UTRECHT UNIVERSITY

COMPUTER SCIENCE

MASTER THESIS

Using column generation for the Time Dependent Vehicle Routing Problem with Soft Time Windows and Stochastic Travel Times

Author:
G.P.T VAN LENT

Supervisors:
Dr. ir. J.M. VAN DEN AKKER
Dr. J.A. HOOGEVEEN

January 25, 2018

Abstract

The vehicle Routing Problem with Time Windows and time dependent stochastic travel times (TDVRPTW) is extended by using continuous distribution functions to model variation in travel times directly into the objective function. The used model is flexible by allowing different travel time distributions for different periods of the day to allow the modeling of realistic situations with often changing levels of traffic congestion. In this thesis the cases where the travel times are either normally distributed or gamma distributed are covered, although the techniques can be applied to other distributions as well.

Using this model the objective function can accurately predict the cost of a route, taking into account both the travel costs (traveling and vehicle cost) as well as the service costs (probability of being late or a driver suffering overtime). To meet minimum service level requirements we maintain a minimum feasibility of serving each customer on time. The model allows an early decision to not further explore routes that already are infeasible due to not meeting a threshold probability of serving some customers on their route within their time window. The problem is solved using column generation to solve a linear programming problem consisting of selecting a sub set of routes out of a set of selected high quality candidate routes. Candidate routes are selected using two different heuristics to solve the pricing problem, local search and dynamic programming.

Contents

1 Introduction

The Vehicle Routing Problem(VRP) is defined as the problem to find the least cost routes for vehicles from one depot to a set of customers. The routes require that each customer is visited once by exactly one vehicle and that they all end and start at the same depot. The Time-Dependent Vehicle Routing Problem with Time Windows and Stochastic Travel Times (TDVRPTW) is a variant on the VRP. In this variant each customer must be visited within a given time interval specific for that customer. Each customer has a time window with a starting and an ending time in between the customer must be served. The day is divided into time periods to allow the modeling of variation in the traffic congestion during the day. The travel times are stochastic with for each arc in each time period different distribution parameters.

Because stochastic travel times are considered there is no deterministic notion of serving a customer on time. Instead the probability that a customer is served within its time window is considered. Commonly there are two types of time windows, hard time windows and soft time windows. In case of hard time windows it is not allowed to serve a customer outside of its time window. In case of soft time windows it is allowed to serve a customer outside of its time window, but this induces a penalty in the objective cost. We do not allow a customer to be served early, if a vehicle arrives at a customer before the release date the vehicle is required to wait until the time window of the customer starts. This means that we only need to consider late servicing, the case where the vehicle might arrive after the end if the time window. Since we work with probabilities, we will set a hard and a soft constraint on the probability that a customer is served on time. On the probability of being late at a customer we set a penalty and the user can define the minimum required probability of a vehicle being on time at a customer.

Several solution approaches are already proposed and documented for the VRP and for TDVRPTW. Lekkerkerker [13] solves the problem using a column generation heuristic to generate a schedule, and uses 100 simulated worlds with different travel times to test whether a solution is reliable enough to be accepted. Our approach continues on this approach, but instead of checking the reliability with the use of simulation worlds, we will use probability distribution functions for the travel time directly in the objective function. One major advantage is that this removes the correlation between the number of simulation worlds used and the performance and calculation time.

This thesis starts with an overview of the previous literature on vehicle routing, from the classical case of VRP to the specific problem of TDVRPTW. Chapter 3 explains the relevance of the research problem. This is followed by the formal problem definition. Next the use of continuous distribution functions is explained in detail. Followed by a description of column generation to solve linear programming and two different methods, local search and dynamic programming, to solve the

pricing problem required to build the linear programming. Finally we will describe the experiments and their results.

2 Literature review

The problem currently known as the Vehicle Routing Problem(VRP) started with the introduction of the "Truck Dispatching Problem" by Dantzig and Ramser [3], modeling how a fleet of homogeneous trucks could serve the demand for oil of a number of gas stations from a central hub, with a minimum travel distance. Clarke and Wright [2] reformulated this problem in a linear optimization problem, the VRP, using a fleet of trucks with varying capacity to serve customers from a depot.

