.7.1 Future research	48
C Disallowing waiting	49
C.1 Experiments	49
C.1.1 Results	50
D Comparison on VRPTW instances	50
E Tables	51
F Additional wind results	57

1 Introduction

Cities are getting more and more congested and environmental impact is becoming a larger point of interest for both customers and companies. Because of this more delivery companies are choosing to use cargo bikes instead of vans for the last-mile delivery of packages. This brings an interesting challenge as most vehicle routing problem solutions are created with vans in mind.

The vehicle routing problem generally consists of one or more depots from which the vehicles leave, a set of vehicles to deliver the packages, and a set of customers with a certain demand. The goal is to assign customers to vehicles and compute a route for the vehicles such that each customer is visited while optimizing some objective function. One example of such an objective function is minimizing the distance traveled. The VRP is an NP-hard problem as it is a generalization of the TSP problem. There are many different variants of the VRP, such as time windows in which the customer's demand has to be fulfilled. Some of these variants will be discussed in Section 2

This thesis will focus on the vehicle routing problem with electric cargo bikes. Specifically, it will focus on the vehicle routing problem with load-dependent travel times for cargo bicycles (or VRPLTT) as proposed by Fontaine[33], which is a variant of the capacitated vehicle routing problem with time windows using a single depot. This paper tackles one of the ways bicycles differ from vans from a routing perspective, namely, load dependency. Cargo bikes have a lot less power and weigh a lot less compared to vans. Because of this, the difference in travel speed caused by the weight of the load and also by the slope of the road is significant for cargo bicycles, whereas this is negligible for vans. Taking this into account can significantly improve the routing solution when using these cargo bicycles.

In this thesis, we will try to improve and extend the results of Fontaine[33] in multiple ways. Firstly, we present a hybrid solution method based on combining column generation and local search. This solution method is further explained in Section 4. We hope that we can improve the found solutions by applying this method. Secondly, we extend the model proposed by Fontaine[33] by incorporating the effects of a constant wind acting upon the model, which influences the routing decisions made by the algorithm. Lastly, in the model proposed by Fontaine[33], travel times are deterministic. This might not be a correct representation of the real world as, especially in the city, travel times are not deterministic. Therefore, the second goal of this thesis is to implement stochastic travel times for the VRPLTT. This extension will be touched upon in the appendices.

We will not consider the state of charge in this model because it became clear that this is not a problem in the real world after talking with a cargo bicycle courier. They do run out of battery during a route, but they often combat this by bringing multiple charged batteries on a route, as it is easy to swap these.

This thesis first discusses the literature on the vehicle routing problem and load-dependent variants. This is followed by a formal problem description of the VRPLTT and the extension of constant wind, after which we introduce the proposed algorithm in detail. Next, we describe the experiments and their results. In the appendices, we first analyze multiple configurations of the proposed algorithm and discuss the choice of our final configuration. Then we introduce the stochastic extension to the VRPLTT and discuss the previous literature on stochasticity in the VRP. Afterward, we show how to include stochastic delay in the VRPLTT model. Lastly, we consider the VRPLTT where waiting for the start of a time window is not allowed.

2 Literature

Many papers have been written on the Vehicle Routing Problem (VRP) and its variants. The VRP started out as the Truck Dispatching problem introduced by Dantzig and Ramser [1], in which gasoline trucks needed to transport gasoline from bulk storage to service stations. A procedure based on a relaxed integer linear programming formulation is used to find assignments to trucks with minimum total travel distance. Clarke and Wright[2] later reformulated this to incorporate trucks with varying capacity to serve customers, while also improving the solution method of Dantzig and Rasmer by up to 17%.

The VRP is NP-Hard and therefore, most exact solution methods are only used on small instances or use heuristics to find good solutions. Laporte[11] reviews the main results for the classical VRP with only vehicle capacity constraints. He compares the results of three main categories of algorithms, exact algorithms, classical heuristics, and metaheuristics. He concluded that the best exact algorithms were based on branch and cut, but were inconsistent in terms of performance. For larger instances, metaheuristics are preferred as these typically yield results within 1% of the best-known solution value. Most of these metaheuristics use local search, genetic search, or a combination of both. Some solution methods adopt a hybrid approach, which we will discuss in the next section.

2.1 Hybrid approach to the VRP

Hybrid solution methods have been tried multiple times for the VRP and similar models. Most combine a form of a (meta)heuristic and a set partitioning or set covering model which optimizes the results found by the heuristic. One of the first to do so were Rochat and Taillard[6] in 1995. They used a hybrid solution method consisting of a Tabu search with probabilistic diversification in the form of random initial solutions and randomness in the algorithm. They also used an intensification technique. This intensification technique focuses more intently on regions that were previously found to be good. They combined this tabu search with a set partitioning (SP) formulation to optimize the solutions found by the tabu search. Using this technique, they improved the best-known solution for VRP and VRP with time windows for multiple different instances.

Later in 2005 Alvarenga et al.[9] proposed a hybrid solution method combining a genetic algorithm (GA) and a set partitioning formulation. They focused on producing a fast genetic algorithm as a route generator to use the set partitioning formulation more effectively. They approach this in multiple phases. First, they use the GA to generate routes that are optimized using the SP formulation. They generate a reduced problem from the solution of this SP model by inserting the customers of each route with a probability of 50%. This reduced problem is again solved using the GA. These two steps are repeated until a time limit has been reached, after which the results are optimized again using the SP formulation. Using this technique they improved or matched many best-known solutions for the VRPTW.

In 2013 Subramanian et al.[18] proposed a hybrid solution method for a class of vehicle routing problems. Their proposed method consists of a sequence of SP models with columns generated by a metaheuristic, which is solved using a MIP solver. The meta-heuristic used is the ILS-RVND, which uses Variable Neighborhood Descent with Random neighborhood ordering (RVND). If the number of customers is relatively small the SP model was only executed once after the metaheuristic finished. For larger instances the SP model is executed after each iteration of the metaheuristic that improved the local optimum, updating the ILS-RVND parameters based on its results. This solution method was tested on many different variants of the VRP. Even though this algorithm was developed to apply to many different problems the authors concluded that it was quite competitive with heuristics that were specifically constructed for the tested variants.

Similarly, Villegas et al.[20] propose a hybrid solution for the truck and trailer routing problem. In this problem, we have a set of customers that either need to be served by a truck, or a truck pulling a trailer. The hybrid solution method is based on a greedy randomized adaptive search procedure (GRASP) with ILS which local optima are used as columns in an SP formulation. They found that the two-phase approach clearly improved the results of the metaheuristic.

Montoya et al.[26] propose a hybrid solution for the electric vehicle routing problem with a nonlinear charging function. In this hybrid solution, a variable neighborhood descent is used, where the search is further diversified by concatenating the routes to a TSP tour and perturbing that. Afterward, the found routes are used as columns in an SP problem and solved to obtain a solution. They found that their algorithm worked well, getting close to or matching the results of the MILP formulation.

in 2019 Hiermann et al. [31] also proposed a hybrid solution method for the vehicle routing and fleet mix problem with multiple vehicle classes. They designed a metaheuristic based on a hybrid genetic algorithm, which combines a genetic algorithm with local search optimization. The routes found by this metaheuristic are optimized by solving an SP problem. This is done multiple times during the execution

using the results of the SP problem as input for the local search. Their solution produced average solutions of equal or higher quality than existing algorithms.

In still unpublished work, Villegas et al. [30] propose a hybrid solution for the technician routing and scheduling problem with conventional and electric vehicles, where a set of technicians have to be routed to customers using a non-homogeneous fleet of vehicles. They use a parallel form of GRASP to generate columns as input for a set covering problem. They compared their algorithm against the state of the art of similar problems and found that it performed competitively.

A similar problem to the VRP is the dial-a-ride problem, in which customers have a pickup and drop-off location and need to be served within a respective time window by a vehicle. Parragh and Schmid[17] propose a hybrid solution method combining a large neighborhood search (LNS) and column generation iteratively solving a set covering problem. They concluded that the hybrid solution effectively improved on only the LNS and found several new best-known solutions.

2.2 Load dependency

Some variants of the VRP take the load of the vehicle into account. Most of these variants incorporate this load dependency in the objective function, i.e. traveling is more expensive when the vehicle is more heavily loaded. Some also include this load into the constraints of the model, i.e. traveling along an edge takes longer when more heavily loaded.

To tackle this Bektas et al. [14] introduce the pollution routing problem or PRP. In this model, the goal is to minimize pollution or energy consumption instead of the distance traveled. They compare different objective functions: Distance, weighted distance(load multiplied by distance), energy consumption, or costs. For the latter objective function, they define a cost for pollution in terms of the amount of CO₂ produced, such that this is also taken into account. The problems were modeled as an ILP and solved using CPLEX. The results showed that when optimizing for costs the wages for the drivers were more significant than the fuel costs. This caused the model to prefer more polluting routes. It also showed that minimizing the energy and weighted distance did not always result in the same outcome. Minimizing the weighted distance might even cause an increase in energy consumption compared to distance minimization.

In 2015 Zachariadis et al.[23] consider the load-dependent vehicle routing problem (LDVRP) with the pickup and delivery extension. This model uses the weighted distance, often expressed as ton-kilometer, and was compared to the PRP energy consumption objective which tries to minimize the energy consumption also taking speed into account. For cases of low speed and high tare weight(weight of an empty vehicle), the PRP energy consumption objective and LDVRP are fairly similar according to the authors. They also introduce the simultaneous pickup and delivery extension for the LDVRP. For pickup and delivery problems, the vehicles do not only deliver goods to the customers but also pick up goods. They try to find the best solution using a local search approach, which allows infeasible solutions during the search process with a penalty term incorporated to disincentivize picking an infeasible solution. They concluded that their model was best suited for heavy-duty transportation logistics activities. Furthermore, they also concluded that their solution method performed well on the LDVRP.

A closely related problem to the VRP is the Chinese Postman Problem or CPP. In the CPP the goal is to visit every edge of a graph instead of every vertex in VRP. Corberán et al. [28] introduce a new variant of the CPP called the Chinese Postman Problem with load-dependent costs (CPP-LC). In the new variant, the costs of traversing an edge are made load-dependent instead of constant as it is in the original CPP. For example, imagine an edge is a street where every house has to be visited and has a certain demand. This demand is delivered to every house and after every house, the load of the vehicle is decreased. The authors assume that the demand is uniformly distributed along an edge and calculate the cost of traversing an edge accordingly. This load dependency is taken into account in the objective function of the model. The main difference between the CPP-LC and the LDVRP is that it is an arc-routing problem instead of the node-routing problem of LDVRP. They defined two metaheuristics to solve this model: An iterated local search (ILS) and a variable neighborhood search (VNS). Both these metaheuristics obtained good results within small computing times in the experiments conducted

for this paper.

All previously discussed models only used load dependency in the objective function. But, Fontaine[33] introduced the VRP with load-dependent travel times for cargo bicycles (VRPLTT). In this model, the load of the cargo bicycle and the gradient of the road affect the travel times on the edges. He used these load-dependent travel times to calculate more accurate routing for the bicycles. He solved this model using a MIP formulation and an Adaptive Large Neighborhood Search(ALNS). This new model was able to save significant amounts of travel time compared to traditional VRP solutions when considering the impacts of the load of the bicycle and the gradient of the road. To calculate the load-dependent travel times in the VRPLTT model, we need the gradient of the road. However, this gradient is most often not constant between two customers. Thus, to calculate the load-dependent travel times, a new graph is created where each edge has a constant gradient, essentially splitting the existing edges into multiple new ones. This may, however, cause different routes to be taken between two customers for different load levels. Therefore, for each load level, an all-pair shortest path algorithm is run to create the travel time matrix. The ALNS was also tested on benchmark instances for the VRPTW. The author concluded that the ALNS performed well on both the VRPTW and the VRPLTT models.

3 Problem description

In this section, the formal description of the vehicle routing problem with load-dependent travel times and the extension regarding constant wind will be discussed.

3.1 The vehicle routing problem with load-dependent travel times

First, we will discuss the problem as defined by Fontaine [33]. In this model, we have a single depot, several customers that need to be visited within their respective time windows by exactly one bicycle, and a homogeneous fleet of bicycles. The bicycles need to leave from the depot, visit the customers in its route and then return to the depot while minimizing the total time traveled. Now, as stated before the weight of the bicycle and the gradient of the route influence the velocity at which a cargo bicycle can travel. Therefore, if a customer lives on a hill for example we might want to avoid visiting that customer until later in the route, as the bicycle is the lightest at end of a route. The objective function for the vehicle routing problem with load-dependent travel times (VRPLTT) consists of only the load-dependent travel times. Labor costs are often the most crucial factor determining the costs of a delivery plan, especially when using cheaper vehicles like cargo bicycles. Because cargo bicycles are comparatively cheap it is realistic to limit the cost function to the travel time. By taking the load of the vehicle into account we can minimize the actual travel times with more precision as shown by Fontaine[33].

In this model, we plan around electric cargo bicycles. By European law, these bicycles can only deliver a maximum power of $250W$ and may only support the cyclist up to $25km/h$. If the bicycle would exceed any of these limits it would require a driver's license to operate it. Because of this limited assistance, careful planning helps improve the solution. In the following subsections, we will discuss the model in detail.

3.1.1 Model representation

Fontaine[33] defined the VRPLTT on a graph $G = (N, E)$ where we have the following parameters,

- $N = \{0, 1, \dots, n\}$ is the set of customer nodes and depot
- E the set of arcs (i, j) where both $i \in N$ and $j \in N$
- Customer 0, the depot node where all trips should start and end
- $N_0 = \{1, \dots, n\}$ is the set of customer nodes

- q_i , the demand mass of customer $i \in N_0$
- s_i , the service time of customer $i \in N_0$
- $[a_i, b_i]$ the timewindow of customer $i \in N_0$
- d_{ij} , the distance of the edge between customer i and j
- Q the capacity of the vehicles of the homogeneous fleet
- L the set of load levels
- t_{ij}^l travel time between i and j with loadlevel l

And the variables

- x_{ij} binary variable indicating whether edge (i, j) is driven
- f_{ij} is the load of the vehicle on arc (i, j)
- z_{ij}^l is a binary variable indicating whether loadlevel l is used on edge (i, j)
- y_i is the arrival time at customer i

3.1.2 Calculating travel times

Before we can truly define the model we need to define the travel time calculation. In contrast to many other load-dependent models, these are dependent on the weight of the vehicle and therefore the load. The travel times between two points are calculated using well-known physical equations. Namely,

- Air resistance, which is defined as

$$F_d = \frac{\rho C_d A}{2} v^2 \quad (1)$$

where ρ is the air density, C_d is the coefficient of drag and A is the frontal area of the cargo bike. For a typical cargo bike with a cargo hold with a frontal area of 0.2 m^2 the $C_d = 1.18$ and $A = 0.83 \text{ m}^2$ according to Bombach[24]. The standard air density of $\rho = 1.1 \text{ kg/m}^3$ at sea level with a temperature of $25 \text{ }^\circ\text{C}$ was used by Fontaine[33].

- Rolling resistance, which is defined as,

$$F_R = C_r \cdot m \cdot g \cdot \cos(\arctan(h)) \quad (2)$$

Where C_r is the coefficient of rolling resistance, m the mass of the vehicle, g the gravitational constant and h the gradient of the road. For the value of C_r Fontaine[33] used $C_r = 0.01$ as detailed by Wilson & Papadopoulos[32] and $g = 9.81 \text{ m/s}^2$.

- The gravitational force, which acts on the vehicle when climbing up a slope is defined as,

$$F_G = m \cdot g \cdot \sin(\arctan(h)) \quad (3)$$

- Finally, the model defined by Fontaine[33] incorporates the frictional forces F_F of the mechanical parts of the cargo bicycle. These are considered to be 5% of the total force applied to the system.

Using all these defined formulas the total power required to travel v meters per second on a road with gradient h using a bike of mass m is equal to

$$P = (F_D + F_R + F_G + F_F)v \quad (4)$$

$$= (F_D + F_R + F_G)v + 0.05P \quad (5)$$

$$P = \frac{(F_D + F_R + F_G)v}{0.95} \quad (6)$$

To calculate the velocity of the bicycle using this equation we can solve for v with power input P , gradient h , and mass m . We assume that the cyclist and bike both deliver constant power. However, solving directly for v is not trivial. Therefore, we employ a technique based on code from [33] provided by Fontaine, which solves for v by starting at the maximum speed and gradually reducing v until the required power to travel that speed is less than or equal to the input power. Using this technique, we can calculate the velocity when traveling from customer i to customer j with load level l . With this velocity and the distance d_{ij} we can directly calculate the travel time from customer i to customer j with load level l , t_{ij}^l .

To reduce the complexity of the model the mass range is split into load levels. Each load level l has a certain upper r^l and lower bound p^l . Vehicles within the same load level are binned and have the same travel time, which is calculated with $m = \frac{p^l + r^l}{2}$

3.1.3 MIP formulation

The objective function of the VRPLTT model is

$$\min \sum_{(i,j) \in E} \sum_{l \in L} t_{ij}^l z_{ij}^l \quad (7)$$

This objective function minimizes the total travel time over the solution taking the load-dependent travel times into account. This objective function is subject to the following constraints.

$$\sum_{j \in N} x_{ij} = 1 \quad \forall i \in N_0 \quad (8)$$

$$\sum_{i \in N} x_{ij} = 1 \quad \forall j \in N_0 \quad (9)$$

$$\sum_{j \in N} f_{ji} - \sum_{j \in N} f_{ij} = q_i \quad \forall i \in N_0 \quad (10)$$

$$q_j x_{ij} \leq f_{ij} \leq (Q - q_i) x_{ij} \quad \forall (i, j) \in E \quad (11)$$

$$\sum_{l \in L} z_{ij}^l = x_{ij} \quad \forall (i, j) \in E \quad (12)$$

$$\sum_{l \in L} p^l z_{ij}^l \leq f_{ij} \leq \sum_{l \in L} r^l z_{ij}^l \quad \forall (i, j) \in E \quad (13)$$

$$y_i - y_j + s_i + \sum_{l \in L} t_{ij}^l z_{ij}^l \leq M_{ij}(1 - x_{ij}) \quad \forall i \in N, j \in N_0, i \neq j \quad (14)$$

$$a_i \leq y_i \leq b_i \quad \forall i \in N_0 \quad (15)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in E \quad (16)$$

$$f_{ij} \geq 0 \quad \forall (i, j) \in E \quad (17)$$

$$z_{ij}^l \in \{0, 1\} \quad \forall (i, j) \in E, l \in L \quad (18)$$

Constraints 8 and 9 ensure that every customer is visited once by stating that every customer should have exactly one incoming and one outgoing edge. Constraint 10 ensures that the demand of each customer is met and that the load of the vehicle is decreased by this demand after each customer visit, conserving the flow. Constraint 11 ensures that the vehicle capacity is not violated. Constraint 12 makes sure that

when an edge is selected only one load level travel time is selected and otherwise none. The selection of the correct load level is ensured in constraint 13 by stating that the current weight of the vehicle should be within the bounds of the selected load level. Constraint 14 updates the visiting times between the customers in the routes. The difference between these arrival times should be larger than or equal to the sum of the load-dependent travel time and the service time of the origin node if the edge is taken. These constraints were derived from a set of non-linear constraints by Fontaine[33] in a similar way to Bektas and Laporte [14] and Cordeau et al.[10] These papers further give a lower bound to these constraints, which was adapted by Fontaine[33] to $M_{ij} = \max\{0, b_i + s_i + t_{ij}^1 - a_j\}$. Since multiple travel times can be selected, the slowest travel time t_{ij}^1 was used for this lower bound. Constraint 15 enforces the arrival at each customer within its time window. Lastly, constraints 16, 17 and 18 ensure valid values for the decision variables.

This formulation allows early arrival, which is not penalized. Only the time traveled is counted towards the objective function. This early arrival is allowed by equation 14 as the difference in visiting times between the origin customer and the destination customer has to be at least the sum of travel time and service time, but it is allowed to be larger than that. Thus waiting is allowed without penalty.

3.2 The vehicle routing problem with load-dependent travel time under constant wind

In this subsection, we will discuss how a constant wind can be incorporated into the VRPLTT model. Now as a route always returns to the depot, and therefore essentially is a circle, one could be forgiven to think that this would have no impact on the solution. However, as we have limited input power, a maximum travel speed, the load of the vehicle, and slopes to take into account it does make a difference.

The wind was incorporated in a 2D setting. That is, the calculations involving the wind are done using 2D vectors. As the amount of air resistance grows quadratic in the wind speed this is not as straightforward as projecting the total wind on the system in the direction of the cyclist. To take the impact on the air resistance of wind into account some changes have to be made in the formal definition of the model. To incorporate the wind, we need to adjust the calculation of air resistance in equation (1) as follows.