The traditional objective of the VRP is to minimize the total cost, which is mostly defined as the sum of the cost of the distances traveled, the number of vehicles used, the cost of overtime or any combination of these. There is a wide range of literature about the VRP and all kinds of variations on the problem as the topic is very relevant nowadays. The problem however is NP-hard, meaning that it takes most likely exponential time in the size of the problem instance to solve it optimally. Laporte [12] reviews the main results of different exact algorithms and heuristics for the VRP. Concluding that VRP remains difficult to solve optimally. The exact algorithms that performed best are based on a branch-and-cut approach and are able to solve instances involving up to about 100 customers. For larger problem instances, metaheuristics are preferred, as they reduce the calculation time and still yield results close to the optimal or best-known solution.

Currently there exist many variations on the VRP model as there is an increasing aim to incorporate more real-life complexities, such as time windows, stochastic travel times, time dependent travel times (reflecting peak hours, traffic jams etc), pickup and delivery and dynamic input changes.
This brings substantial complexity to the problem. In the variant Vehicle Routing Problem with Time windows (VRPTW) each customer has a time window within the deliveries must take place. This time window problem is further divided in Soft Time Windows and Hard Time Windows. In the case of Hard Time Windows a solution is not feasible if a customer is served outside its Time Window. In the case of Soft Windows, a violation is allowed but results in a penalty within the objective function. An optimal solution will therefore still generally minimize the number of customers served outside of their time window.

In the next section we will review in more detail the vehicle routing problem with stochastic travel time elements.

2.1 Vehicle routing problem with stochastic travel times

In the traditional formulation of the VRP and the VRPTW all elements are considered deterministic. However in real life situation, all kinds of uncertainty exist. An example is uncertainty in travel times due to traffic jams or traffic lights. If these uncertainties are ignored, theoretically optimal schedules may still perform

poorly in practice. To overcome this problem, VRP with Stochastic Travel times was introduced. Gendreau et al. [7] review some of the problems and show that uncertainty can play a role in many of the components of the VRP. Stochastic demands, stochastic customers and stochastic travel times are some examples of this. Gendreau et al. [7] reviewed different algorithms and concluded that algorithms that take stochasticity into account perform better in general in real life situations.

Tas et al [18] consider the VRP with stochastic travel times including soft time windows and service costs. With service cost we mean a penalty in case a customer is serviced outside his time window, in case the delivery is to early or to late.

They introduce an objective function which consists of the service and the transportation costs. Allowing different combinations of importance between these two components are considered, as they may depend on a company's preference. They process the expected overtime, early, and late services in their objective function to model the soft time windows. However they do not consider time-dependency of travel times during the day. An optimal solution is sought using a Tabu search method (a local search approach), which however often leads to a local maximum. In a later paper Tas et al. [19] improve their method by solving the same objective function with a column generation method, which resulted in better results. An initial branch-and-price method is used to obtain an integer solution, which is the usual performance bottleneck for column generation approaches .

As the level of traffic congestion normally varies during the day, algorithms which consider time dependent travel times are more accurate in modeling real world problems. Malandraki and Daskin [14] examine the time dependent vehicle routing problem (TDVRP). In their model travel times are computed by a simple step function, consisting of multiple piece wise constant functions, which give a travel time depending on the time of the day. This means that a schedule may become more efficient if a vehicle simply waits to leave a customer until a better time of the day. The drawback of this method is that it does not satisfy the first-in-first-out (FIFO) property, meaning that if a vehicle leaves at a node i to go to node j at a given time, any other vehicle leaving node i for node j at a later moment will arrive later at node j. Not satisfying this property leads to inconsistencies and is not a realistic representation of the real world. Vehicles now might wait at some location until a time that speeds are higher and then arrive at the desired location before another vehicle who left before. It would make more sense if the vehicle just starts driving and when the boundary between the consecutive time periods is crossed the speed for the vehicle changes. Ichoua, Gendreau and Potvin [10] introduced a step wise speed model, which has as main point that they do not assume a constant speed over the entire distance between 2 nodes. The speed changes when the boundary between two consecutive time periods is crossed. This guarantees the FIFO property. Results showed that the time-dependent model provided very significant improvements over the model with fixed travel time for the robustness of the schedule.