Let \vec{v}_c be the wind that the cyclist would experience in wind still circumstances. This vector points in the opposite direction to the traveling direction of the cyclist and has length v , where v is the speed of the cyclist. Let \vec{v}_w be the wind vector. For the total experienced wind, we now have

$$\vec{v}_t = \vec{v}_c + \vec{v}_w.$$

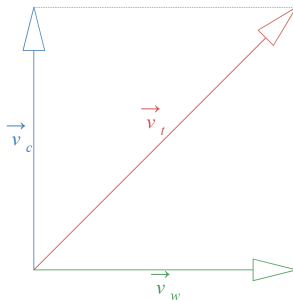


Figure 1: Example of calculating \vec{v}_t

Let \vec{e}_c and \vec{e}_t denote the vectors with length 1 in the directions of \vec{v}_c and \vec{v}_t , respectively, so $\vec{v}_t = |\vec{v}_t| \vec{e}_t$. From equation (1) we know that the total air resistance for the cyclist is proportional to the square of the speed. However, we only need the component in direction \vec{e}_c as that is opposite to the travelling

direction of the cyclist. Therefore in equation (1) we should replace v^2 by:

$$|\vec{v}_t|^2 \vec{e}_t \cdot \vec{e}_c$$

For the air resistance experienced by the cyclist, we now obtain

$$F_d = \frac{\rho C_d A}{2} (|\vec{v}_t|^2 \vec{e}_t \cdot \vec{e}_c) \quad (19)$$

Suppose, for example, we have a cyclist cycling directly north with speed v under a tailwind with speed $2v$. We have $\vec{v}_t = \begin{pmatrix} 0 \\ v \end{pmatrix} + \begin{pmatrix} 0 \\ -2v \end{pmatrix} = \begin{pmatrix} 0 \\ -v \end{pmatrix}$, $\vec{e}_c = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, and $\vec{e}_t = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$. Thus we get that

$$\begin{aligned} F_d &= \frac{\rho C_d A}{2} (|\vec{v}_t|^2 \vec{e}_t \cdot \vec{e}_c) \\ &= \frac{\rho C_d A}{2} (v^2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ -1 \end{pmatrix}) \\ &= \frac{\rho C_d A}{2} (-v^2) \end{aligned}$$

As $\rho, C_d, A > 0$ this results in a negative air resistance thus helping the cyclist, as expected, in this example.

These calculations for incorporating the wind are done during the calculation of the travel speed explained in Section 3.1.2. As this equation for the air resistance ties into the calculated travel speed, no other changes to the model have to be made to include this constant wind.

4 Simulated annealing with ILP optimization

Solving the model directly using an ILP is very time-consuming and not practically feasible, especially on larger instances. Fontaine [33] used an ALNS to solve this problem. We implemented a similar solution method to the methods described in Section 2.1, combining a metaheuristic with some form of ILP optimization. These hybrid techniques have proven to be successful in outperforming state-of-the-art algorithms and the metaheuristics on which they are based.

The basic idea of the method is to combine some form of local search and an ILP. The local search method produces a lot of feasible solutions from which columns can be extracted, replacing the column generation. As local search can generally produce quite good results, the set of extracted columns contains a lot of potentially good trips. After the local search step, we can apply the ILP on the generated columns to possibly find an even better solution than the results of the local search. We implemented this idea using simulated annealing which is used to generate the columns. These columns are then used as input for a set cover problem creating a two-staged solution approach.

4.1 Simulated annealing

The local search is based on simulated annealing. Simulated annealing[3] (SA) is a local search based method that incorporates methods to escape local optima during the search, with as goal to find the global optimum more effectively. The SA algorithm not only allows improvements during the local search but can also accept worse solutions based on the so-called *Boltzmann* function. By updating the so-called temperature parameter the probability of accepting worse solutions is decreased as the search progresses. This essentially creates two phases. In the first phase, the algorithm is allowed to explore more of the search space, while in the second phase it becomes more like a traditional local search algorithm only accepting improvements. This SA technique has been successfully applied to many different problems such as the VRP and its variants.

Following a common technique to increase the performance of local search algorithms employed by for example Fontaine[33] and Hemmelmayr et al.[16], we allow infeasibility in the search space. Tardiness

is penalized with weight w_t . This weight is linearly increased as the temperature decreases to incentivize feasible solutions more as the search progresses.

Our SA algorithm incorporates a restart mechanism to escape local optima in which it arrives during the search. If the best solution has not changed for a certain amount of iterations and the temperature is below a certain threshold, the current solution is set to the best solution found so far and the temperature is increased. By increasing the temperature the best solution is shaken up to increase the search space. This strategy is applied multiple times to increase the chance of the restart finding an improvement, as preliminary testing showed that not every restart had an effect. This does, however, come at the cost of increased computing time for often small improvements.

To further increase the search space, multi starts are used. Multiple different instances are run separately from each other. Each having a slightly different initial solution, and, of course, making different changes to the solution, causes the multi starts to find different solutions. This does not only improve the solution found by the SA algorithm but also increases the effectiveness of the second stage of the solution method.

4.1.1 Neighborhoods

Different neighborhoods were considered for the SA-based first stage. 9 different neighborhoods were implemented and tested in different subsets. The results of these tests are described in Appendix A. In the end, the following configuration showed the best performance, while remaining simple.

1. Move random customer
Randomly choose two customers c_1 and c_2 . Insert c_1 directly before c_2 . This classical neighborhood is employed in many different simulated annealing implementations.
2. Swap two random customers
Randomly select two random customers c_1 and c_2 and swap these customers such that c_1 is positioned where c_2 was and c_2 where c_1 was. This repeated 4 times and the best swap is chosen as the neighborhood.
3. Reverse a random subroute
First, a random route is selected. For this route two non equal customers c_1 and c_2 are randomly selected, after which the sub route from c_1 to c_2 is reversed including the randomly selected customers. This neighborhood is sometimes also referred to as 2-opt in literature.
4. Scramble a random subroute
First, a random route is selected. In this route two customers are randomly selected, after which the sub route between the two customers is put into a random order, scrambling part of the route. This neighborhood allows larger changes to the solution at once, which can shake up the current solution more.
5. Move random customer to the best position in a random route
Randomly choose a customer c from one of the existing routes and randomly select a random route r . Then, customer c is inserted into r in the position with the smallest increase in objective value. This neighborhood is repeated 4 times and the best placement is chosen as the neighbor.
6. Move a random customer to its best possible position in any other route than the route this customer is currently placed in
Randomly select a customer and insert it in the best position in any of the other routes. By not considering the route in which the randomly selected customer is located the customer is always moved if the neighbor is accepted.

Each neighborhood is selected randomly with equal probability, except for neighborhood 6. This neighborhood has a 10 times smaller chance of being selected compared to the other neighborhoods. This choice was made for two reasons. Firstly, this neighborhood is very expensive in terms of computing

time compared to the other neighborhoods as it checks every position in all routes, except for the route the selected customer is located in. For larger instances, this could result in more than 20 times as many positions calculated as for some other neighborhoods. This cost is reduced somewhat by implementing caching to prevent recalculating the scores for the same position multiple times. By reducing the probability of this neighborhood being selected it is executed less often and therefore has a smaller impact on the performance. Secondly, as it is quite a greedy neighborhood it could cause the simulated annealing to get stuck in local optima. By adjusting the probability for this neighborhood this can be prevented. During preliminary testing, it showed that reducing the probability of this neighborhood to the chosen probability did not seem to reduce the overall performance of the algorithm while improving the execution time.

4.1.2 Initial solution

A good initial solution can influence the quality of the final solution and convergence rate[27]. Although in general, this effect can be quite small it might be beneficial to create an initial solution that is not completely random.

The initial solution for the proposed algorithm is generated using a simple greedy procedure based on the nearest neighbor heuristic. It works as follows:

1. First select the customer with the earliest upper time window. This is the seed customer for the first route and is removed from the list of customers. The customer with the earliest upper time window is selected to minimize the number of late arrivals in the initial solution
2. Select the customer who adds the least amount of costs to the route, taking into account penalties for being too late and the travel times. These travel times assume the vehicle to be under maximum load. The initial solution is created in a single pass per route, and therefore we don't know the actual weight in this part during the creation. Repeat this step until no customers fit in this route
3. Select a seed customer for the next route using the same technique as in step 1 and repeat step 2 and this step until all customers are inserted into a route.

This procedure creates an initial solution of reasonable quality. However, as the proposed algorithm employs multi starts it might be beneficial to create slightly different routes for each start. Therefore, a random number uniformly distributed between 0 – 5 is added to the costs of adding that customer if the customer could be served on time in step 2. This introduces some randomness to selecting the best customer to add to the route in step 2. This causes slight differences in each route.

In some cases with a limited amount of available vehicles, this nearest-neighbor procedure can't fit all customers in a single pass. Therefore, if this is the case the remaining customers are inserted greedily into the initial solution minimizing travel time and occurred penalties.

4.1.3 Solution acceptance

If a neighbor improves the value of the objective function of the current solution it is always accepted as the new solution. If the neighborhood provides a solution that does not improve the current solution it is accepted with probability $e^{\frac{x}{T}}$, Where x is the improvement in the objective function provided by the neighbor. As this function is only applied if the neighbor is worse than the current solution x is always negative and thus $e^{\frac{x}{T}}$ produces a probability. T is the so-called temperature parameter. As T decreases during execution, fewer solutions that do not improve the objective function will be selected, descending more greedily as the execution progresses.

4.1.4 Selecting routes for the second stage

The ILP stage of the SA-ILP algorithm requires a collection of routes. Giving the ILP more routes gives it more freedom to find a new subset which improves the solution. This does, however, come at the cost of requiring more computing time to find the best subset of the given routes. Therefore, passing all explored routes to the second stage might not be ideal as early in the execution the quality of the solution is very low and is changed often. This creates many sub-optimal routes that most likely will never be used in the optimal solution.

The first idea that might come to mind is just using the final solutions of the different multi starts. However, in preliminary testing, this proved to be sub-optimal as well, as very few unique routes were discovered per start, limiting the effectiveness of the ILP stage and therefore requiring many more starts to achieve improvements and thus more computing time and/or power. To improve the number of unique routes found by the different starts, routes are selected if the current temperature is below the threshold β . Furthermore, the best solutions are also always added to the collection of routes. This is a compromise between selecting all routes, which selects too many different routes of which many are of poor quality, and only selecting the final solutions, which selects too few. As the temperature of the Simulated Annealing is below β , the routes of the solution at that moment are likely of decent quality, decreasing the number of useless columns.

4.2 ILP optimization

After the Simulated Annealing algorithm is finished, the columns are passed to the ILP to optimize the solution. It is a set covering model which picks a subset of columns included in the model, minimizing the costs. These costs are calculated during the local search phase and saved alongside the route. As almost all constraints are checked during the local search phase, the only constraints that need to be enforced are the constraints between routes. In this model, the only constraints that need to be enforced are that every customer is served exactly once by the solution and that the total number of routes is less than or equal to the number of available vehicles. The mathematical model is:

$$\min \sum_{r \in R} x_r c_r \quad (20)$$

$$\sum_{r \in R} x_r a_{ri} = 1 \quad \forall i \in C \quad (21)$$

$$\sum_{r \in R} x_r \leq m \quad (22)$$

$$x_r, c_r, a_{ri} \in \{0, 1\} \quad \forall i \in C, r \in R \quad (23)$$

Where x_r denotes whether route r is chosen, c_r denotes the cost of route r , a_{ri} denotes whether route r contains customer i and m denotes the number of available vehicles. This does assume a homogeneous fleet of vehicles, which is the case in the model to be implemented. C denotes the set of all customers and lastly, R denotes the set of all selected routes after the local search step.

To speed up the execution of the ILP it incorporates warm starts. In a warm start, the ILP is initialized with some existing (partial) solution, from which the search can start. As the simulated annealing algorithm already provided a valid and probably reasonably good solution it can be used for such a warm start. Therefore, the best final solution of the first phase is used as the initial solution of this ILP.

5 Experiments

5.1 Data

All experiments were performed on instances created by Fontaine[33] and can be found online here[34]. These instances consist of 5 cities, namely: Madrid, Fukuoka, Pittsburgh, Sydney, and Seattle. For each of these instances, 100 customers are defined. Each customer has a location defined as latitude and longitude, demand, time window, and service time. Furthermore, for each customer, the distance to each other customer, calculated using google maps is given. The tests are also run on the subsets of the first 50 and 75 customers giving 15 instances in total. During these experiments, we assume the same characteristics for the bicycles as [33], where the bike and cyclist combined have an empty weight of $140kg$, a maximum capacity of $150kg$ and the bicycle and cyclist combined produce $350W$ of power. Furthermore, in similar fashion to [33] we use 10 load levels for our experiments.

Finally, for testing the impact of the introduction of a constant wind an extra instance was created representing the city of Utrecht in the Netherlands. Utrecht was chosen as it is the city where Utrecht University is located and it is relatively flat, which might give an interesting extra comparison when under the influence of wind. This was created using randomly generated points 2km around the city center of Utrecht. These points were converted to real addresses using the OSRM distance API[37], which also calculated the distance between each location. OSRM is based on Open Street Maps and produces similar results to Google maps. Next for each of these new addresses, the elevation was retrieved from the public Open Topo Data API. [35] using the eudem25m dataset.

To calculate the influence of the wind using equation (19) we need to know the direction of the wind experienced by the cyclist in wind still conditions \vec{e}_c . This is created by the cyclist moving and is directed opposite of the direction of travel. This direction of travel is calculated assuming a direct route from the origin customer x_o to the destination customer x_d in the instances discussed previously. However, to calculate this direction we need the Cartesian coordinates of x_o and x_d , while the VRPLTT instances provide latitude and longitude coordinates from the geographic coordinate system or GCS. To convert from GCS to a Cartesian coordinate system, the equirectangular projection was used[8]. This is a simple projection system that is suitable for projections of small areas, such as the used instances. In this approximation, we ignore the influence of any buildings and height differences on the wind to keep to model relatively simple.

5.2 Experimental setup

All experiments explained in this section were performed on a machine powered by an *Intel(R) Xeon(R) Gold 6130* CPU running at 2.1GHz with 16 cores and 32 threads, 128GB ram, and running *Ubuntu 18.04.6*. During the testing of our algorithm, some other experiments using a single core were running on the same machine. The algorithm was implement in *C# .NET 6.0* and is publicly available in this repository [36]. The ILP optimization is implemented in Gurobi 9.5.0 using the *C#* API.

5.2.1 Parameter configuration

For all experiments, the following parameters were chosen based on preliminary testing. More details of these tests can be found in the appendices.

- The initial temperature T is set at 1
- The temperature is updated every 10000 iterations
- The cooling parameter α is set as 0.99. Every time the temperature is updated, the current temperature is multiplied with α

- The search restarts if the temperature is below 0.02 and there has not been an improvement in the past 600000 iterations
- The local search restarts 7 times
- The penalty for being late, P_{late} , is 2 plus the number of minutes by which the vehicle t_{late} is late and grows linearly with temperature decrease.

$$P_{late} = \frac{2 + t_{late}}{T}$$

- The columns save parameter β as explained in Section 4.1.4 is set at 0.2 and columns are saved after improving as well as deteriorating the solution.
- 16 multistarts are used to stabilize the performance
- The ILP has a time limit of 1 hour

5.2.2 Experiments

Each VRPLTT instance was run 5 times and for each, the solution, score for the entire algorithm, local search score, local search, and ILP time were recorded. Each experiment was repeated 5 times to match the testing methodology of Fontaine[33] and make direct comparison possible.

The impact of introducing wind was tested on each of the VRPLTT instances as defined by Fontaine[33] and the newly introduced Utrecht instance. It was tested with wind directed in each of the 4 cardinal directions with a wind speed of $6.75m/s$. This wind speed was chosen as it is the average of wind force 4 on the Beaufort scale, which represents a moderate breeze. Each instance was run 5 times. For the tests including wind the ILP time limit was reduced to 1200 seconds as the main focus of these experiments is the impact of the wind on the solution, other experiments showed running it longer did not improve the solutions significantly and in the interest of time.

To see if the wind changes the way the vehicles are routed we compared the results in multiple ways. Firstly, we directly compare the resulting solutions. Secondly, we introduce the measure wind time (WT), which is based on the time spent cycling against the wind direction to compare to the case without any wind. For each edge (i, j) and load level l combination (i, j, l) taken by the solution we first calculate the time it would have taken to travel that edge without any wind T_{ij}^0 . Then using the direction opposite of the travel direction, $e_{cij}^{\vec{}}$, and the direction of the wind $e_w^{\vec{}}$, which both are $2D$ vectors of length 1, we can calculate the cosine of the angle between the wind and direction of travel by taking a dot product between these two directions. This essentially calculates to what extent the wind helps or hinders the cyclist. By multiplying this dot product with T_{ij}^0 we get a value for each edge, which is positive if the wind comes from the front, 0 if it comes directly from the sides, and negative if it comes from behind. Summing this value for each edge in the solution R we obtain WT

$$WT = \sum_{(i,j,l) \in R} (e_{cij}^{\vec{}} \cdot e_w^{\vec{}}) T_{ij}^0 \quad (24)$$

In a simple model, where the travel time between two customers is constant, this measure would always return the value 0. However, as we have limited input power, a maximum travel speed, the load of the vehicle, and slopes to take into account it is not zero in our case.

5.3 Results VRPLTT

The results of the SA-ILP algorithm on the default VRPLTT instances created by Fontaine[33] are shown in Table 1. These are compared to the reported results in [33] for the ALNS with a 10-fold iteration limit or ALNS (long). Fontaine[33] reported the results for both the ALNS algorithm and the ALNS

algorithm with a 10-fold iteration limit, the latter producing slightly better scores. Therefore, the ALNS (long) was chosen for comparison. For each instance the average score over 5 runs and the best score over 5 runs are reported, where the best of each of those values is highlighted. If both algorithms are tied on one of the reported scores, neither of them is highlighted.

Table 1: Scores of 5 runs compared to the scores of ALNS (long)

C	network	SA-ILP		ALNS (long)		diff		LS	ILP	Total
		avg 5	best 5	avg 5	best 5	avg(%)	best(%)	time(s)	time(s)	time(s)
50	Fukuoka	73.87	73.87	73.95	73.87	0.11	0.00	168.54	2.77	171.31
50	Madrid	73.97	73.97	74.32	74.32	0.47	0.47	145.44	0.51	145.95
50	Pittsburgh	87.07	87.07	87.18	87.10	0.13	0.03	155.80	1.31	157.11
50	Seattle	58.93	58.93	59.16	59.00	0.39	0.12	161.64	76.31	237.95
50	Sydney	95.34	95.34	95.28	95.28	-0.06	-0.06	156.46	6.17	162.63
75	Fukuoka	95.15	95.15	95.73	95.39	0.61	0.25	162.27	3601.04	3763.31
75	Madrid	97.69	97.69	98.32	97.67	0.64	-0.02	149.91	7.59	157.50
75	Pittsburgh	110.71	110.71	111.65	111.16	0.84	0.40	173.37	10.32	183.69
75	Seattle	79.83	79.83	80.01	80.01	0.22	0.22	190.66	48.09	238.75
75	Sydney	122.51	122.39	123.51	123.42	0.81	0.83	135.09	25.24	160.33
100	Fukuoka	118.87	118.87	119.62	119.34	0.63	0.39	176.41	3609.54	3785.95
100	Madrid	124.96	124.80	127.27	126.64	1.82	1.45	164.22	11.88	176.10
100	Pittsburgh	134.32	133.66	136.61	135.69	1.68	1.50	171.44	93.99	265.43
100	Seattle	101.32	101.32	102.36	102.14	1.02	0.80	180.24	402.31	582.55
100	Sydney	149.65	149.46	151.51	151.05	1.23	1.05	150.62	2652.79	2803.41

In most instances, the SA-ILP outperforms the ALNS (long) in terms of travel times or score. On some small instances, the ALNS (long) produces equal or slightly better results, although the difference remains small. The differences between the SA-ILP and the results by Fontaine[33] are smaller in general for the small instances. On the medium-sized instances the ALNS (long) scores slightly better on the Madrid instance, but for every other instance, the SA-ILP improves on the ALNS(long). The gaps between the SA-ILP and ALNS (long) become slightly larger as well compared to the small instances, especially on average, indicating stable performance of the SA-ILP algorithm.

For the largest instances of 100 customers, the SA-ILP scores better for every instance and the gaps are again larger than for the previously discussed instances. The differences are especially large for the Madrid and Pittsburgh instances, where the average gap is more than 1.5%

This performance does come at the cost of execution time for some instances, especially the medium Fukuoka and large Fukuoka and Sydney instances. In these instances, the ILP execution hits or gets close to the time limit in most cases. One interesting notion about the execution time of the LS is that there is no significant difference between the instance sizes. This is most likely due to the identical vehicles used in all instances. The computation time is mostly dependent on the length of the routes that are changed to create a neighbor. Because the capacity is equal for the smaller instances, fewer vehicles are used, and therefore the number of customers in each route stays roughly equal (13.16 for large and 12.50 for small instances on average)

Table 2: ILP improvement

C	network	AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		LS+ILP	LS	imp(%)	LS+ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	0.00	73.87	73.87	0.00	171.32	2.77	168.54
50	Madrid	73.97	73.97	0.00	73.97	73.97	0.00	145.95	0.51	145.44
50	Pittsburgh	87.07	87.07	0.00	87.07	87.07	0.00	157.12	1.31	155.80
50	Seattle	58.93	58.93	0.00	58.93	58.93	0.00	237.95	76.31	161.64
50	Sydney	95.34	95.34	0.00	95.34	95.34	0.00	162.63	6.17	156.46
75	Fukuoka	95.15	95.15	0.00	95.15	95.15	0.01	3763.31	3601.04	162.27
75	Madrid	97.69	97.69	0.00	97.69	97.69	0.00	157.50	7.59	149.91
75	Pittsburgh	110.71	110.71	0.01	110.71	110.75	0.04	183.68	10.32	173.37
75	Seattle	79.83	79.83	0.00	79.83	79.83	0.00	238.75	48.09	190.66
75	Sydney	122.51	122.71	0.17	122.39	122.79	0.32	160.33	25.24	135.09
100	Fukuoka	118.87	118.91	0.03	118.87	118.87	0.00	3785.95	3609.54	176.41
100	Madrid	124.96	125.45	0.39	124.80	125.80	0.80	176.11	11.88	164.22
100	Pittsburgh	134.32	135.18	0.64	133.66	134.33	0.50	265.43	93.99	171.44
100	Seattle	101.32	101.55	0.22	101.32	101.72	0.38	582.56	402.31	180.24
100	Sydney	149.65	149.94	0.19	149.46	150.02	0.37	2803.41	2652.79	150.62

Although the ILP takes quite a lot of time it also provides improvements to the solution. Table 2 shows the scores with and without the ILP inclusion and the percentage of improvement it provides. These numbers are given for the average and best of the same 5 runs as in Table 1. For the case of the best solution, the local search score belonging to the best full score is given, and not the best local search score. For the smallest instances, the ILP does not provide any improvement over the results from the SA algorithm, but it also has small computing times compared to the other instances. For the medium instances, the ILP does provide some improvement, but also requires more computing time. However, the most improvement comes from the largest instances. This might be expected as in these instances there are a lot more possible routes and therefore more possibilities for the SA algorithm to miss the best combination. This makes it possible for the ILP to improve the solution more than in the other instances. In some of the cases, such as for the Madrid instance with 100 customers, the minimum score of the 5 runs shows that the ILP has the ability to even out the scores when the local search performs worse than on average. It seems to stabilize the performance of the algorithm. Overall the ILP optimization seems promising for larger and more complicated instances, but less useful for the smaller instances.

However, looking at the average execution time of the ILP phase and the accompanied average improvement the question arises whether a time limit of an hour is reasonable. The ILP reaches the time limit on both the 75 and 100 customer cases of the Fukuoka instance, while not providing any large improvements. Furthermore, in most cases where the ILP provides more improvement the execution time of the ILP remains below 10 minutes, with Sydney with 100 customers being the only exception. The execution time might, therefore, be limited further to decrease the total execution time.

5.4 Results wind

The results regarding the constant wind are shown in Tables 3-6, each for a different wind direction. Each table shows the average WT value, the length of the route with the wind included, and the percentage of routes that are still valid when the wind is included for both planning with and without the wind. Logically, when planning with the wind all routes are valid as they are planned for that wind. Finally, the execution times for the algorithm considering the wind during planning are given.

We can see that for each wind direction and instance combination the WT value is lower when planning with the wind compared to without the wind. This indicates that the algorithm does make different choices to try to avoid the wind as much as possible. This smart planning around the wind can also be seen in the total length of the route. The routes found when ignoring the wind during planning, are longer on average compared to when we are planning the routes with the wind in mind.

This difference can be seen for each wind direction and instance combination, indicating that the avoiding of the wind we concluded when analyzing the *WT* values also reduces the length of the solutions. One exception to this is the Utrecht instance with a southerly wind. In this instance, taking wind into account produces a longer solution. One probable explanation for this is that there might be some route, which is significantly faster than its alternatives, that is not valid anymore under the southerly wind and therefore not taken when planning while considering this wind.

The improvement of taking the wind into account differs quite a bit. On some instance and wind direction combinations, differences in route length of up to 3 percent can be seen, while others, such as the Pittsburgh instance with a northerly wind, only differ by about 0.1%. However, on average it does provide an improvement on all instance and wind combinations.

Another interesting notion is that many routes planned by the algorithm ignoring the wind on the system are not valid anymore with the wind. It does not manage to arrive at all customers on time. On average for each instance and wind direction combination at most 90% of the routes planned are still valid under the influence of the wind, while for the Sydney instance with westerly wind, less than 55% of the routes are still valid on average.

Table 3: Wind results for westerly wind

C	network	With wind			Without wind			Length diff (%)	Total time (s)	ILP time (s)	LS time (s)
		WT	Length (min)	Valid (%)	WT	Length (min)	Valid (%)				
100	Fukuoka	-2.46	134.50	100.00	1.10	134.88	72.50	0.28	1368.78	1200.54	168.24
100	Madrid	-1.96	147.96	100.00	4.12	149.99	75.00	1.37	162.58	9.53	153.05
100	Pittsburgh	-35.02	142.11	100.00	-20.44	144.71	57.50	1.83	209.41	38.88	170.53
100	Seattle	-39.20	99.83	100.00	-37.69	100.34	90.00	0.51	705.05	526.79	178.26
100	Sydney	-30.79	163.33	100.00	-16.50	168.25	54.29	3.01	266.39	116.22	150.17
100	Utrecht	-2.16	192.34	100.00	2.82	194.27	75.00	1.00	174.57	0.75	173.82

Table 4: Wind results for easterly wind

C	network	With wind			Without wind			Length diff (%)	Total time (s)	ILP time (s)	LS time (s)
		WT	Length (min)	Valid (%)	WT	Length (min)	Valid (%)				
100	Fukuoka	-3.31	131.64	100.00	-1.10	132.81	85.00	0.89	1372.57	1200.37	172.20
100	Madrid	-11.89	139.38	100.00	-4.12	141.44	70.00	1.48	181.09	15.89	165.20
100	Pittsburgh	12.15	158.81	100.00	20.44	160.95	55.00	1.35	180.57	13.49	167.08
100	Seattle	30.28	114.10	100.00	37.69	116.00	75.00	1.67	342.94	169.27	173.67
100	Sydney	10.06	184.94	100.00	16.50	187.22	57.14	1.23	686.09	540.17	145.92
100	Utrecht	-3.12	191.25	100.00	-2.82	191.43	75.00	0.09	175.89	0.98	174.91

Table 5: Wind results for southerly wind

C	network	With wind			Without wind			Length diff (%)	Total time (s)	ILP time (s)	LS time (s)
		WT	Length (min)	Valid (%)	WT	Length (min)	Valid (%)				
100	Fukuoka	2.50	154.76	100.00	10.74	157.46	72.50	1.74	1360.35	1200.73	159.62
100	Madrid	-25.48	133.93	100.00	-14.85	136.01	90.00	1.55	178.18	13.20	164.98
100	Pittsburgh	-20.20	140.80	100.00	-9.02	142.47	80.00	1.19	261.89	90.92	170.97
100	Seattle	-24.38	103.79	100.00	-15.40	104.63	77.50	0.81	1039.77	862.28	177.49
100	Sydney	4.72	165.12	100.00	7.88	167.03	57.14	1.16	744.41	592.50	151.91
100	Utrecht	-2.53	190.91	100.00	-2.28	190.10	75.00	-0.42	177.13	0.57	176.56

Table 6: Wind results for northerly wind

C	network	With wind			Without wind			Length diff (%)	Total time (s)	ILP time (s)	LS time (s)
		WT	Length (min)	Valid (%)	WT	Length (min)	Valid (%)				
100	Fukuoka	-13.63	138.46	100.00	-10.74	139.46	72.50	0.72	1369.11	1200.34	168.77
100	Madrid	10.58	148.71	100.00	14.85	149.51	57.50	0.54	165.40	4.78	160.62
100	Pittsburgh	5.62	150.75	100.00	9.02	150.97	51.79	0.15	298.56	115.65	182.91
100	Seattle	12.67	111.78	100.00	15.40	113.13	77.50	1.21	466.33	298.26	168.07
100	Sydney	-18.51	160.69	100.00	-7.88	163.10	68.57	1.50	345.68	195.98	149.70
100	Utrecht	-3.31	189.16	100.00	2.28	189.87	75.00	0.38	179.52	0.88	178.64

Figures 4 -7 in Appendix F show 4 different solutions from the Pittsburgh instance under different wind directions. Pittsburgh was chosen as this showed the most difference in the numeric results. The wind directions are displayed as an arrow to the left of the image. The routes in each solution are color-coded to match the most similar route in each image. The similarity was calculated using an adapted form of the Levenshtein distance. This Levenshtein distance is calculated between a pair of routes and denotes the number of customers that need to be replaced with a different customer to change one route into the other. These images show that there are some significant differences between the different solutions. For example, comparing the orange route in Figure 4 with the same route in Figure 5, we can see the difference in the routing. When the wind is coming from the east we can see the orange route seemingly taking this wind from behind as long as possible, while when the wind is coming from the east a part of the route is reversed to again seemingly take as much wind advantage as possible. Looking at the yellow route in Figures 6 and 7 other significant changes can be found. Here Figure 6 seemingly takes the route as far north as possible, using as much tailwind as possible, while in Figure 7 it seemingly stops earlier and uses a different southwards path to use more of the wind blowing to the south. The routes found under the different wind circumstances are somewhat different. It shows that the algorithm apparently tries to find ways around the wind to minimize its impact.

6 Conclusion

We presented a new hybrid algorithm for the vehicle routing problem with load-dependent travel times consisting of a simulated annealing based local search phase, which generates columns for a set covering model, which optimizes the results further. This algorithm outperformed the originally proposed algorithm for this model in almost all instances tested in our computational experiments.

The ILP optimization provided reasonable improvements for the larger instances and seems a promising addition to our simulated annealing method for more complicated instances. It also is a relatively easy extension to implement in such an algorithm which could be implemented in many other algorithms.

Furthermore, we presented how one can model a constant wind into the VRPLTT and change the travel times according to the influence of this wind. This could also be implemented in any other VRP model which uses travel times that are decoupled from the traveled distance. Our computational experiments show that our proposed algorithm planning under this constant wind tries to avoid headwinds. It improves the solution compared to ignoring the wind during the planning and reduces the travel times, while also preventing routes for which the time windows can no longer be met because the wind was ignored during the planning.

6.1 Future research

The proposed algorithm shows good performance on the VRPLTT instances. However, some avenues can still be explored. For example, experimentation could be done regarding selecting the right columns for the second phase. Furthermore, the algorithm may be tried on other models as well. Additionally, testing the algorithm on larger and more complicated VRPLTT instances might be an interesting step as well.

The computational experiments of the constant wind showed interesting results. This constant wind could also be incorporated into other VRP models to test its impact. Furthermore, experiments could be conducted where the wind is not constant over the planning horizon, but changes over time. This might further incentivize different planning approaches to the solution.

Acknowledgements

First of all, this author would like to thank his supervisors Dr. ir. J.M. van den Akker and Dr. J.A. Hoogeveen for their help and feedback throughout the process of this master thesis. Secondly, this author would like to thank L.P. Stoop for providing access to his server for our computational experiments. Thirdly, the author would like to thank Dr. R.M. van Westen, Dr. M.L.J. Baatsen, and D. Kuijper from the physics department for helping to create and validate the calculations for the air resistance under a constant wind. Lastly, the author would like to thank P. Fontaine, for answering his questions and providing reference code.

References

- [1] G. B. Dantzig and J. H. Ramser. “The Truck Dispatching Problem”. In: *Management Science* 6.1 (1959), pp. 80–91. DOI: 10.1287/mnsc.6.1.80. eprint: <https://doi.org/10.1287/mnsc.6.1.80>. URL: <https://doi.org/10.1287/mnsc.6.1.80>.
- [2] G. Clarke and J. W. Wright. “Scheduling of Vehicles from a Central Depot to a Number of Delivery Points”. In: *Operations Research* 12.4 (1964), pp. 568–581. ISSN: 0030364X, 15265463. URL: <http://www.jstor.org/stable/167703>.
- [3] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi. “Optimization by simulated annealing”. In: *science* 220.4598 (1983), pp. 671–680.
- [4] M. M. Solomon. “Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints”. In: *Operations Research* 35.2 (1987), pp. 254–265. DOI: 10.1287/opre.35.2.254. eprint: <https://doi.org/10.1287/opre.35.2.254>. URL: <https://doi.org/10.1287/opre.35.2.254>.
- [5] G. Laporte, F. Louveaux, and H. Mercure. “The Vehicle Routing Problem with Stochastic Travel Times”. In: *Transportation Science* 26.3 (1992), pp. 161–170. ISSN: 00411655, 15265447. URL: <http://www.jstor.org/stable/25768536>.
- [6] Y. Rochat and É. D. Taillard. “Probabilistic diversification and intensification in local search for vehicle routing”. In: *Journal of heuristics* 1.1 (1995), pp. 147–167.
- [7] M. Gendreau, G. Laporte, and R. Séguin. “Stochastic vehicle routing”. In: *European Journal of Operational Research* 88.1 (1996), pp. 3–12. ISSN: 0377-2217. DOI: [https://doi.org/10.1016/0377-2217\(95\)00050-X](https://doi.org/10.1016/0377-2217(95)00050-X). URL: <https://www.sciencedirect.com/science/article/pii/037722179500050X>.
- [8] J. P. Snyder. *Flattening the earth: two thousand years of map projections*. University of Chicago Press, 1997.
- [9] G. B. Alvarenga, G. R. Mateus, and G. de Tomi. “A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows”. In: *Computers Operations Research* 34.6 (2007). Part Special Issue: Odysseus 2003 Second International Workshop on Freight Transportation Logistics, pp. 1561–1584. ISSN: 0305-0548. DOI: <https://doi.org/10.1016/j.cor.2005.07.025>. URL: <https://www.sciencedirect.com/science/article/pii/S0305054805002376>.
- [10] J.-F. Cordeau, G. Laporte, M. W. Savelsbergh, and D. Vigo. “Vehicle routing”. In: *Handbooks in operations research and management science* 14 (2007), pp. 367–428.

- [11] G. Laporte. “What you should know about the vehicle routing problem”. In: *Naval Research Logistics (NRL)* 54.8 (2007), pp. 811–819. DOI: <https://doi.org/10.1002/nav.20261>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.20261>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.20261>.
- [12] T. Stewart, L. W. G. Strijbosch, J. J. A. Moors, and P. van Batenburg. *A Simple Approximation to the Convolution of Gamma Distributions (Revision of DP 2006-27)*. English. WorkingPaper. Pagination: 24. Operations research, 2007.
- [13] S. Nadarajah and S. Kotz. “Exact Distribution of the Max/Min of Two Gaussian Random Variables”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 16.2 (2008), pp. 210–212. DOI: 10.1109/TVLSI.2007.912191.
- [14] T. Bektaş and G. Laporte. “The Pollution-Routing Problem”. In: *Transportation Research Part B: Methodological* 45.8 (2011). Supply chain disruption and risk management, pp. 1232–1250. ISSN: 0191-2615. DOI: <https://doi.org/10.1016/j.trb.2011.02.004>. URL: <https://www.sciencedirect.com/science/article/pii/S019126151100018X>.
- [15] D. Koning. “Using Column Generation for the Pickup and Delivery Problem with Disturbances”. MA thesis. Department of Information and Computing Sciences, Utrecht University, June 2011.
- [16] V. C. Hemmelmayr, J.-F. Cordeau, and T. G. Crainic. “An adaptive large neighborhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics”. In: *Computers Operations Research* 39.12 (2012), pp. 3215–3228. ISSN: 0305-0548. DOI: <https://doi.org/10.1016/j.cor.2012.04.007>. URL: <https://www.sciencedirect.com/science/article/pii/S0305054812000871>.
- [17] S. N. Parragh and V. Schmid. “Hybrid column generation and large neighborhood search for the dial-a-ride problem”. In: *Computers Operations Research* 40.1 (2013), pp. 490–497. ISSN: 0305-0548. DOI: <https://doi.org/10.1016/j.cor.2012.08.004>. URL: <https://www.sciencedirect.com/science/article/pii/S0305054812001694>.
- [18] A. Subramanian, E. Uchoa, and L. S. Ochi. “A hybrid algorithm for a class of vehicle routing problems”. In: *Computers Operations Research* 40.10 (2013), pp. 2519–2531. ISSN: 0305-0548. DOI: <https://doi.org/10.1016/j.cor.2013.01.013>. URL: <https://www.sciencedirect.com/science/article/pii/S030505481300021X>.
- [19] D. Taş, N. Dellaert, T. van Woensel, and T. de Kok. “Vehicle routing problem with stochastic travel times including soft time windows and service costs”. In: *Computers Operations Research* 40.1 (2013), pp. 214–224. ISSN: 0305-0548. DOI: <https://doi.org/10.1016/j.cor.2012.06.008>. URL: <https://www.sciencedirect.com/science/article/pii/S0305054812001402>.
- [20] J. G. Villegas, C. Prins, C. Prodhon, A. L. Medaglia, and N. Velasco. “A matheuristic for the truck and trailer routing problem”. In: *European Journal of Operational Research* 230.2 (2013), pp. 231–244. ISSN: 0377-2217. DOI: <https://doi.org/10.1016/j.ejor.2013.04.026>. URL: <https://www.sciencedirect.com/science/article/pii/S037722171300324X>.
- [21] D. Taş, M. Gendreau, N. Dellaert, T. van Woensel, and A. G. de Kok. “Vehicle routing with soft time windows and stochastic travel times: A column generation and branch-and-price solution approach”. In: *European Journal of Operational Research* 236.3 (2014). Vehicle Routing and Distribution Logistics, pp. 789–799. ISSN: 0377-2217. DOI: <https://doi.org/10.1016/j.ejor.2013.05.024>. URL: <https://www.sciencedirect.com/science/article/pii/S0377221713004232>.
- [22] J. F. Ehmke, A. M. Campbell, and T. L. Urban. “Ensuring service levels in routing problems with time windows and stochastic travel times”. In: *European Journal of Operational Research* 240.2 (2015), pp. 539–550. ISSN: 0377-2217. DOI: <https://doi.org/10.1016/j.ejor.2014.06.045>. URL: <https://www.sciencedirect.com/science/article/pii/S0377221714005487>.

- [23] E. E. Zachariadis, C. D. Tarantilis, and C. T. Kiranoudis. “The load-dependent vehicle routing problem and its pick-up and delivery extension”. In: *Transportation Research Part B: Methodological* 71 (2015), pp. 158–181. ISSN: 0191-2615. DOI: <https://doi.org/10.1016/j.trb.2014.11.004>. URL: <https://www.sciencedirect.com/science/article/pii/S0191261514001933>.
- [24] E. Bombach. *Traktionskonzept für ein Lastenfahrzeug*. Springer, 2016.
- [25] M. M. Lekkerkerker. “Robust scheduling of the vehicle routing problem with time windows”. MA thesis. Department of Information and Computing Sciences, Utrecht University, Sept. 2016.
- [26] A. Montoya, C. Guéret, J. E. Mendoza, and J. G. Villegas. “The electric vehicle routing problem with nonlinear charging function”. In: *Transportation Research Part B: Methodological* 103 (2017). Green Urban Transportation, pp. 87–110. ISSN: 0191-2615. DOI: <https://doi.org/10.1016/j.trb.2017.02.004>. URL: <https://www.sciencedirect.com/science/article/pii/S0191261516304556>.
- [27] V. F. Yu, A. P. Redi, Y. A. Hidayat, and O. J. Wibowo. “A simulated annealing heuristic for the hybrid vehicle routing problem”. In: *Applied Soft Computing* 53 (2017), pp. 119–132. ISSN: 1568-4946. DOI: <https://doi.org/10.1016/j.asoc.2016.12.027>. URL: <https://www.sciencedirect.com/science/article/pii/S1568494616306524>.
- [28] Á. Corberán, G. Erdoğan, G. Laporte, I. Plana, and J. M. Sanchis. “The Chinese Postman Problem with Load-Dependent Costs”. In: *Transportation Science* 52.2 (2018), pp. 370–385. DOI: 10.1287/trsc.2017.0774. eprint: <https://doi.org/10.1287/trsc.2017.0774>. URL: <https://doi.org/10.1287/trsc.2017.0774>.
- [29] G. P. T. van Lent. “Using column generation for the Time Dependent Vehicle Routing Problem with Soft Time Windows and Stochastic Travel Times”. MA thesis. Department of Information and Computing Sciences, Utrecht University, Jan. 2018.
- [30] J. Villegas, C. Guéret, J. E. Mendoza, and A. Montoya. “The technician routing and scheduling problem with conventional and electric vehicle”. working paper or preprint. June 2018. URL: <https://hal.archives-ouvertes.fr/hal-01813887>.
- [31] G. Hiermann, R. F. Hartl, J. Puchinger, and T. Vidal. “Routing a mix of conventional, plug-in hybrid, and electric vehicles”. In: *European Journal of Operational Research* 272.1 (2019), pp. 235–248. ISSN: 0377-2217. DOI: <https://doi.org/10.1016/j.ejor.2018.06.025>. URL: <https://www.sciencedirect.com/science/article/pii/S0377221718305605>.
- [32] D. G. Wilson and T. Schmidt. *Bicycling science*. MIT press, 2020.
- [33] P. Fontaine. “The vehicle routing problem with load-dependent travel times for cargo bicycles”. In: *European Journal of Operational Research* (2021). ISSN: 0377-2217. DOI: <https://doi.org/10.1016/j.ejor.2021.09.009>. URL: <https://www.sciencedirect.com/science/article/pii/S0377221721007682>.
- [34] P. Fontaine. *Datasets of the VRPLTT*. 2021. URL: <https://github.com/FontainePirmin/vrpltt> (visited on 06/01/2022).
- [35] ajnisbet. *Open Topo Data public API*. 2022. URL: <https://www.opentopodata.org/#public-api> (visited on 06/01/2022).
- [36] S. C. Hesselmann. *SA-ILP Code*. 2022. URL: <https://github.com/samhesselmann/SA-ILP> (visited on 07/10/2022).
- [37] Project-OSRM. *Open Source Routing Machine*. 2022. URL: <http://project-osrm.org/> (visited on 06/01/2022).

A Selecting the right neighborhoods

Many different combinations of different neighborhoods were tried to find the best possible configuration for our simulated annealing algorithm. In this Appendix, we will discuss the differences between those configurations and the different results obtained using them and discuss why the final configuration was chosen. We considered the following neighborhoods for our algorithm

1. Move random customer

Randomly choose two customers c_1 and c_2 . Insert c_1 directly before c_2 . This classical neighborhood is employed in many different simulated annealing implementations.

2. Swap random customers

Randomly select two random customers c_1 and c_2 and swap these customers such that c_1 is positioned where c_2 was and c_2 where c_1 was.

3. Swap random customers inside a route

Randomly select two random customers c_1 and c_2 from the same route and swap these customers such that c_1 is positioned where c_2 was and c_2 where c_1 was.

4. Reverse a random subroute

First, a random route is selected. For this route two non equal customers c_1 and c_2 are randomly selected, after which the sub route from c_1 to c_2 is reversed including the randomly selected customers. This neighborhood is sometimes also referred to as 2-opt in literature.

5. Scramble a random subroute

First, a random route is selected. In this route two customers are randomly selected, after which the sub route between the two customers is put into a random order, scrambling part of the route. This neighborhood allows larger changes to the solution at once, which can shake up the current solution more.

6. Remove a random customer

Randomly select a customer and remove it from the current solution. This customer is stored for later addition to the solution. A penalty is incurred for each removed customer, which is dependent on temperature, such that it is more rewarding to generate feasible solutions when the temperature is lower

7. Add a random removed customer

Randomly select one of the removed customers and add it to a random position in a random route.

8. Move a random customer to the best position in a random route

Randomly choose a customer c from one of the existing routes and randomly select a random route r . Then, customer c is inserted into r in the position with the smallest increase in objective value.

9. Move a random customer to its best possible position in any route except for the route it is currently located in

Randomly select a customer and insert it in the best position in any of the other routes. By not considering the route in which the randomly selected customer is located the customer is always moved if the neighbor is accepted.

These were also tested with multiple repetitions. That is, if a neighborhood is repeated n times, we choose the best neighbor found in these n repetitions and use that neighbor as the result of the neighborhood. This increases the probability of the neighborhood providing an improvement. This does, however, make the neighborhoods slightly more greedy which could hamper the results later during the execution. Therefore, we tested neighborhoods with and without repetitions and some with different amounts of repetitions. During the development of the algorithm, many different configurations were tried, but the following configurations were tested more extensively.

1. All neighborhoods selected and no repetitions
2. All neighborhoods selected where all simple neighborhoods are repeated 4 times. Here, all simple neighborhoods are neighborhoods 1, 2, 3, 6 and 7.
3. All neighborhoods selected, where all simple neighborhoods are repeated 4 times except add and remove. Thus neighborhoods 1, 2 and 3 are repeated 4 times
4. All neighborhoods selected where neighborhoods 2, 3 and 8 are repeated 4 times
5. All neighborhoods selected, where neighborhoods 2 and 3 are repeated 4 times and neighborhood 8 is repeated 2 times
6. All neighborhoods except 7 and 6 are selected, where neighborhoods 2, 3 and 8 are repeated 4 times
7. A configuration similar to configuration 4, but without the greedy neighborhoods 8 and 9.
8. A configuration similar to configuration 4, but the number of iterators per alpha change is reduced to 5000, and the number of iterations of no improvement before a restart is reduced to 200000
9. A configuration similar to configuration 4, but with a higher penalty for the add and remove operators. This penalty was increased to 2.
10. A configuration similar to configuration 9, but with two changes. The removed customer penalty was increased to 4, while the penalty was made linear in the number of customers.
11. An configuration similar to configuration 10, but with again two changes. The removed customer penalty was increased to 8, while the penalty was made linear in the temperature.
12. An configuration similar to configuration 4, but without the extra internal swap neighborhood.
13. A configuration combining configurations 6, and 12. Removing both the extra internal swap and the add and remove neighborhoods. This configuration was added based on results obtained by these two configurations. Both performed well, and combining them would create a simpler algorithm. Therefore, a combination of the two was added.

An overview of these configurations can be found in Tables 7 and 8. Table 7 contains the different configuration of parameters for each configuration. Table 8 contains the neighborhoods for each configuration. It shows the number of repetitions of the neighborhood if it is used in the configuration. Each of these configurations was run 5 times and the different scores and execution times were recorded. This list of configurations is, of course, not exhaustive. However, in the interest of time, we could not test more configurations.

Table 7: Parameter configurations for each algorithm configuration

Configuration	P_{late}	$P_{removed}$	$P_{removed}^{pow}$	$P_{removed}^{Tpow}$	α	β	$It_{achange}$	$It_{restart}$	$T_{initial}$	$T_{restrart}$	Restarts
1	2	0.1	2	2	0.99	0.2	10000	600000	1	0.02	7
2	2	0.1	2	2	0.99	0.2	10000	600000	1	0.02	7
3	2	0.1	2	2	0.99	0.2	10000	600000	1	0.02	7
4	2	0.1	2	2	0.99	0.2	10000	600000	1	0.02	7
5	2	0.1	2	2	0.99	0.2	10000	600000	1	0.02	7
6	2	0.1	2	2	0.99	0.2	10000	600000	1	0.02	7
7	2	0.1	2	2	0.99	0.2	10000	600000	1	0.02	7
8	2	0.1	2	2	0.99	0.2	5000	200000	1	0.02	7
9	2	2	2	2	0.99	0.2	10000	600000	1	0.02	7
10	2	4	1	2	0.99	0.2	10000	600000	1	0.02	7
11	2	8	1	1	0.99	0.2	10000	600000	1	0.02	7
12	2	0.1	2	2	0.99	0.2	10000	600000	1	0.02	7
13	2	0.1	2	2	0.99	0.2	10000	600000	1	0.02	7

Table 8: Neighborhood repetitions for each tested configuration

Configuration	Neighborhood									
	Move (1)	Swap (2)	Swap inside route (3)	Reverse subroute (4)	Scramble subroute (5)	Remove a random customer(6)	Add a random removed customer(7)	Move to best position in random route (8)	Move to best position in any other route (9)	
1	1	1	1	1	1	1	1	1	1	
2	4	4	4	1	1	4	4	1	1	
3	4	4	4	1	1	1	1	1	1	
4	1	4	4	1	1	1	1	4	1	
5	1	4	4	1	1	1	1	2	1	
6	1	4	4	1	1	-	-	4	1	
7	1	4	4	1	1	1	1	-	-	
8	1	4	4	1	1	1	1	4	1	
9	1	4	4	1	1	1	1	4	1	
10	1	4	4	1	1	1	1	4	1	
11	1	4	4	1	1	1	1	4	1	
12	1	4	-	1	1	1	1	4	1	
13	1	4	-	1	1	-	-	4	1	

A.1 Configuration results

In this section, we will discuss the different results obtained for each configuration. We ran the experiments for the basic VRPLTT described in section 5.1 for each configuration described in the previous

section and recorded the scores. These scores can be found in Tables 21-33 in Appendix E. These tables show the score, the LS score, and the improvement provided by the ILP for both the average and the best of the 5 runs for each instance. Table 9 shows the average result for each configuration, where results that equal the best solution found are highlighted. From this table, we can see that there is quite a large difference in performance between some of the configurations. For example, configuration 7 performs significantly worse than all other configurations. Only matching the best solution for the Fukuoka and Sydney instances with 50 customers, for which each configuration reached the same score. Furthermore, it provides the worst solution for all other instances, showing that the greedy neighborhoods 8 and 9 are quite important for the results of the algorithm. One interesting observation from the results of configuration 7 in Table 27 is that the ILP seems to provide a larger improvement over the LS in this configuration compared to other configurations. It seems that when the first stage provides worse results the ILP is able to optimize the solution further.

Another interesting difference between the configurations is that some configurations have vastly different execution times compared to other configurations. For example, while configuration 8 shown in Table 28 seems to perform slightly worse in terms of score compared to other configurations such as configurations 9 and 12 it does outperform them in terms of execution time. This same consideration between execution time and score can be seen with configuration 1.

There also is quite a big difference between the best results found for some instances, especially for the Fukuoka instance with 100 customers. While most configurations seem to achieve scores just below 119, others like configuration 8 achieve significantly lower best scores. These lower scores seem to be lucky finds as the averages on these instances do not differ nearly as much as the best-found scores.

Table 9: Average scores for all configurations for each instance. The score is highlighted if it is the best score on that instance

$ C $	Instance	1	2	3	4	5	6	7	8	9	10	11	12	13
50	Fukuoka	73.87	73.87	73.87	73.87	73.87	73.87	73.87	73.87	73.87	73.87	73.87	73.87	73.87
50	Madrid	73.97	73.97	73.97	73.97	73.97	73.97	74.01	73.97	73.97	73.97	73.97	73.97	73.97
50	Pittsburgh	87.07	87.07	87.07	87.07	87.07	87.07	87.18	87.07	87.07	87.07	87.07	87.07	87.07
50	Seattle	58.95	58.97	58.93	58.93	58.95	58.93	59.00	58.95	58.93	58.93	58.93	58.95	58.93
50	Sydney	95.34	95.34	95.34	95.34	95.34	95.34	95.34	95.34	95.34	95.34	95.34	95.34	95.34
75	Fukuoka	95.14	95.12	95.10	95.12	95.07	95.10	95.35	95.08	95.15	95.15	95.13	95.11	95.15
75	Madrid	97.83	97.75	97.88	97.72	97.80	97.69	97.95	97.77	97.71	97.78	97.72	97.71	97.69
75	Pittsburgh	110.75	110.75	110.74	110.77	110.72	110.71	111.59	110.72	110.71	110.71	110.71	110.83	110.71
75	Seattle	79.83	79.83	79.83	79.83	79.83	79.83	80.03	79.83	79.83	79.83	79.83	79.83	79.83
75	Sydney	122.75	122.68	122.67	122.68	122.94	122.69	123.64	122.85	122.56	122.62	122.62	122.65	122.51
100	Fukuoka	118.87	118.93	118.90	118.89	118.79	118.88	118.99	118.58	118.70	118.87	118.87	118.62	118.87
100	Madrid	125.04	125.09	124.96	124.99	125.02	124.96	125.80	125.02	124.91	125.02	125.12	124.96	124.96
100	Pittsburgh	134.62	134.33	134.39	133.99	134.10	134.45	136.63	134.40	134.38	134.47	134.43	134.14	134.32
100	Seattle	101.53	101.34	101.40	101.32	101.41	101.32	102.19	101.61	101.36	101.35	101.36	101.37	101.32
100	Sydney	150.01	149.91	149.82	149.85	149.86	149.74	150.87	150.03	149.69	149.81	149.71	149.87	149.65
Best solutions		4	5	5	8	5	7	2	6	9	5	6	5	9

A.2 The final configuration

For the final configuration, we mainly focused on the algorithm scores and not on the execution times. To choose the final configuration a paired two-sided student’s t-test was performed for each instance and each combination of 2 algorithms. Using a significance value of 0.05 we analyzed how many statistically significant differences there were between the tested algorithm configurations. These results are shown in Table 10. The rows of this table denote the number of instances on which a configuration is significantly worse than a vertical configuration, while the columns show the number of instances it is significantly better than the other configurations. The configurations which are not significantly worse than any other configuration are highlighted.

Table 10: The number of instances where a significant difference was found using a student’s t-test with a significance value of 0.05

Config	1	2	3	4	5	6	7	8	9	10	11	12	13
1	-	0	0	1	0	1	0	0	2	1	1	0	2
2	0	-	0	1	0	1	0	0	2	1	1	0	1
3	0	0	-	1	0	1	0	0	2	0	2	1	2
4	0	0	0	-	0	0	0	0	0	0	0	0	0
5	0	1	0	1	-	1	0	0	1	1	1	1	1
6	0	0	0	0	0	-	0	0	0	0	0	0	0
7	9	7	8	9	7	9	-	7	8	9	8	7	9
8	0	1	2	1	1	2	0	-	3	1	3	1	3
9	0	0	0	0	0	0	0	0	-	0	0	0	0
10	0	0	0	0	0	0	0	0	0	-	0	0	0
11	0	0	0	0	0	0	0	0	0	0	-	0	0
12	0	0	0	0	0	0	0	0	0	0	0	-	0
13	0	0	0	0	0	0	0	0	0	0	0	0	-

From this table 7 configurations stand out, namely 4,6, 9, 10,11, 12, and 13. These configurations did not perform significantly worse than any other configuration on any test instance and there also are no significant differences between these 7 configurations. These 7 configurations seem to perform the best on our test cases.

For the final configuration, the algorithm was chosen to be as simple as possible. Therefore, configuration 13 was chosen. It uses the least amount of neighborhoods of these 7 configurations. As stated earlier, the configuration 13 was added later as both removing the extra internal swap neighborhood and the add and remove neighborhoods did not seem to decrease the performance of the algorithm, while making it simpler overall. This new configuration proved to be successful and thus was chosen as the final configuration.

B Stochastic travel times

The third focus of this thesis is expanding the VRPLTT model to include stochastic travel times and determine the improvement in on-time performance this brings. To create the stochastic load-dependent travel times, we first calculate the corresponding load-dependent travel times. The load-dependent travel time of an edge is the time it would take to travel that section at the maximum possible speed which is calculated using the gradient of the road, the mass of the bike, and the input power. Now if we are assuming that power is constant and that the bike can't reach speeds over its maximum speed, this load-dependent travel time is the minimum time that is required to travel that section. Therefore, it will also be the minimum value of the chosen distribution. To include the stochasticity in the travel times we introduce it as a delay function on top of the deterministic travel times. This solution still encompasses the advantages of the original calculation of the travel times in the VRPLTT model, while also including uncertainty. Furthermore, as we want the stochastic extension to include the load dependency of the original model, the parameters of the distribution will also be influenced by the weight of the bicycle. This is also logical in the real world. For example, the travel time of a more heavily loaded bicycle might be influenced more by some disturbance than the same disturbance happening to a lighter bicycle. A heavier bicycle takes longer to get up to speed with the same input power and might need to be cycled more safely in general compared to the less heavily loaded bicycle.

B.1 Literature

Before we explain how this expansion was made to the VRPLTT, quickly discuss the literature touching upon similar problems within the VRP. In many VRP models, the travel time between two customers is deterministic. However, this does not hold up in the real world where disturbances such as traffic, red lights, etc. can influence the time it takes to travel between two points. These disturbances can be included by implementing stochastic travel times. The vehicle routing problem with stochastic travel times has been studied extensively. One of the first to do so were Laporte et al.[5] in 1992. They used 3 different formulations to address the problem. One chance-constrained model, which ensures that the probability that a route's travel time exceeds some length is at most some value α and 2 recourse models, which include these probabilities of exceeding some length as penalties in the objective function. They solved these using a branch and cut algorithm. However, they did not compare the results between the chance-constrained model and the recourse models as the chance-constrained model was not tested because of its similarity to traditional VRP models.

The benefit of stochastic parameters compared to deterministic parameters was later studied by Gendreau et al.[7]. They reviewed the benefit of different stochastic parameters such as stochastic travel times and stochastic demand by reviewing different algorithms. They concluded that algorithms that take stochasticity into account perform better in the real world.

In 2015 Ehmke et al.[22] implemented a feasibility check for routing problems with time windows and stochastic travel times. This feasibility check ensures that the route is on time for each customer with at least a certain probability and can be plugged into any algorithm for the VRPTW. They used normal distributions for the stochastic travel times. Independent normal distributions can be added, however, time windows might cause waiting time which means that the arrival time at a customer has to be computed as a maximum between the normally distributed arrival time and a deterministic lower bound of the time window. The resulting distribution is not normally distributed but can be approximated as such according to the authors. The authors concluded from different experiments that approximating this relation as a normal distribution was quite accurate.

Tas et al.[19] consider the vehicle routing problem with stochastic travel times including soft time windows and travel times. They introduce service costs for arriving outside the time window of a customer into the objective function by incorporating the total expected delay and earliness. They try to find a solution using a Tabu search method. In a later paper[21], they improve their solution approach by applying column generation and branch-and-price instead. They found that this outperformed their previous approach.

In 2011 Koning et al.[15] looked into the pickup and delivery problem, which is a variant of the VRP, with disturbances and time dependency. These disturbances were placed in certain areas simulating local congestion. Normal distributions were used to approximate the delays with a separate delay function for time dependency. To calculate the full delay over the entire route simulations were used. They solved the problem using column generation. They concluded that introducing a deviation to the expected travel times showed that many deterministic solutions were not very robust and were outperformed by even the simplest local search algorithms for the stochastic case.

Lekkerkerker et al.[25] extended the VRP with time windows to allow time-dependent and stochastic travel times. Their solution allowed for any probability distribution to be used and included the time dependency in these distributions. They also used simulations to calculate the probabilities of arriving on time at each customer. They applied column generation heuristics to solve the problem, using different methods to solve the pricing problem. For their experiment, they applied translated lognormal distributions for the stochastic travel times.

Later, Van Lent et al.[29] considered the VRP with time-dependent stochastic travel times with soft time windows. They derived distributions for the entire route from the individual legs of the journey for a more accurate approximation. For this, they used normal and gamma approximations. Furthermore, they also used Tree distribution approximations using the normal and gamma distributions for hard time window starts. Finally, they solved their model using column generation. They concluded that the max normal tree and the gamma tree methods worked best for approximating the distributions when using hard time windows. This work is the main source for the calculations in the next sections.

B.2 Changes to the model

To incorporate the stochastic delay into the model we need to make some slight changes to the model. We include the stochastic component by penalizing the probability of arriving early or late at a customer. Depending on the configuration, only late arrivals are penalized. We define

- $P(E_i^r)$ as the probability of arriving before the start of the time window of customer i in route r
- $P(L_i^r)$ as the probability of arriving after the end of the time window of customer i in route r

These variables are included in the objective function to penalize early and late arrival if they are not allowed. The new formulation of the objective function is

$$\min \sum_{(i,j) \in E} \sum_{l \in L} t_{ij}^l z_{ij}^l + \sum_{r \in R} \sum_{i \in r} (P(E_i^r)c_e + P(L_i^r)c_l) \quad (25)$$

Where R is the collection of routes that make up the solution, c_e is the penalty for earliness, and c_l is the penalty for lateness. This function is not a linear function and can't be included in the ILP formulation, but it can be used during the local search.

B.3 Gamma distribution

In this subsection, we will discuss the calculations required to calculate the arrival times at the customers under the stochastic delay using Gamma distributions. The Gamma distribution was chosen as the true distribution for the delay experienced along a part of the route. As we don't have any data to retrieve a correct distribution from, we had to make an educated guess ourselves. The Gamma distribution was chosen for 3 reasons. Firstly, the shape of the distribution was deemed logical in this scenario of electric bicycles and is used in literature as well [19][22][29]. Most couriers will arrive slightly after the minimal travel time as they are supported by the electrical motor and it is beneficial for them to cycle fast. Secondly, it is an additive distribution, making it easier to approximate the final distributions during the route without simulating the solution during the local search. Simulating would be very expensive because the entire route would have to be recalculated for almost every change because of the weight dependency. Lastly, the distribution starts at 0 and therefore negative values always have a probability of 0. This adheres to our assumption that the load-dependent travel times calculated are the minimum time required for traveling between two customers.

A Gamma distributed value $X \sim \text{Gamma}(\alpha, \lambda)$ is defined by its shape α and rate λ parameters. Adding two gamma distributions can be done if both distributions have the same value for the rate parameter. Therefore, we assume an equal rate parameter for each distribution in the model. Using the gamma distributions and the deterministic load-dependent travel times belonging to parts of the route we can calculate the actual arrival times at the customers. The Gamma distributions serve as a delay on top of the deterministic travel times. Therefore, the travel and arrival times consist of 2 components: a deterministic component of minimum arrival time and a stochastic component which is some Gamma distributed value.

B.3.1 Calculating arrival times without waiting

Firstly, we consider the case where early arrival at a customer is not allowed. This simplifies the calculations for the arrival times as we can do a simple addition between the distributions of different legs of the route. If early arrival is allowed a maximization between the stochastic component and a constant value is required. The result of this maximization is no longer gamma-distributed. We use a similar approach as applied by Van Lent[29] and our calculations are based on her results. However, our travel times consist of two components and therefore, the calculations are slightly different.

Each edge between customers has a deterministic travel time and a gamma distribution defined for each load level. We define the variables

- t_{ij}^l as the deterministic travel time required to travel from customer i to j with load level l
- α_{ij}^l as the scale parameter of the gamma distribution of the delay when traveling from customer i to j with load level l
- λ as the rate parameter of the gamma distribution of the delay. This is assumed equal for each delay.

Using these variables we can denote the travel time T_{ij}^l from i to j with load level l as

$$T_{ij}^l = t_{ij}^l + \text{Gamma}(\alpha_{ij}^l, \lambda)$$

Now as waiting is not allowed we can calculate the arrival times on some route r , which is a set of customers which are visited consecutively, relatively simple. To formulate this calculation we define the variables

- s_i as the service time at customer i
- d_i as the demand of customer i
- A_i^r as the arrival time at customer i in route r

Suppose we have a part of a route r consisting of customers i_1, i_2 and i_3 , where the vehicle has load l_1 when traveling from i_1 to i_2 and load l_2 when traveling from i_2 to i_3 . Using the previous definitions and the definition of the travel time between customers we can denote the arrival time at customer i_3 in route r as

$$\begin{aligned} A_{i_3}^r &= A_{i_2}^r + s_{i_2} + T_{i_2 i_3}^{l_2} \\ &= A_{i_1}^r + s_{i_1} + s_{i_2} + T_{i_1 i_2}^{l_1} + T_{i_2 i_3}^{l_2} \\ &= A_{i_1}^r + s_{i_1} + s_{i_2} + t_{i_1 i_2}^{l_1} + \text{Gamma}(\alpha_{i_1 i_2}^{l_1}, \lambda) + t_{i_2 i_3}^{l_2} + \text{Gamma}(\alpha_{i_2 i_3}^{l_2}, \lambda) \end{aligned}$$

From these equations, we can see the two components emerge. The service times are deterministic and the travel times exist of a deterministic and a stochastic component. Thus the deterministic component exists of the sum of service times and deterministic travel times, while the stochastic component consists of the sum of gamma distributions over the route. As we assumed that these gamma distributions had equal λ parameters we can calculate the sum of these Gamma distributions as a new Gamma distribution as follows

$$\text{Gamma}(\alpha_1, \lambda) + \text{Gamma}(\alpha_2, \lambda) = \text{Gamma}(\alpha_1 + \alpha_2, \lambda)$$

Thus we can write A_i^r as

$$A_{i_3}^r = A_{i_1}^r + s_{i_1} + s_{i_2} + t_{i_1 i_2}^{l_1} + t_{i_2 i_3}^{l_2} + \text{Gamma}(\alpha_{i_1 i_2}^{l_1} + \alpha_{i_2 i_3}^{l_2}, \lambda)$$

This can be combined into a sum of a deterministic value and a single gamma distribution for each point in the route. We denote the values as follows

- $\alpha_{A_i^r}$ is the shape parameter of the stochastic component of the arrival time at customer i in route r
- λ is the rate parameter of the stochastic component which is constant for every route and customer
- M_i^r is the sum of deterministic travel time and service time values and therefore the minimum arrival time.

Using these definitions we can define A_i^r as

$$A_i^r = M_i^r + \text{Gamma}(\alpha_{A_i^r}, \lambda)$$

To calculate the probability of visiting a customer on time we need the cumulative distribution function, which is the integral over the probability density function. The probability density function for the gamma distribution is defined as

$$f(x; \alpha, \lambda) = \frac{x^{\alpha-1} e^{-\lambda x} \lambda^\alpha}{\Gamma(\alpha)}$$

for $x, \alpha, \beta > 0$ and $\Gamma(\alpha)$ is the gamma function, $\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} dx$. The cumulative distribution is then

$$F(x; \alpha, \lambda) = \int_0^x f(t; \alpha, \lambda) dt = \frac{\gamma(\alpha, \lambda x)}{\Gamma(\alpha)}$$

Here $\gamma(\alpha, \lambda x)$ is the lower incomplete gamma function, which is defined as $\gamma(\alpha, \lambda x) = \int_0^{\lambda x} t^{\alpha-1} e^{-t} dt$. Now combining all the previous results we can calculate the probability of being early $P(E_i^r)$ at customer i in route r as

$$\begin{aligned} P(E_i^r) &= P(A_i^r < a_i) \\ &= P(A_i^r - M_i^r < a_i - M_i^r) \end{aligned}$$

As M_i^r encompasses all deterministic values of the arrival time, $A_i^r - M_i^r$ equals the stochastic component of A_i^r , which is a continuous distribution. Thus we can calculate $P(E_i^r)$ as follows

$$P(E_i^r) = F(\max(a_i - M_i^r, 0); \alpha_{A_i^r}, \lambda)$$

Similarly we can calculate being late $P(L_i^r)$ at customer i in route r as

$$\begin{aligned} P(L_i^r) &= P(A_i^r > b_i) \\ &= 1 - P(A_i^r \leq b_i) \\ &= 1 - P(A_i^r - M_i^r \leq b_i - M_i^r) \\ &= 1 - F(\max(b_i - M_i^r, 0); \alpha_{A_i^r}, \lambda) \end{aligned}$$

The sum of deterministic travel times and service times M_i^r is subtracted from the bounds for these calculations as the distributions only include the delay in the travel times. Furthermore, the max over that value and 0 is taken as the gamma distribution is only defined for values larger than or equal to 0. If the value is negative, the probability should be 0; thus, is this max value valid.

B.3.2 Calculating arrival times with waiting

When early arrival and waiting for the start of a customer's time window is allowed calculating the arrival times becomes a bit more complicated. We need to take a maximization between the lower bound of the customer's time window and the gamma-distributed travel time. The maximization between a constant and a gamma-distributed variable is not gamma-distributed. However, we can approximate this with a gamma-distributed value. Figure 2 shows an example of the case where waiting is allowed at some customer i in route r . We need to calculate the distribution of the maximum of a_i and the Gamma distribution shown in this figure.

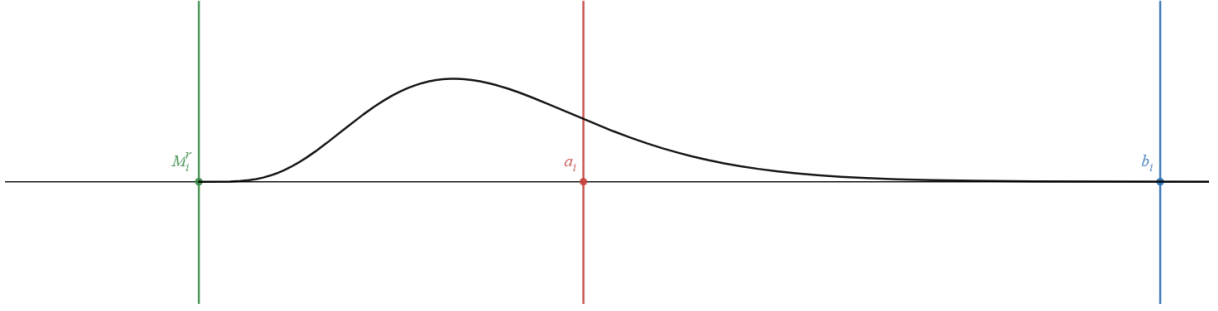


Figure 2: Example of the case where waiting is allowed with the Gamma distribution

Suppose we have a part of a route r consisting of customers i_1 and i_2 , where the vehicle has load l_1 when traveling from i_1 to i_2 . Furthermore, we denote the difference between the lower bound of the customer's time window and the minimum arrival time $a_{i_2} - M_{i_2}^r$ as d_i^r . We only have to consider the case where $d_i^r \geq 0$. If the difference between the minimum travel time and the lower time window value is smaller than 0, we know that the maximization between the distribution of the delay and the time window always is the value of the delay distribution as the delay is assumed strictly positive. Because waiting is allowed we have that

$$A_{i_2}^r = M_{i_2}^r + \max(d_i^r, S)$$

with

$$S \sim \text{Gamma}(\alpha_S, \lambda) \text{ where } \alpha_S = \alpha_{A_{i_1}^r} + \alpha_{i_1 i_2}^{l_1}$$

For a gamma distributed value $X \sim \text{Gamma}(\alpha, \lambda)$ the following equations hold

$$E[X] = \frac{\alpha}{\lambda}$$

$$\text{Var}(X) = \frac{\alpha}{\lambda^2}$$

Thus if we can estimate the mean and variance of the distribution of the maximum of a gamma distributed value and a constant, we can approximate this distribution using a Gamma distribution. In a similar approach to [29], we write the expected value of the maximization $Y = \max(d_i^r, S)$ as

$$E[Y] = d_i^r P_{d_i^r} + \int_{d_i^r}^{\infty} x f(x) dx$$

Where $f(x)$ is the probability density function of S

$$f(x) = \frac{x^{\alpha_S - 1} e^{-\lambda x} \lambda^{\alpha_S}}{\Gamma(\alpha_S)}$$

And $P_{d_i^r}$ is the probability that the value of S is lower than d_i^r , which can be calculated using the cumulative distribution of S

$$P_{d_i^r} = F_S(d_i^r) = \frac{\gamma(\alpha_S, \lambda d_i^r)}{\Gamma(\alpha_S)}$$

Using these results we can write the expected value as

$$E[Y] = d_i^r \frac{\gamma(\alpha_S, \lambda d_i^r)}{\Gamma(\alpha_S)} + \int_{d_i^r}^{\infty} x \frac{x^{\alpha_S-1} e^{-\lambda x} \lambda^{\alpha_S}}{\Gamma(\alpha_S)} dx \quad (26)$$

Which can also be written as

$$E[Y] = d_i^r \frac{\gamma(\alpha_S, \lambda d_i^r)}{\Gamma(\alpha_S)} + \frac{\lambda^{\alpha_S}}{\Gamma(\alpha_S)} \int_{d_i^r}^{\infty} x^{\alpha_S} e^{-\lambda x} dx$$

We need to integrate $\int x^{\alpha_S} e^{-\lambda x}$ to calculate the result of this value. To calculate this we first apply the substitution rule

$$\int_a^b f(\varphi(x))\varphi'(x)dx = \int_{\varphi(a)}^{\varphi(b)} f(u)du$$

Now as substituting $u = \lambda x$ gives that $dx = \frac{1}{\lambda} du$. This also gives that $x^{\alpha_S} = (\frac{1}{\lambda})^{\alpha_S} u^{\alpha_S}$

$$\int_{d_i^r}^{\infty} x^{\alpha_S} e^{-\lambda x} dx = \lambda^{-\alpha_S-1} \int_{d_i^r \lambda}^{\infty} u^{\alpha_S} e^{-u} du \quad (27)$$

We have that the upper incomplete gamma function $\Gamma(\alpha, x) = \int_x^{\infty} t^{\alpha-1} e^{-t} dt$. We can see that the integral in equation 27 is in the same form as the upper incomplete gamma function. By taking $s = \alpha_S + 1$ and $t = u$ we get that

$$\begin{aligned} E[Y] &= d_i^r \frac{\gamma(\alpha_S, \lambda d_i^r)}{\Gamma(\alpha_S)} + \frac{\lambda^{\alpha_S}}{\Gamma(\alpha_S)} \int_{d_i^r}^{\infty} x^{\alpha_S} e^{-\lambda x} dx \\ &= d_i^r \frac{\gamma(\alpha_S, \lambda d_i^r)}{\Gamma(\alpha_S)} + \frac{\lambda^{\alpha_S}}{\Gamma(\alpha_S)} \lambda^{-\alpha_S-1} \int_{d_i^r \lambda}^{\infty} u^{\alpha_S} e^{-u} du \\ &= d_i^r \frac{\gamma(\alpha_S, \lambda d_i^r)}{\Gamma(\alpha_S)} + \frac{\lambda^{\alpha_S}}{\Gamma(\alpha_S)} \lambda^{-\alpha_S-1} \Gamma(\alpha_S + 1, d_i^r \lambda) \\ &= d_i^r \frac{\gamma(\alpha_S, \lambda d_i^r)}{\Gamma(\alpha_S)} + \frac{\Gamma(\alpha_S + 1, d_i^r \lambda)}{\lambda \Gamma(\alpha_S)} \end{aligned}$$

To create the estimation we also need to know the variance. We know that

$$Var(Y) = E[Y^2] - E[Y]^2$$

And thus we also need to calculate $E[Y^2]$. This can be calculated by squaring the values in equation (26)

$$E[Y^2] = (d_i^r)^2 \frac{\gamma(\alpha_S, \lambda c)}{\Gamma(\alpha_S)} + \int_{d_i^r}^{\infty} x^2 \frac{x^{\alpha_S-1} e^{-\lambda x} \lambda^{\alpha_S}}{\Gamma(\alpha_S)} dx$$

We rewrite this similarly as before as

$$E[Y^2] = (d_i^r)^2 \frac{\gamma(\alpha_S, \lambda d_i^r)}{\Gamma(\alpha_S)} + \frac{\lambda^{\alpha_S}}{\Gamma(\alpha_S)} \int_{d_i^r}^{\infty} x^{\alpha_S+1} e^{-\lambda x} dx$$

Using the same substitution $u = \lambda x$ as earlier and then writing it as the upper incomplete gamma function we can then rewrite this to

$$\begin{aligned}
E[Y^2] &= (d_i^r)^2 \frac{\gamma(\alpha_S, \lambda d_i^r)}{\Gamma(\alpha_S)} + \int_{d_i^r}^{\infty} x^2 \frac{x^{\alpha_S-1} e^{-\lambda x} \lambda^{\alpha_S}}{\Gamma(\alpha_S)} dx \\
&= (d_i^r)^2 \frac{\gamma(\alpha_S, \lambda d_i^r)}{\Gamma(\alpha_S)} + \frac{\lambda^{\alpha_S}}{\Gamma(\alpha_S)} \int_{d_i^r}^{\infty} x^{\alpha_S+1} e^{-\lambda x} dx \\
&= (d_i^r)^2 \frac{\gamma(\alpha_S, \lambda d_i^r)}{\Gamma(\alpha_S)} + \frac{\lambda^{\alpha_S}}{\Gamma(\alpha_S)} \lambda^{-\alpha_S-2} \int_{d_i^r \lambda}^{\infty} u^{\alpha_S+1} e^{-u} du \\
&= (d_i^r)^2 \frac{\gamma(\alpha_S, \lambda d_i^r)}{\Gamma(\alpha_S)} + \frac{\Gamma(\alpha_S + 2, d_i^r \lambda)}{\lambda^2 \Gamma(\alpha_S)}
\end{aligned}$$

Using these results for the $E[Y]$ and $Var(Y)$ we can finally calculate the resulting parameters for the Gamma approximation of Y , $Z \sim Gamma(\alpha_Z, \lambda_Z)$ as follows

$$\begin{aligned}
\alpha_Z &= \frac{E[Y]^2}{Var(Y)} \\
\lambda_Z &= \frac{E[Y]}{Var(Y)}
\end{aligned}$$

Now that we have found the shape and rate parameters for the approximation of Y , we can calculate the probability of arriving late using the cumulative distribution as discussed in the previous section with $\alpha_{A_i^r} = \alpha_Z$ and $\lambda_{A_i^r} = \lambda_Z$. As we allow waiting in this case we no longer need to calculate the probability of being early. Calculating the probability of being late can be done as follows

$$P(L_i^r) = 1 - F(max(b_i - M_i^r, 0); \alpha_{A_i^r}, \lambda_{A_i^r})$$

One issue with this approximation is that we cannot ensure that Z has a constant rate parameter. This makes the addition of the different gamma variables quite a bit more complicated when we need to calculate the arrival time at the next customer. At this next customer, we need to add the gamma-distributed delay to the current delay Z , which can not directly be done as the rate parameters are not necessarily equal.

Suppose we have two gamma-distributed values $X \sim Gamma(\alpha_x, \lambda_x)$ and $Y \sim Gamma(\alpha_y, \lambda_y)$, these can not be added together as before, as λ_x does not necessarily equal λ_y . This can, however, be solved with another approximation of the addition of different gamma-distributed values. We use the moment-matching method or MMM approximation as detailed by Stewart et al. [12]. This results in the following approximation

$$\begin{aligned}
\alpha_{MMM} &= \frac{\mu^2}{\sigma^2} \\
\lambda_{MMM} &= \frac{\mu}{\sigma^2}
\end{aligned}$$

Where

$$\begin{aligned}
\mu &= \alpha_x \frac{1}{\lambda_x} + \alpha_y \frac{1}{\lambda_y} \\
\sigma^2 &= \alpha_x \frac{1}{\lambda_x^2} + \alpha_y \frac{1}{\lambda_y^2}
\end{aligned}$$

This proved to be quite accurate according to [12].

Sadly this approximation of the maximum of a gamma-distributed value and a constant can not be implemented in most programming languages, such as C#, without major modifications. It requires very high precision in the intermediate steps of the calculation which can not be provided by the datatypes available in C#. We can however approximate this by scaling the distribution such that the value of the shape parameter is limited. Suppose we have $X \sim \text{Gamma}(\alpha, \lambda)$. We can create a Gamma distribution with equal mean, but an x times smaller shape parameter with $Y \sim \text{Gamma}(\frac{\alpha}{x}, \lambda \cdot x)$. However, Y also has a x times larger variance, losing accuracy in the approximation.

Although this implementation does not produce any errors it is very slow to estimate the values of these gamma distribution values and therefore not feasible in the local search algorithm. During testing, the implementation of calculating the probabilities from the actual distributions when waiting slowed the local search down by a factor of 10^5 . Therefore, we needed to approach this differently.

B.4 Normal approximation

To tackle the problems of precision and performance we use a normal approximation. For a gamma distribution $Gamma(\alpha, \lambda)$, the mean can be calculated by $\frac{\alpha}{\lambda}$, and the variance can be calculated by $\frac{\alpha}{\lambda^2}$. Using this we can approximate this distribution as $N(\frac{\alpha}{\lambda}, \frac{\alpha}{\lambda^2})$.

One problem, however, with the normal distribution is that it is not strictly positive. This does not cohere with our assumptions about the delay. Thus we will approximate the $Gamma(\alpha, \lambda)$ using a conditional normal approximation. Furthermore, we also approximate the gamma delay by a conditional $N^*(\frac{\alpha}{\lambda}, \frac{\alpha}{\lambda^2})$, which is a normally distributed value, but given that it is greater or equal to 0. One problem is that calculations such as addition are not trivial using this distribution. To tackle this, we approximate the N^* distribution using a normal distribution. First, if we have some value $s \sim N(\mu, \sigma^2)$, and $x = \max(s, 0)$ then $x \sim N^*(\mu, \sigma^2)$. We will approximate $\max(N(\frac{\alpha}{\lambda}, \frac{\alpha}{\lambda^2}), 0)$ with a normal distribution. How to approximate such a maximization between a constant and a normal distribution will be discussed in Section B.4.2. The calculations in the following two subsections apply to both our approximations and are based on the results of Van Lent[29].

B.4.1 Calculating arrival times without waiting

Similar to the Gamma distribution, we start with the case where waiting for the start of the time window of a customer is not allowed. This makes the calculations a bit easier because normal distributions can be added together to form a new normal distribution, just like with the Gamma distribution. For the normal case, the travel times also exist of two components. A deterministic component and a stochastic component. Each edge between customers has a deterministic travel time and normal approximation of the actual gamma distribution describing the delay along that edge defined for each load level. We define the variables

- t_{ij}^l as the deterministic travel time required to travel from customer i to j with load level l
- μ_{lij} as the mean of the normal distribution approximation of the delay when traveling from customer i to j with load level l
- σ_{lij}^2 as the variance of the normal distribution approximation of the delay when traveling from customer i to j with load level l

With these variables we can now define the travel time from customer i to j with load level l as

$$T_{ij}^l = t_{ij}^l + N(\mu_{lij}, \sigma_{lij}^2)$$

Suppose we have a part of a route r consisting of customers i_1, i_2 and i_3 , where the vehicle has load l_1 when traveling from i_1 to i_2 and load l_2 when traveling from i_2 to i_3 . Using the previous definitions for the travel time between two customers and the variables defined for the Gamma calculation we can define the arrival time at customer i_3 in route r with this normal approximation

$$\begin{aligned} A_{i_3}^r &= A_{i_2}^r + s_{i_2} + T_{i_2 i_3}^{l_2} \\ &= A_{i_1}^r + s_{i_1} + s_{i_2} + T_{i_1 i_2}^{l_1} + T_{i_2 i_3}^{l_2} \\ &= A_{i_1}^r + s_{i_1} + s_{i_2} + t_{i_1 i_2}^{l_1} + t_{i_2 i_3}^{l_2} + N(\mu_{l_1 i_1 i_2}, \sigma_{l_1 i_1 i_2}^2) + N(\mu_{l_2 i_2 i_3}, \sigma_{l_2 i_2 i_3}^2) \end{aligned}$$

From these equations, we can again see the two components emerge. The deterministic component consists of the sum of service times and deterministic travel times while the stochastic component consists of the sum of normally distributed delays. The sum of two independent normal distributions $X \sim N(\mu_1, \sigma_1^2)$ and $Y \sim N(\mu_2, \sigma_2^2)$, $Z = X + Y$ has the following distribution

$$Z \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

And thus we can denote the arrival time at customer i in route r , A_i^r as

$$A_{i_3}^r = A_{i_1}^r + s_{i_1} + s_{i_2} + t_{i_1 i_2}^{l_1} + t_{i_2 i_3}^{l_2} + N(\mu_{l_1 i_1 i_2} + \mu_{l_2 i_2 i_3}, \sigma_{l_1 i_1 i_2}^2 + \sigma_{l_2 i_2 i_3}^2)$$

Now, combining this into a sum of deterministic values and a sum of normally distributed values we can calculate the probabilities of being on time at some customer. We denote

- $\mu_{A_i^r}$ as the mean of the stochastic component of the arrival time at customer i in route r
- $\sigma_{A_i^r}^2$ as the variance of the stochastic component of the arrival time at customer i in route r
- M_i^r as the sum of deterministic travel times and service times. This is the minimum arrival time possible.

Using these definitions we can define A_i^r as

$$A_i^r = M_i^r + N(\mu_{A_i^r}, \sigma_{A_i^r}^2)$$

Before we can use these values we need the probability density and the cumulative distribution functions. The probability density function for a normal distribution with mean μ and variance σ^2 is given by

$$f(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The cumulative distribution function is given by

$$\begin{aligned} F(x; \mu, \sigma^2) &= \int_{-\infty}^x f(t; \mu, \sigma^2) dt \\ &= \phi\left(\frac{x-\mu}{\sigma}\right) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right] \end{aligned}$$

Where $\operatorname{erf}(x)$ is the so called error function.

Using this definition for the cumulative distribution we can calculate the probabilities of arriving early or late at customer i in route r similarly to [29]. However, we do have one difference, and that is that we assumed the delay to be strictly positive, which is not necessarily the case with a normal distribution. Thus, if $M_i^r \geq a_i$, $P(E_i^r) = 0$. Consequently giving the following calculations

$$P(E_i^r) = \begin{cases} 0 & \text{if } M_i^r > a_i \\ P(A_i^r < a_i | A_i^r \geq M_i^r) & \text{otherwise} \end{cases}$$

Thus for the case where $M_i^r \leq a_i$ we get that

$$\begin{aligned} P(E_i^r) &= P(A_i^r < a_i | A_i^r \geq M_i^r) \\ &= P(A_i^r - M_i^r < a_i - M_i^r | A_i^r - M_i^r \geq 0) \\ &= \frac{P(A_i^r - M_i^r < a_i - M_i^r \wedge A_i^r - M_i^r \geq 0)}{P(A_i^r - M_i^r \geq 0)} \\ &= \frac{F(a_i - M_i^r; \mu_{A_i^r}, \sigma_{A_i^r}^2) - F(0; \mu_{A_i^r}, \sigma_{A_i^r}^2)}{1 - F(0; \mu_{A_i^r}, \sigma_{A_i^r}^2)} \end{aligned}$$

Thus we can conclude

$$P(E_i^r) = \begin{cases} 0 & \text{if } M_i^r > a_i \\ \frac{F(a_i - M_i^r; \mu_{A_i^r}, \sigma_{A_i^r}^2) - F(0; \mu_{A_i^r}, \sigma_{A_i^r}^2)}{1 - F(0; \mu_{A_i^r}, \sigma_{A_i^r}^2)} & \text{otherwise} \end{cases}$$

We can calculate the probability of being late similarly

$$P(L_i^r) = \begin{cases} 1 & \text{if } M_i^r > b_i \\ P(A_i^r > b_i | A_i^r \geq M_i^r) & \text{otherwise} \end{cases}$$

Thus for the case where $M_i^r \leq b_i$ we get that

$$\begin{aligned} P(L_i^r) &= P(A_i^r - M_i^r > b_i - M_i^r | A_i^r - M_i^r \geq 0) \\ &= \frac{P(A_i^r - M_i^r > b_i - M_i^r \wedge A_i^r - M_i^r \geq 0)}{P(A_i^r - M_i^r \geq 0)} \\ &\text{As } b_i - M_i^r \geq 0 \text{ we get} \\ &= \frac{P(A_i^r - M_i^r > b_i - M_i^r)}{P(A_i^r - M_i^r \geq 0)} \\ &= \frac{1 - P(A_i^r - M_i^r \leq b_i - M_i^r)}{P(A_i^r - M_i^r \geq 0)} \\ &= \frac{1 - F(b_i - M_i^r; \mu_{A_i^r}, \sigma_{A_i^r}^2)}{1 - F(0; \mu_{A_i^r}, \sigma_{A_i^r}^2)} \end{aligned}$$

And thus we can conclude

$$P(L_i^r) = \begin{cases} 1 & \text{if } M_i^r > b_i \\ \frac{1 - F(b_i - M_i^r; \mu_{A_i^r}, \sigma_{A_i^r}^2)}{1 - F(0; \mu_{A_i^r}, \sigma_{A_i^r}^2)} & \text{otherwise} \end{cases}$$

B.4.2 Calculating arrival times with waiting

Just like with gamma distributions, the maximum between a normal distribution and a constant no longer is normally distributed. But to make it easier to analyze the arrival times later on in the route we want to approximate it with a normal distribution. In this section, we discuss how we can formulate such an approximate distribution.

This approximation is based on the approximation of the maximum of two independent normal distributions as detailed by Nadarajah and Kotz[13]. Ehmke et al.[22] tested the accuracy of approximating a maximum of a constant and a normally distributed value with a normally distributed value, and came to the conclusion that this approximation is indeed an accurate representation of the maximum. Van Lent[29] also used this approximation in her thesis. The calculations in this section are based on her findings. Figure 3 shows an example of the case where waiting is allowed at some customer i in route r . We need to calculate the distribution of the maximum of a_i and the normal distribution shown in this figure.

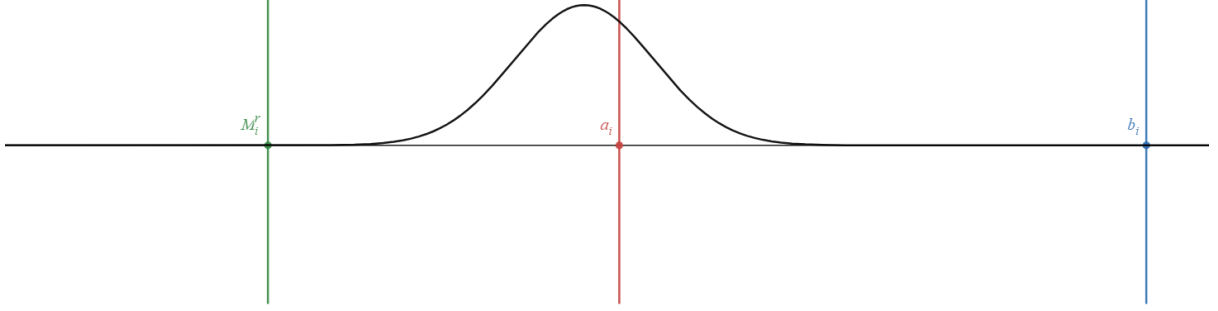


Figure 3: Example of the case where waiting is allowed with the Normal distribution

As we are going to calculate the maximum between two normal distributions as tested by Emke et al.[22] we treat the difference between the minimum travel time and the lower time window value of customer i in route r , $d_i^r = a_i - M_i^r$ as a normally distributed value B_i^r with a variance of zero

$$B_i^r \sim N(d_i^r, 0)$$

We only have to consider the case where $d_i^r \geq 0$. If the difference between the minimum travel time and the lower time window value is smaller than 0, we know that the maximization between the distribution of the delay and the time window always is the value of the delay distribution as the delay is assumed strictly positive.

Suppose we have a part of a route r consisting of customers i_1 and i_2 , where the vehicle has load l_1 when traveling from i_1 to i_2 . Because waiting is allowed we have that

$$A_{i_2}^r = M_{i_2}^r + \max(B_{i_2}^r, S)$$

with

$$S_i^r \sim N(\mu_S, \sigma_S^2) \text{ where } \mu_S = \mu_{A_{i_1}^r} + \mu_{l_1 i_1 i_2} \text{ and } \sigma_S^2 = \sigma_{A_{i_1}^r}^2 + \sigma_{l_1 i_1 i_2}^2$$

As S and B_i^r are not correlated we can calculate the the expected value and variance using the method of Nadarajah and Kotz[13]. They state that if we have two normally distributed values $X_1 \sim N(\mu_1, \sigma_1^2)$ and $X_2 \sim N(\mu_2, \sigma_2^2)$, the maximisation $X = \max(X_1, X_2)$ can be calculated using the following formulas

$$E[X] = \mu_1 \Phi\left(\frac{\mu_1 - \mu_2}{\theta}\right) + \mu_2 \Phi\left(\frac{\mu_2 - \mu_1}{\theta}\right) + \theta \phi\left(\frac{\mu_1 - \mu_2}{\theta}\right)$$

$$E[X^2] = (\sigma_1^2 + \mu_1^2) \Phi\left(\frac{\mu_1 - \mu_2}{\theta}\right) + (\sigma_2^2 + \mu_2^2) \Phi\left(\frac{\mu_2 - \mu_1}{\theta}\right) + (\mu_1 + \mu_2) \theta \phi\left(\frac{\mu_1 - \mu_2}{\theta}\right)$$

Here $\theta = \sqrt{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2}$, and Φ and ϕ are the cumulative density function and probability density function of the standard normal distribution respectively.

Van Lent [29] used these formulas by Nadarajah and Kotz[13] to calculate the maximization between the constant and a normal distribution. We adapted her implementation as follows

$$E[As_i^r] = \mu_S \Phi\left(\frac{\mu_S - d_i^r}{\sigma_S}\right) + d_i^r \Phi\left(\frac{d_i^r - \mu_S}{\sigma_S}\right) + \sigma_S \phi\left(\frac{\mu_S - d_i^r}{\sigma_S}\right)$$

$$E[As_i^{r^2}] = (\sigma_S^2 + \mu_S^2) \Phi\left(\frac{\mu_S - d_i^r}{\sigma_S}\right) + d_i^{r^2} \Phi\left(\frac{d_i^r - \mu_S}{\sigma_S}\right) + (\mu_S + d_i^r) \sigma_S \phi\left(\frac{\mu_S - d_i^r}{\sigma_S}\right)$$

Here As_i^r is the stochastic component of the arrival time at customer i in route r . Using these two results we can calculate the variance of the stochastic component of the arrival time at customer i in route r by applying the general rule for variance

$$\sigma^2(As_i^r) = E[As_i^{r2}] - E[As_i^r]^2$$

Now that we have found the expected value and the variance of the new normal distribution we can calculate the probability of arriving late using the cumulative distribution as discussed in the previous section with $\mu_{A_i^r} = E[As_i^r]$ and $\sigma_{A_i^r}^2 = \sigma^2(As_i^r)$. As we allow waiting in this case we no longer need to calculate the probability of being early. Calculating the probability of being late can be done as follows

$$P(L_i^r) = \begin{cases} 1 & \text{if } M_i^r > b_i \\ \frac{1-F(b_i-M_i^r; \mu_{A_i^r}, \sigma_{A_i^r}^2)}{1-F(0; \mu_{A_i^r}, \sigma_{A_i^r}^2)} & \text{otherwise} \end{cases}$$

B.5 Experiments

To test the effects of including the stochastic delay in the travel times we conducted multiple experiments. These experiments were conducted using the parameter configurations explained in Section 5.2.1. However, after preliminary testing, a small change to the move neighborhood (1) was made. During preliminary testing, it became apparent that when the move operator is allowed to move a customer to a newly created empty route it would create too many routes. Without this possibility the found solution has less than 10 routes, while with it enabled it found solutions with more than 30 routes, achieving a score a lot worse than without. Therefore, the decision was made to disable the possibility for the move neighborhood to create new routes. For the experiments where waiting was not allowed the earliness penalty c_e and the lateness penalty c_l were set at 10. Furthermore, the start times were optimized deterministically as explained in Appendix C as optimizing including the distributions was outside the scope for this Thesis and without it, performance was reduced in both the on-time percentage and route length. For the experiments where waiting was allowed, the earliness penalty was set at 0, while the lateness penalty remained at 10. Finally, the last change made was that the number of restarts was reduced to 3 instead of 7 in the original algorithm in the interest of time.

Each experiment was repeated 5 times, similar to the experiments for the basic VRPLTT. The experiments for the VRPSLTT were performed on the same instances as defined by Fontaine[33]. However, in the interest of time, it was only tested on the largest instances of 100 customers.

B.5.1 Distribution creation

As we have no data about travel time distributions on these instances we had to create these delay distributions. These delay distributions are dependent on both the travel time and the weight of the bicycle. We assume the true distribution to be a gamma distribution with shape parameter α and rate parameter λ where

$$\alpha = 1.5 + \frac{t}{2} + \frac{w}{290}$$
$$\lambda = 10$$

Where t denotes the travel time of the vehicle for that leg of the journey and w is the weight of the vehicle for that leg of the journey. These values for the parameters were chosen as they created logical results. As an example, the results for the mean and mode of these distributions were calculated for 10 load levels. The travel times were between 0 and 12 minutes and the mean of the resulting delay varied between ± 10 to ± 50 seconds, while the mode varied between ± 5 and ± 45 seconds.

B.5.2 Methodology

For each instance, the following experiments were performed.

1. No waiting allowed, but planning without using probabilities ($c_e = c_l = 0$ in the objective function), only planning using the mean of the true distribution on top of the travel time. This allows comparing just deterministic planning to planning using the distributions. In this setup, it does not allow any deterministic earliness or lateness in the final solution.
2. No waiting allowed, planning using the true gamma distributions
3. No waiting allowed, planning using the normal approximation
4. No waiting allowed, planning using the N^* approximation
5. Waiting allowed, planning deterministically using just the mean of the distributions, similarly to the case where waiting is not allowed. In this setup, it does not allow any deterministic lateness in the final solution

6. Waiting allowed, planning using the normal approximation
7. Waiting allowed, planning using the N^* approximation
8. Waiting allowed, but planning using the true gamma distribution, while ignoring the waiting during the addition of the distributions, skipping the maximization between a constant and a distribution. This essentially plans deterministically (i.e. when M_i^r is smaller than a_i , the deterministic arrival time gets adjusted to a_i), but takes the delay into account on top of this. This implementation is, therefore, a bit more pessimistic, as the same delay is counted but in this case, the deterministic early arrival is also counted. It was included to test the benefits of properly incorporating the addition during planning.

After the algorithm finished execution, the found solutions were simulated 1.000.000 times to retrieve the actual performance of that solution. For each solution, the following values were recorded

- The average on-time percentage or OTP. We define OTP as the percentage of customer visits that are on time. Suppose we have a solution with 100 customers, which is run one time. This would have 100 customer visits. If we arrive late at a single customer the OTP would be $\frac{99}{100} = 0.99$. Now suppose we run this solution 100 times. We, therefore, have 10000 customer visits. If during the simulation the vehicle does not arrive on time at a customer once we would have an OTP of $\frac{9999}{10000} = 0.9999$.
- The worst average OTP for each route in the solution. The OTP is also recorded for each customer and the OTP value of the customer with the worst OTP in each route is reported. This makes it possible to analyze the weakest links in the routes.
- The average route length
- Predicted OTP during planning, which is calculated using the probability of being on time.
- Predicted worst OTP during planning, which is calculated using the probability of being on time.

B.6 Results

The results for the experiments described in the previous section are shown in Tables 11-17. These show the score the algorithm achieved and the ILP improvement of the score. This score is not just the travel time as it includes the penalty for being early or late and does not include the impact on the travel time by the delay. Additionally, the average travel time, OTP, and worst OTP of the simulations are given. Lastly, the expected OTP and worst OTP calculated using the distributions during planning are given.

B.6.1 Without waiting

First, we will discuss the results when waiting for the start of a customer's time window is not allowed. One of the first things one may notice from Tables 11-14 is that the ILP improves the solution a lot more compared to the results of the basic VRPLTT. Interestingly even when just the mean of the distribution is used the ILP phase has a lot more influence as shown in Table 11. This might be caused by the extra travel time making it harder for the local search to find the correct combination of routes for the optimal solution.

Table 11 shows that planning deterministically provides poor results in terms of OTP and worst OTP. For each instance, less than 60% of the customer visits were on time on average, while the worst OTP was less than 20% on average. The worst results were found for the Sydney instance where the average OTP was just 39% and on average, the routes in the solution had customers with an OTP of less than 4% in our simulations.

Tables 12, 13, and 14 show that including the distributions in the planning increases the OTP massively. Both planning using the true distribution, and planning using the normal approximations achieve

an average OTP of over 99.97% on all instances, while the average worst OTP from the simulations is over 99.5%. When we look at the average route length or travel time, the solutions found by planning deterministically are slightly shorter. Because it does not take any probabilities into account, it plans more tightly compared to planning with the distributions, resulting in slightly shorter routes with a lower OTP. It seems that we can improve the route length at the cost of the OTP.

Comparing the true distribution in Table 12 to the approximations in Tables 13 and 14 we can see that they produce comparable results in terms of simulated route length and OTP, but slight differences can be seen in the simulated OTP and worst OTP in favor of the true distribution. Larger differences can be found in the average expected OTP and worst OTP. While the true distribution's expected OTP and worst OTP are very close to the actual OTP, the expected OTP of the normal approximations seems more pessimistic. It seems to underestimate the actual OTP. However, this does not seem to cost it too much because, as stated earlier, the results are very similar.

Both normal approximations produce comparable results in the final simulations. No clear winner can be found for the simulated Length, OTP, or worst OTP. However, the expected OTP and worst OTP of the conditional N^* approximation seem to be slightly closer to reality than the conditional normal approximation. It produces expected OTP and worst OTP values that are more accurate to the simulations on all instances tested.

Table 11: Results for experiment 1; No waiting allowed, planning deterministically using the mean

C	network	AVG		AVG Simulated			AVG Expected		ILP time (s)	LS time (s)
		score	ILP imp (%)	Length (min)	OTP (%)	Worst OTP (%)	OTP (%)	Worst OTP (%)		
100	Fukuoka	149.20	11.41	149.20	59.4519	17.6799	-	-	3600.38	570.45
100	Madrid	154.73	11.22	154.73	57.4519	14.3884	-	-	22.90	518.21
100	Pittsburgh	162.97	10.46	162.97	58.9751	19.9727	-	-	28.34	608.56
100	Seattle	128.58	13.06	128.58	52.2005	8.9076	-	-	1586.02	645.79
100	Sydney	180.69	9.80	180.69	38.9566	4.0667	-	-	177.33	457.09

Table 12: Results for experiment 2; No waiting allowed, planning using the true gamma distribution

C	network	AVG		AVG Simulated			AVG Expected		ILP time (s)	LS time (s)
		score	ILP imp (%)	Length (min)	OTP (%)	Worst OTP (%)	OTP (%)	Worst OTP (%)		
100	Fukuoka	122.34	7.30	152.32	99.9985	99.9792	99.9984	99.9796	2047.57	527.72
100	Madrid	128.30	3.38	158.39	99.9850	99.8106	99.9835	99.8102	10.97	526.66
100	Pittsburgh	138.45	1.59	169.23	99.9950	99.9321	99.9947	99.9326	95.38	594.06
100	Seattle	104.31	1.93	133.12	99.9750	99.6284	99.9732	99.6275	139.72	535.32
100	Sydney	152.70	2.79	184.16	99.9950	99.9260	99.9946	99.9259	123.56	554.32

Table 13: Results for experiment 3; No waiting allowed, planning using the normal approximation

C	network	AVG		AVG Simulated			AVG Expected		ILP time (s)	LS time (s)
		score	ILP imp (%)	Length (min)	OTP (%)	Worst OTP (%)	OTP (%)	Worst OTP (%)		
100	Fukuoka	122.15	7.32	151.97	99.9798	99.6968	99.7730	98.1703	2053.62	433.95
100	Madrid	128.35	3.38	158.56	99.9858	99.8394	99.7035	97.7970	14.18	429.37
100	Pittsburgh	138.58	1.09	169.36	99.9876	99.8305	99.1907	97.3908	95.92	474.90
100	Seattle	104.20	1.80	133.12	99.9726	99.6101	99.0542	96.9620	159.22	441.91
100	Sydney	152.84	3.16	184.31	99.9869	99.8081	99.8154	98.4413	135.28	440.52

Table 14: Results for experiment 4; No waiting allowed, planning using the N^* approximation

C	network	AVG		AVG Simulated			AVG Expected		ILP time (s)	LS time (s)
		score	ILP imp (%)	Length (min)	OTP (%)	Worst OTP (%)	OTP (%)	Worst OTP (%)		
100	Fukuoka	122.13	6.86	151.94	99.9767	99.6444	99.7942	98.3387	1885.68	497.55
100	Madrid	128.15	3.35	158.19	99.9701	99.5877	99.7462	98.0417	9.94	498.02
100	Pittsburgh	138.51	1.72	169.36	99.9924	99.8958	99.3634	97.8674	94.04	554.51
100	Seattle	104.22	1.62	133.20	99.9736	99.6220	99.2352	97.4244	137.94	492.35
100	Sydney	152.85	2.82	184.35	99.9917	99.8806	99.8852	98.6278	153.46	510.57

B.6.2 With waiting

Secondly, we will discuss the results when waiting for the start of a customer’s time window is allowed. When we look at the results when planning deterministically using just the mean of the distribution in Table 15 we see a similar story compared to without waiting. It only achieves an average simulated OTP of less than 60%. One big difference with the case where waiting is allowed is the worst OTP. The worst OTP is below 3% for all instances, while 4 out of 5 score an average worst OTP below 0.05%. Because waiting is allowed there is more freedom in planning, which apparently allows it to create an even tighter planning, resulting in worse worst OTP. However, Table 15 also shows that the ILP improves the solution by quite a lot, similar to the case without waiting.

Now when we compare those results to the results found by considering the distributions during planning we can see that that improves the result in terms of OTP and worst OTP by a lot. All approximations score an average simulated OTP of over 99.96% and an average simulated worst OTP of at least 99.4%. However, this time the difference in average simulated route length with the deterministic planning is a lot higher compared to without waiting. It seems that when waiting is allowed one can win more in terms of route length by dropping the penalty for being late.

When we compare the normal approximations with proper addition in Tables 16 and 17 to the true distribution which ignores addition in Table 18 a few differences can be noticed. Firstly, the gamma distribution that ignores addition actually provides a better OTP and worst OTP. This is logical as it is more conservative as it plans the routes deterministically for the early arrival. This does come at the cost of longer routes on average. This shows that actually planning using the maximization of the distribution and the lower bound of a customer’s time window does result in better results in terms of route length. Interestingly, the ILP improvement is a lot smaller for these cases compared to the cases where waiting is not allowed. Lastly, comparing the expected OTP and worst OTP in Table 18 to the simulated results we can see the slightly pessimistic planning explained earlier as it underestimates its actual results.

Finally, comparing the two normal approximations, again no clear difference in the simulated results can be found, with varying results over the instances. However, there are slightly more major differences for some instances. For example, on the Fukuoka instance, the conditional normal approximation achieves higher OTP and worst OTP results compared to the conditional N^* approximation, with only a minimal increase in route length. This difference is, however, not consistent over the instances. Looking at the differences in expected OTP and worst OTP we no longer see the consistent difference between the two approximations that was seen when waiting was not allowed. Both slightly overestimate their OTP and worst OTP.

Table 15: Results for experiment 5; Waiting allowed, planning deterministically using the mean

C	network	AVG		AVG Simulated			AVG Expected		ILP time (s)	LS time (s)
		score	ILP imp (%)	Length (min)	OTP (%)	Worst OTP (%)	OTP (%)	Worst OTP (%)		
100	Fukuoka	140.90	13.10	140.90	54.4366	2.8579	-	-	3025.53	756.13
100	Madrid	145.54	11.78	145.54	50.3192	0.0000	-	-	19.37	525.13
100	Pittsburgh	158.09	10.85	158.09	53.9693	0.0407	-	-	3367.58	768.81
100	Seattle	121.56	13.71	121.56	52.8429	0.0000	-	-	2520.43	689.91
100	Sydney	170.97	10.32	170.97	57.2547	0.0000	-	-	165.76	474.15

Table 16: Results for experiment 6; Waiting allowed, planning using the normal approximation

C	network	AVG		AVG Simulated			AVG Expected		ILP time (s)	LS time (s)
		score	ILP imp (%)	Length (min)	OTP (%)	Worst OTP (%)	OTP (%)	Worst OTP (%)		
100	Fukuoka	119.75	0.09	149.64	99.9994	99.9899	99.9998	99.9973	3600.69	731.04
100	Madrid	126.97	0.28	157.06	99.9605	99.5940	99.9873	99.8890	25.48	676.76
100	Pittsburgh	137.02	0.13	167.68	99.9923	99.8842	99.9976	99.9656	151.01	807.93
100	Seattle	102.91	0.26	131.63	99.9709	99.5793	99.9870	99.8257	182.39	822.25
100	Sydney	150.69	0.22	182.16	99.9996	99.9948	100.0000	99.9999	257.26	567.19

Table 17: Results for experiment 7; Waiting allowed, planning using the N^* approximation

C	network	AVG		AVG Simulated			AVG Expected		ILP time (s)	LS time (s)
		score	ILP imp (%)	Length (min)	OTP (%)	Worst OTP (%)	OTP (%)	Worst OTP (%)		
100	Fukuoka	119.81	0.01	149.62	99.9858	99.7788	99.9932	99.8987	3600.47	745.27
100	Madrid	127.06	0.19	157.19	99.9713	99.6958	99.9925	99.9329	28.22	708.16
100	Pittsburgh	137.02	0.43	167.64	99.9887	99.8325	99.9966	99.9528	336.42	842.46
100	Seattle	102.88	0.17	131.61	99.9671	99.4854	99.9887	99.8340	154.55	714.78
100	Sydney	150.66	0.20	182.12	99.9998	99.9963	100.0000	99.9999	317.55	509.31

Table 18: Results for experiment 8; Waiting allowed, planning using the gamma distribution ignoring the maximization

C	network	AVG		AVG Simulated			AVG Expected		ILP time (s)	LS time (s)
		score	ILP imp (%)	Length (min)	OTP (%)	Worst OTP (%)	OTP (%)	Worst OTP (%)		
100	Fukuoka	120.16	0.45	150.01	100.0000	99.9998	99.9951	99.9307	3600.62	544.25
100	Madrid	127.75	0.12	157.90	99.9961	99.9407	99.9929	99.9074	16.85	538.67
100	Pittsburgh	137.93	0.19	168.43	99.9975	99.9652	99.9849	99.7984	165.67	674.75
100	Seattle	103.94	0.38	132.87	99.9995	99.9921	99.9967	99.9575	90.17	539.47
100	Sydney	151.87	0.13	183.21	99.9995	99.9927	99.9854	99.7966	185.14	422.47

B.7 Conclusion

In this appendix, we presented how to incorporate stochastic delay into the vehicle routing problem with load-dependent travel times by calculating the probability of arriving early and/or late at customers. These probabilities were incorporated into the objective function to penalize routes according to their probability of not arriving on time. We showed that both the conditional normal approximation and the conditional N^* approximation of the Gamma distributed stochastic delay worked well in our computational experiments. Both approximations and the true distribution provided large improvements in terms of on-time percentages, increasing service levels compared to deterministic planning.

B.7.1 Future research

For the current instances finding a solution with a high OTP was possible and did not cost a lot in terms of route length. Future research could test this with tighter instances which make it more difficult for the algorithm. Furthermore, our model only uses the probabilities of being early or late in the objective function of the model. It would be an interesting addition to add constraints that constrain the OTP for each customer, especially on the earlier mentioned more difficult instances. Finally, more approximations of the gamma distribution, such as the non-conditional N^* approximation, could be tested for their performance.

C Disallowing waiting

In the VRPLTT model waiting at the customer for the start of its time window is allowed and not penalized as in most other VRP models. Furthermore, only travel time is minimized, not total time. However, as the objective function is minimizing travel time it might be realistic to penalize waiting. When early arrival at a customer is not allowed, the start time of the route is a vital part of finding good solutions. Luckily optimizing the start time for a given route is easily solvable in linear time. In this section, the algorithm used for optimizing the start time will be described. Imagine the following route 0, 1, 2, 3, 4, 0, where each number represents the customer id. Each customer i has a lower time window l_i and an upper time window u_i . The travel time from customer i to customer j is denoted by d_{ij} and the service time for a customer i is defined by st_i . Now we want to find a start time s such that

$$\begin{aligned}
 l_1 &\leq s + d_{01} \\
 l_2 &\leq s + d_{01} + st_1 + d_{12} \\
 l_3 &\leq s + d_{01} + st_1 + d_{12} + st_2 + d_{23} \\
 l_4 &\leq s + d_{01} + st_1 + d_{12} + st_2 + d_{23} + st_3 + d_{34} \\
 u_1 &\geq s + d_{01} \\
 u_2 &\geq s + d_{01} + st_1 + d_{12} \\
 u_3 &\geq s + d_{01} + st_1 + d_{12} + st_2 + d_{23} \\
 u_4 &\geq s + d_{01} + st_1 + d_{12} + st_2 + d_{23} + st_3 + d_{34}
 \end{aligned}$$

Rewriting for s gives

$$s \geq l_1 - d_{01} \tag{28}$$

$$s \geq l_2 - (d_{01} + st_1 + d_{12}) \tag{29}$$

$$s \geq l_3 - (d_{01} + st_1 + d_{12} + st_2 + d_{23}) \tag{30}$$

$$s \geq l_4 - (d_{01} + st_1 + d_{12} + st_2 + d_{23} + st_3 + d_{34}) \tag{31}$$

$$s \leq u_1 - d_{01} \tag{32}$$

$$s \leq u_2 - (d_{01} + st_1 + d_{12}) \tag{33}$$

$$s \leq u_3 - (d_{01} + st_1 + d_{12} + st_2 + d_{23}) \tag{34}$$

$$s \leq u_4 - (d_{01} + st_1 + d_{12} + st_2 + d_{23} + st_3 + d_{34}) \tag{35}$$

This can be solved by going over the route in order to store the maximum value of the right-hand side of equations 28-31 and the minimum of the right-hand side of equations 32-35. Essentially giving a minimum (min_s) and maximum (max_s) start times between which all time windows in the route can be met. Finally, the start time is set to the found value for the minimum. If $min_s > max_s$, it is not possible to meet all time windows. In this case, the start time is set to min_s to avoid late arrivals.

This optimization is performed for each different route tried. Even though it is linearly solvable, it still is a significant time investment during the local search, as the weight dependency requires a recalculation for (almost) all changes. However, it does provide a large improvement in the results found by the algorithm when waiting is not allowed. But as it is not necessary and it is a significant time investment, this step is skipped when waiting is allowed.

C.1 Experiments

To test the impact of not allowing waiting on the total travel time of the solution we repeated the same experiments as for the basic VRPLTT and recorded the scores. In the interest of time, the ILP time limit was reduced to 1200 seconds for these experiments.

C.1.1 Results

Table 19 shows the results of the experiments where waiting is not allowed. Furthermore, the average and minimum results for the original VRPLTT where waiting is allowed from Table 2 were added. One of the first things one may notice is that compared to the results in Table 2 the execution times are larger. This is caused by the start time optimization mentioned in the previous section. Further comparing these tables, for some instances, the scores with and without waiting allowed are very similar. For example, the smallest Madrid and Fukuoka instances do not seem to show any difference, while the scores for the largest Pittsburgh instance even are slightly better than without waiting. However, this most likely happened by chance. Other instances such as the largest Fukuoka and Madrid instances show larger differences between the results. However, even for these instances, the difference is below 1.5% showing that allowing waiting before the start of a customer’s time window does give the ability to find shorter routes, but does not provide dramatic gains in time.

Table 19: Results without waiting

C	network	With waiting		AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		AVG	MIN	LS+ILP	LS	imp(%)	LS+ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	73.87	73.87	0.00	73.87	73.87	0.00	270.43	2.86	267.57
50	Madrid	73.97	73.97	73.97	73.97	0.00	73.97	73.97	0.00	216.11	0.47	215.64
50	Pittsburgh	87.07	87.07	87.49	87.49	0.00	87.49	87.49	0.00	237.06	1.76	235.30
50	Seattle	58.93	58.93	59.01	59.01	0.00	59.01	59.01	0.00	238.52	14.32	224.20
50	Sydney	95.34	95.34	96.66	96.66	0.00	96.66	96.66	0.00	248.61	23.67	224.94
75	Fukuoka	95.15	95.15	95.27	95.28	0.01	95.27	95.27	0.00	1443.08	1202.90	240.18
75	Madrid	97.69	97.69	98.01	98.21	0.20	97.70	97.70	0.00	242.56	17.09	225.47
75	Pittsburgh	110.71	110.71	110.75	110.75	0.00	110.75	110.75	0.00	278.57	10.82	267.75
75	Seattle	79.83	79.83	79.92	79.92	0.00	79.92	79.92	0.00	330.43	49.64	280.79
75	Sydney	122.51	122.39	123.58	124.27	0.56	123.58	124.05	0.38	224.57	30.23	194.34
100	Fukuoka	118.87	118.87	120.62	120.68	0.04	119.42	119.63	0.18	1452.04	1201.18	250.86
100	Madrid	124.96	124.80	126.78	126.93	0.12	126.67	126.77	0.08	292.64	66.28	226.36
100	Pittsburgh	134.32	133.66	134.31	134.99	0.50	133.66	133.74	0.06	345.04	91.11	253.93
100	Seattle	101.32	101.32	101.46	102.02	0.54	101.46	101.97	0.50	995.89	742.69	253.20
100	Sydney	149.65	149.46	151.20	151.35	0.10	150.85	150.85	0.00	1173.33	962.46	210.87

D Comparison on VRPTW instances

The hybrid algorithm was also tested on traditional VRP instances created by Solomon[4]. Many algorithms are still tested on these well-known instances. They consist of 56 instances split into 6 different categories $C1, C2, R1, R2, RC1, RC2$, with slight differences within each category. These results are shown in Table 20. Even though our algorithm was not designed for this model it performs reasonably well with an average gap of 0.89%. It does take quite a bit of computing time to achieve these results. However, these might be reduced with proper tuning of the parameters for this particular model.

Table 20: VRPTW results

Instance	SA-ILP		BKS	gap(%)		Total time(s)
	AVG 5	MIN 5		AVG	MIN	
R1	1186.37	1185.63	1178.98	0.63	0.56	157.00
R2	891.23	887.63	877.20	1.60	1.19	280.50
C1	829.25	828.38	828.38	0.11	0.00	136.25
C2	589.86	589.86	589.86	0.00	0.00	265.81
RC1	1359.00	1352.26	1338.18	1.56	1.05	154.79
RC2	1018.74	1014.89	1003.95	1.47	1.09	247.93
AVG				0.89	0.65	206.14

E Tables

Table 21: Results for configuration 1

C	network	AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		LS + ILP	LS	imp(%)	LS + ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	0.00	73.87	73.87	0.00	85.72	1.95	83.77
50	Madrid	73.97	73.97	0.00	73.97	73.97	0.00	76.01	0.34	75.67
50	Pittsburgh	87.07	87.18	0.13	87.07	87.26	0.22	80.85	0.53	80.32
50	Seattle	58.95	58.95	0.00	58.93	58.93	0.00	98.70	16.77	81.93
50	Sydney	95.34	95.34	0.00	95.34	95.34	0.00	80.13	2.12	78.01
75	Fukuoka	95.14	95.22	0.08	95.02	95.15	0.14	1547.91	1462.72	85.19
75	Madrid	97.83	97.97	0.15	97.69	97.69	0.00	86.04	4.19	81.86
75	Pittsburgh	110.75	110.75	0.00	110.75	110.75	0.00	89.90	3.41	86.49
75	Seattle	79.83	79.83	0.00	79.83	79.83	0.00	124.01	25.26	98.75
75	Sydney	122.75	123.73	0.79	122.39	123.04	0.53	106.75	27.78	78.97
100	Fukuoka	118.87	119.09	0.19	118.87	119.53	0.55	3694.70	3601.26	93.44
100	Madrid	125.04	126.31	1.01	124.80	126.64	1.46	94.14	5.40	88.74
100	Pittsburgh	134.62	135.17	0.41	134.02	134.16	0.10	141.36	47.68	93.68
100	Seattle	101.53	101.98	0.45	101.32	102.07	0.73	223.55	120.77	102.78
100	Sydney	150.01	150.40	0.26	149.77	150.47	0.46	511.92	423.96	87.96

Table 22: Results for configuration 2

C	network	AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		LS + ILP	LS	imp(%)	LS + ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	0.00	73.87	73.87	0.00	133.09	2.53	130.56
50	Madrid	73.97	73.97	0.00	73.97	73.97	0.00	117.34	0.44	116.90
50	Pittsburgh	87.07	87.14	0.09	87.07	87.07	0.00	124.80	0.56	124.23
50	Seattle	58.97	59.00	0.05	58.93	59.01	0.14	148.32	21.28	127.04
50	Sydney	95.34	95.34	0.00	95.34	95.34	0.00	122.03	1.83	120.19
75	Fukuoka	95.12	95.18	0.06	95.02	95.15	0.14	2288.28	2156.86	131.43
75	Madrid	97.75	97.76	0.01	97.69	97.69	0.00	128.64	3.80	124.84
75	Pittsburgh	110.75	110.80	0.05	110.75	111.00	0.23	137.03	4.77	132.26
75	Seattle	79.83	79.83	0.00	79.83	79.83	0.00	184.73	30.26	154.47
75	Sydney	122.68	123.35	0.55	122.68	123.21	0.43	137.38	14.32	123.07
100	Fukuoka	118.93	119.04	0.09	118.87	119.38	0.43	3743.13	3601.35	141.78
100	Madrid	125.09	125.48	0.31	125.07	125.07	0.00	138.70	5.32	133.37
100	Pittsburgh	134.33	135.43	0.81	133.96	135.21	0.92	196.53	52.32	144.21
100	Seattle	101.34	101.97	0.62	101.32	101.84	0.51	366.39	205.36	161.03
100	Sydney	149.91	150.13	0.15	149.64	149.64	0.00	987.70	853.31	134.39

Table 23: Results for configuration 3

C	network	AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		LS + ILP	LS	imp(%)	LS + ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	0.00	73.87	73.87	0.00	122.37	2.30	120.07
50	Madrid	73.97	73.97	0.00	73.97	73.97	0.00	106.92	0.46	106.47
50	Pittsburgh	87.07	87.11	0.04	87.07	87.07	0.00	113.77	0.64	113.13
50	Seattle	58.93	58.95	0.03	58.93	58.93	0.00	132.83	16.13	116.70
50	Sydney	95.34	95.34	0.00	95.34	95.34	0.00	113.33	2.46	110.87
75	Fukuoka	95.10	95.17	0.08	95.02	95.15	0.13	2102.03	1981.50	120.53
75	Madrid	97.88	97.99	0.11	97.76	97.76	0.00	120.46	5.90	114.56
75	Pittsburgh	110.74	110.74	0.00	110.71	110.71	0.00	125.60	4.76	120.84
75	Seattle	79.83	79.83	0.00	79.83	79.83	0.00	174.60	31.85	142.75
75	Sydney	122.67	123.44	0.62	122.39	123.21	0.66	134.83	22.61	112.22
100	Fukuoka	118.90	119.02	0.10	118.87	118.87	0.00	3733.59	3601.43	132.16
100	Madrid	124.96	125.52	0.44	124.80	125.54	0.59	129.08	5.42	123.66
100	Pittsburgh	134.39	135.23	0.62	133.99	134.12	0.10	190.73	57.61	133.12
100	Seattle	101.40	101.86	0.45	101.32	101.92	0.58	319.84	168.68	151.16
100	Sydney	149.82	150.13	0.20	149.57	149.57	0.00	675.42	552.02	123.40

Table 24: Results for configuration 4

C	network	AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		LS + ILP	LS	imp(%)	LS + ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	0.00	73.87	73.87	0.00	141.03	3.17	137.86
50	Madrid	73.97	73.97	0.00	73.97	73.97	0.00	116.38	0.47	115.91
50	Pittsburgh	87.07	87.07	0.00	87.07	87.07	0.00	127.72	0.68	127.04
50	Seattle	58.93	58.93	0.00	58.93	58.93	0.00	143.39	15.24	128.15
50	Sydney	95.34	95.34	0.00	95.34	95.34	0.00	126.70	3.50	123.19
75	Fukuoka	95.12	95.15	0.03	95.02	95.15	0.13	3603.53	3469.45	134.08
75	Madrid	97.72	97.93	0.21	97.69	97.76	0.07	132.43	8.24	124.19
75	Pittsburgh	110.77	110.78	0.01	110.71	110.71	0.00	139.08	4.63	134.45
75	Seattle	79.83	79.83	0.00	79.83	79.83	0.00	206.75	45.90	160.85
75	Sydney	122.68	123.13	0.36	122.39	122.97	0.47	152.85	36.31	116.54
100	Fukuoka	118.89	118.97	0.06	118.87	119.08	0.18	3744.16	3601.22	142.94
100	Madrid	124.99	125.85	0.68	124.80	125.08	0.23	137.52	7.65	129.88
100	Pittsburgh	133.99	134.99	0.74	133.74	134.02	0.21	183.29	39.87	143.42
100	Seattle	101.32	101.89	0.56	101.32	101.76	0.43	388.74	234.53	154.21
100	Sydney	149.85	150.17	0.21	149.58	150.28	0.46	2107.08	1975.17	131.91

Table 25: Results for configuration 5

C	network	AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		LS + ILP	LS	imp(%)	LS + ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	0.00	73.87	73.87	0.00	122.96	2.56	120.40
50	Madrid	73.97	73.97	0.00	73.97	73.97	0.00	105.35	0.47	104.89
50	Pittsburgh	87.07	87.14	0.09	87.07	87.07	0.00	113.95	0.60	113.35
50	Seattle	58.95	58.97	0.03	58.93	58.93	0.00	130.49	15.56	114.93
50	Sydney	95.34	95.34	0.00	95.34	95.34	0.00	111.71	2.79	108.93
75	Fukuoka	95.07	95.19	0.12	95.02	95.25	0.24	3341.24	3220.64	120.60
75	Madrid	97.80	98.00	0.21	97.69	97.69	0.00	119.18	6.65	112.53
75	Pittsburgh	110.72	110.73	0.01	110.71	110.71	0.00	124.49	4.71	119.77
75	Seattle	79.83	79.83	0.00	79.83	79.83	0.00	179.30	36.42	142.88
75	Sydney	122.94	123.69	0.61	122.78	123.74	0.78	148.40	39.28	109.11
100	Fukuoka	118.79	119.09	0.25	117.99	119.12	0.95	3732.65	3600.63	132.03
100	Madrid	125.02	125.55	0.42	124.80	125.28	0.39	124.25	4.61	119.65
100	Pittsburgh	134.10	135.48	1.02	133.66	135.66	1.48	193.62	64.19	129.43
100	Seattle	101.41	101.88	0.46	101.32	101.57	0.24	369.31	224.24	145.07
100	Sydney	149.86	150.17	0.21	149.57	150.28	0.47	1450.56	1328.76	121.80

Table 26: Results for configuration 6

C	network	AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		LS + ILP	LS	imp(%)	LS + ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	0.00	73.87	73.87	0.00	172.17	2.80	169.38
50	Madrid	73.97	73.97	0.00	73.97	73.97	0.01	146.84	0.57	146.27
50	Pittsburgh	87.07	87.07	0.00	87.07	87.07	0.00	159.92	1.38	158.54
50	Seattle	58.93	58.93	0.00	58.93	58.93	0.00	231.27	68.76	162.51
50	Sydney	95.34	95.34	0.00	95.34	95.34	0.00	164.93	5.63	159.31
75	Fukuoka	95.10	95.15	0.05	95.02	95.15	0.13	3776.12	3610.03	166.09
75	Madrid	97.69	97.69	0.00	97.69	97.69	0.00	164.73	9.46	155.27
75	Pittsburgh	110.71	110.71	0.00	110.71	110.71	0.00	181.03	9.61	171.42
75	Seattle	79.83	79.83	0.00	79.83	79.83	0.00	247.46	47.32	200.14
75	Sydney	122.69	122.77	0.07	122.68	122.79	0.09	187.04	45.09	141.94
100	Fukuoka	118.88	118.97	0.08	118.87	118.87	0.00	3789.01	3606.68	182.34
100	Madrid	124.96	125.73	0.61	124.80	126.10	1.03	178.16	12.00	166.17
100	Pittsburgh	134.45	135.05	0.44	133.66	133.76	0.07	320.81	147.43	173.38
100	Seattle	101.32	101.60	0.27	101.32	101.70	0.37	586.23	391.70	194.53
100	Sydney	149.74	149.80	0.04	149.63	149.82	0.13	2115.80	1957.58	158.22

Table 27: Results for configuration 7

C	network	AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		LS + ILP	LS	imp(%)	LS + ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	0.00	73.87	73.87	0.00	95.78	1.27	94.51
50	Madrid	74.01	74.01	0.00	73.97	73.97	0.00	91.58	0.38	91.20
50	Pittsburgh	87.18	87.58	0.46	87.07	87.46	0.44	90.58	0.44	90.14
50	Seattle	59.00	59.01	0.01	58.93	58.93	0.00	97.99	5.50	92.48
50	Sydney	95.34	95.34	0.00	95.34	95.34	0.00	93.96	0.78	93.19
75	Fukuoka	95.35	95.68	0.35	95.19	96.10	0.95	213.65	114.32	99.33
75	Madrid	97.95	98.64	0.70	97.69	98.91	1.23	95.83	2.62	93.21
75	Pittsburgh	111.59	112.20	0.55	110.75	111.11	0.33	104.06	4.35	99.71
75	Seattle	80.03	80.48	0.56	79.83	80.82	1.22	123.93	7.39	116.54
75	Sydney	123.64	124.03	0.31	122.93	123.06	0.10	114.48	14.23	100.25
100	Fukuoka	118.99	119.71	0.60	118.87	119.30	0.36	3314.69	3205.26	109.43
100	Madrid	125.80	127.67	1.46	125.34	127.38	1.60	101.26	4.59	96.67
100	Pittsburgh	136.63	137.94	0.96	136.22	138.12	1.38	196.33	93.59	102.73
100	Seattle	102.19	103.93	1.67	101.84	104.19	2.25	228.51	96.63	131.88
100	Sydney	150.87	151.19	0.21	150.79	151.23	0.29	154.48	46.58	107.90

Table 28: Results for configuration 8

C	network	AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		LS + ILP	LS	imp(%)	LS + ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	0.00	73.87	73.87	0.00	72.18	2.00	70.18
50	Madrid	73.97	73.97	0.00	73.97	73.97	0.00	60.52	0.40	60.12
50	Pittsburgh	87.07	87.18	0.13	87.07	87.07	0.00	64.97	0.53	64.44
50	Seattle	58.95	58.98	0.05	58.93	59.01	0.14	81.45	15.73	65.72
50	Sydney	95.34	95.34	0.00	95.34	95.34	0.00	66.14	2.95	63.18
75	Fukuoka	95.08	95.20	0.13	95.02	95.15	0.14	1538.02	1468.13	69.89
75	Madrid	97.77	97.83	0.06	97.69	97.69	0.00	68.78	4.30	64.47
75	Pittsburgh	110.72	110.88	0.14	110.71	110.75	0.04	74.50	4.56	69.94
75	Seattle	79.83	79.85	0.02	79.83	79.83	0.00	112.80	30.38	82.42
75	Sydney	122.85	123.59	0.60	122.39	123.76	1.11	97.01	35.66	61.35
100	Fukuoka	118.58	119.30	0.60	117.03	119.16	1.79	3676.36	3601.05	75.31
100	Madrid	125.02	126.36	1.06	124.80	125.56	0.61	73.59	4.51	69.08
100	Pittsburgh	134.40	135.41	0.75	133.96	135.05	0.81	133.13	58.00	75.13
100	Seattle	101.61	102.36	0.73	101.43	101.64	0.20	246.53	164.81	81.73
100	Sydney	150.03	150.41	0.25	149.73	150.36	0.42	841.18	771.99	69.19

Table 29: Results for configuration 9

C	network	AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		LS + ILP	LS	imp(%)	LS + ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	0.00	73.87	73.87	0.00	140.30	2.52	137.79
50	Madrid	73.97	73.97	0.00	73.97	73.97	0.00	119.59	0.46	119.13
50	Pittsburgh	87.07	87.07	0.00	87.07	87.07	0.00	129.29	1.10	128.19
50	Seattle	58.93	58.93	0.00	58.93	58.93	0.00	187.53	55.43	132.10
50	Sydney	95.34	95.34	0.00	95.34	95.34	0.00	135.09	5.06	130.03
75	Fukuoka	95.15	95.19	0.04	95.15	95.15	0.01	3798.71	3663.84	134.86
75	Madrid	97.71	97.90	0.20	97.69	98.40	0.72	135.77	8.59	127.18
75	Pittsburgh	110.71	110.71	0.01	110.71	110.75	0.04	147.77	9.18	138.59
75	Seattle	79.83	79.83	0.00	79.83	79.83	0.00	203.66	42.87	160.79
75	Sydney	122.56	122.74	0.15	122.39	122.87	0.39	140.68	23.57	117.11
100	Fukuoka	118.70	119.04	0.29	117.98	119.16	0.99	3750.29	3601.14	149.14
100	Madrid	124.91	125.27	0.29	124.80	125.07	0.22	145.88	9.28	136.60
100	Pittsburgh	134.38	135.24	0.64	133.96	135.34	1.03	209.57	68.45	141.12
100	Seattle	101.36	101.64	0.28	101.32	101.66	0.33	510.93	351.63	159.30
100	Sydney	149.69	149.87	0.12	149.50	149.91	0.28	1185.22	1056.69	128.54

Table 30: Results for configuration 10

C	network	AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		LS + ILP	LS	imp(%)	LS + ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	0.00	73.87	73.87	0.00	138.52	2.41	136.11
50	Madrid	73.97	73.97	0.00	73.97	73.97	0.00	118.18	0.49	117.69
50	Pittsburgh	87.07	87.07	0.00	87.07	87.07	0.00	128.01	1.25	126.75
50	Seattle	58.93	58.93	0.00	58.93	58.93	0.00	178.94	48.78	130.16
50	Sydney	95.34	95.34	0.00	95.34	95.34	0.00	132.86	4.37	128.48
75	Fukuoka	95.15	95.19	0.04	95.15	95.15	0.01	3752.16	3618.59	133.58
75	Madrid	97.78	97.88	0.10	97.69	97.69	0.00	138.23	12.63	125.60
75	Pittsburgh	110.71	110.71	0.00	110.71	110.71	0.00	144.78	8.07	136.72
75	Seattle	79.83	79.83	0.00	79.83	79.83	0.00	195.49	37.17	158.33
75	Sydney	122.62	122.80	0.15	122.39	123.01	0.50	143.34	27.73	115.61
100	Fukuoka	118.87	119.05	0.15	118.87	118.88	0.01	3749.04	3603.31	145.73
100	Madrid	125.02	125.76	0.59	124.80	125.14	0.27	144.18	10.01	134.17
100	Pittsburgh	134.47	135.07	0.45	133.66	133.96	0.22	223.48	84.73	138.75
100	Seattle	101.35	101.78	0.42	101.32	101.55	0.22	483.26	328.98	154.27
100	Sydney	149.81	150.08	0.18	149.63	150.46	0.55	861.70	735.34	126.36

Table 31: Results for configuration 11

C	network	AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		LS + ILP	LS	imp(%)	LS + ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	0.00	73.87	73.87	0.00	139.89	2.60	137.29
50	Madrid	73.97	73.97	0.00	73.97	73.97	0.00	119.34	0.46	118.88
50	Pittsburgh	87.07	87.11	0.04	87.07	87.07	0.00	128.33	1.30	127.03
50	Seattle	58.93	58.93	0.00	58.93	58.93	0.00	187.30	56.08	131.21
50	Sydney	95.34	95.34	0.00	95.34	95.34	0.00	133.44	4.08	129.36
75	Fukuoka	95.13	95.15	0.03	95.02	95.15	0.14	3781.30	3646.42	134.88
75	Madrid	97.72	97.84	0.12	97.69	97.69	0.00	135.37	8.76	126.61
75	Pittsburgh	110.71	110.71	0.00	110.71	110.71	0.00	146.08	8.63	137.45
75	Seattle	79.83	79.83	0.00	79.83	79.83	0.00	195.06	35.21	159.86
75	Sydney	122.62	122.80	0.15	122.39	122.67	0.23	152.35	35.94	116.41
100	Fukuoka	118.87	119.00	0.11	118.87	118.87	0.00	3747.52	3601.00	146.52
100	Madrid	125.12	125.97	0.68	124.80	126.14	1.07	155.78	20.70	135.08
100	Pittsburgh	134.43	134.94	0.38	133.66	133.66	0.00	237.53	97.12	140.41
100	Seattle	101.36	101.72	0.35	101.32	101.61	0.28	523.59	367.27	156.31
100	Sydney	149.71	150.08	0.24	149.57	149.89	0.22	1628.92	1501.64	127.27

Table 32: Results for configuration 12

C	network	AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		LS + ILP	LS	imp(%)	LS + ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	0.00	73.87	73.87	0.00	132.36	2.80	129.56
50	Madrid	73.97	73.97	0.00	73.97	73.97	0.00	121.21	0.44	120.77
50	Pittsburgh	87.07	87.18	0.13	87.07	87.07	0.00	129.97	0.75	129.22
50	Seattle	58.95	58.95	0.00	58.93	58.93	0.00	148.76	17.43	131.34
50	Sydney	95.34	95.34	0.00	95.34	95.34	0.00	126.88	3.71	123.16
75	Fukuoka	95.11	95.15	0.05	95.02	95.15	0.14	3760.67	3620.95	139.72
75	Madrid	97.71	97.78	0.07	97.69	97.99	0.31	131.82	6.52	125.29
75	Pittsburgh	110.83	110.93	0.09	110.71	110.71	0.00	142.84	6.61	136.23
75	Seattle	79.83	79.83	0.00	79.83	79.83	0.00	196.06	40.10	155.96
75	Sydney	122.65	123.20	0.44	122.39	122.97	0.47	147.08	31.05	116.04
100	Fukuoka	118.62	118.99	0.31	117.50	118.95	1.22	3740.28	3600.65	139.64
100	Madrid	124.96	125.73	0.61	124.80	125.21	0.33	134.98	5.02	129.96
100	Pittsburgh	134.14	135.42	0.95	133.66	135.54	1.39	210.85	66.81	144.04
100	Seattle	101.37	101.78	0.40	101.32	101.75	0.42	425.02	275.64	149.38
100	Sydney	149.87	150.06	0.13	149.58	150.21	0.42	1845.37	1714.78	130.59

Table 33: Results for configuration 13

C	network	AVG 5			MIN 5			Total time(s)	ILP time(s)	LS time(s)
		LS + ILP	LS	imp(%)	LS + ILP	LS	imp(%)			
50	Fukuoka	73.87	73.87	0.00	73.87	73.87	0.00	171.31	2.77	168.54
50	Madrid	73.97	73.97	0.00	73.97	73.97	0.00	145.95	0.51	145.44
50	Pittsburgh	87.07	87.07	0.00	87.07	87.07	0.00	157.11	1.31	155.80
50	Seattle	58.93	58.93	0.00	58.93	58.93	0.00	237.95	76.31	161.64
50	Sydney	95.34	95.34	0.00	95.34	95.34	0.00	162.63	6.17	156.46
75	Fukuoka	95.15	95.15	0.00	95.15	95.15	0.00	3763.31	3601.04	162.27
75	Madrid	97.69	97.69	0.00	97.69	97.69	0.00	157.50	7.59	149.91
75	Pittsburgh	110.71	110.71	0.01	110.71	110.75	0.04	183.69	10.32	173.37
75	Seattle	79.83	79.83	0.00	79.83	79.83	0.00	238.75	48.09	190.66
75	Sydney	122.51	122.71	0.17	122.39	122.79	0.33	160.33	25.24	135.09
100	Fukuoka	118.87	118.91	0.03	118.87	118.87	0.00	3785.95	3609.54	176.41
100	Madrid	124.96	125.45	0.39	124.80	125.80	0.79	176.10	11.88	164.22
100	Pittsburgh	134.32	135.18	0.64	133.66	134.33	0.50	265.43	93.99	171.44
100	Seattle	101.32	101.55	0.22	101.32	101.72	0.39	582.55	402.31	180.24
100	Sydney	149.65	149.94	0.19	149.46	150.02	0.37	2803.41	2652.79	150.62

F Additional wind results

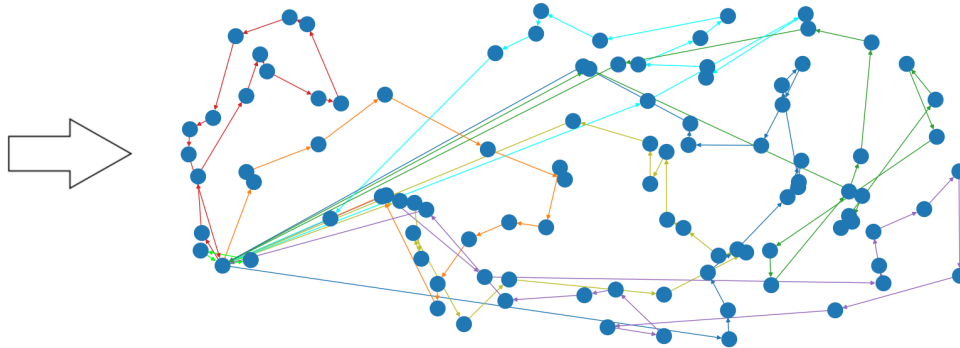


Figure 4: A solution found with westerly wind

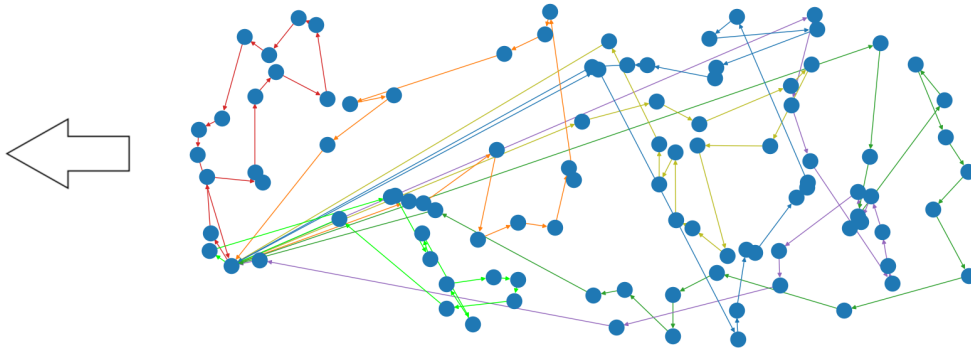


Figure 5: A solution found with easterly wind

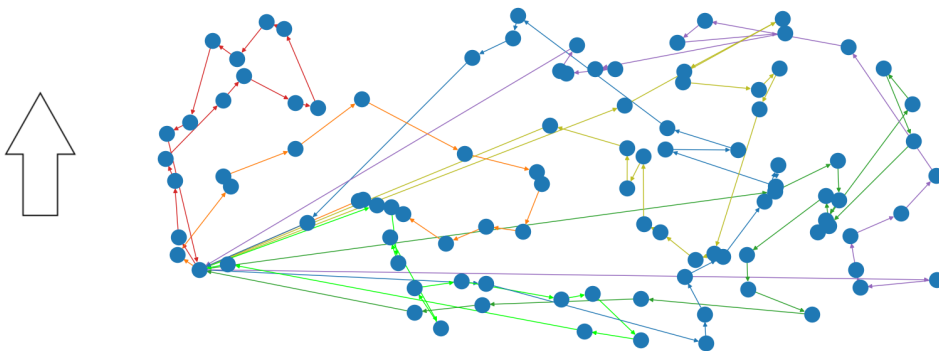


Figure 6: A solution found with southerly wind

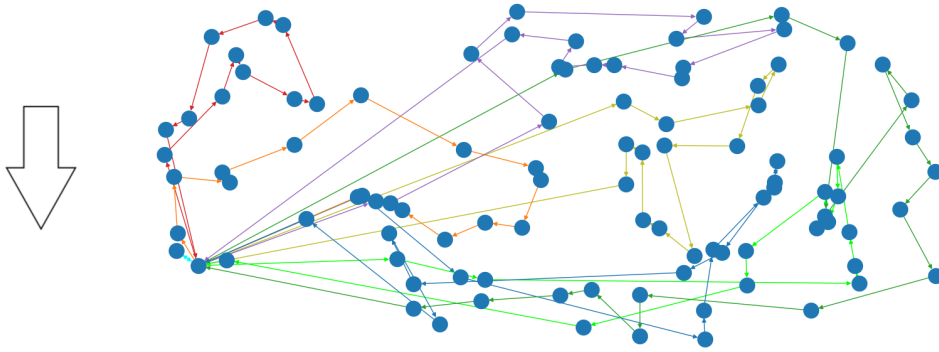


Figure 7: A solution found with northerly wind