Lekkerkerker [13] solves TDVRPTW with stochastic travel times using column generation, with heuristics to find good candidate routes and then tests the reliability of the generated schedule, by running multiple simulation worlds, until the reliability of a schedule is above a desired threshold. They explored multiple methods to solve the pricing problem, the local search method and dynamic programming heuristic found good solutions while the mixed integer programming methods were not performing well.

3 Relevance

The optimization of vehicle routing models has a number of benefits to logistics and transport operations. Transportation costs are estimated to account for around half of the logistic costs in industries, with some industries, like the food and drink industries, it even accounts for up to 70% of the costs [1]. This number seems to increase even further because of an increasing demand in smaller, faster and more on time deliveries. The increasing amount of transportation and the high cost pressure from the raising energy prices and the taxes for ecological impacts causes the gain in importance of efficient vehicle routing problems. Less travel time and distance mean a decrease in fuel consumption and CO2 emissions. It improves the resource utilization and therefor the cost for government and companies dealing with transportation problems. Real world Vehicle Routing Problems are hard optimization problems with many challenges and variations. There are many research studies which study Vehicle Routing Problems, however the research on time dependent and stochastic travel times is not so comprehensive. The studies that do include these factors often show significant results, however computation time is still an issue and many studies use an heuristic algorithm to compensate this with a near optimal result. Improving the computation time for these problems allows for fast recalculation of the routes in case an unexpected incident occurs. Increasing the flexibility for adding new constraints, which is the case with the set-partition solution method, allows to change the requirements more easily over time.

4 Formal problem definition

In this section we formulate the problem as an integer linear programming problem, which we later will write as a set partition problem. A similar objective formulation can already be found in the paper of Tas et al. [18]. The objective function for the Time Dependent Stochastic Vehicle Routing Problem with Time Windows takes transportation cost as well as service costs into account. The transportation costs consist of the total distance traveled and the total number of vehicles used. The service cost component is based on penalties for the early and late arrival at a customer which are based on stochastic variables. In the end of this chapter we discuss the calculation of the expected delay, earliness and expected overtime. In general we can define functions for the expected delay, earliness and overtime for different probability distributions of the travel times, initially we will consider the case that the travel times are normal distributed and after that we consider the case where the travel times are gamma distributed.

4.1 Objective goals

Within our VRP we have two defining factors:

- Transportation costs
- Customer service satisfaction

While we naturally want to keep transportation costs as low as possible, we also have to maintain an acceptable level of customer satisfaction and serve customers within a given time window. To balance the two we will define a single cost-, or objective function, which takes both factors into account, which we can then minimize to get to an optimal solution. The ratio of importance of both factors can be adjusted to the preferences of the user by increasing or decreasing the penalties in the service component.

4.2 Graph representation

We start by modeling our VPR in a graph:

- $N = \{0, 1, ..., n\}$, The set of all customers nodes.
- $A = \{(i, j)|i, j \in N, i \neq j\}$, All potential driving routes between all customers.
- V is the set of vehicles.
- 0, The depot where all vehicles begin their trip.
- d_{ij}, The driving cost from node i to node j.
- T_{ij}^{p} Time dependent travel time from customer i to j in period p.

The hours of the days are divided into consecutive periods, each period is defined by its starting and its ending time and each time during the day must be contained in exactly one period.

4.3 Vehicle representation

For our vehicles we have multiple costs associated with operating a vehicle:

- c_f, the cost of using one additional vehicle in the schedule.
- c_o, the cost of the probability that the driver is scheduled longer than a normal workday.

4.4 Customer representation

For our customers we have a given demand and a time window, with penalties on serving outside of that time window.

- $[l_i, u_i]$, The lower and upper bound of the time window in which customer i needs to be served
- c_e, the cost of the probability of arriving at a customer earlier than its start window. In the case the vehicle has to wait to serve the customer until its start window, this is the penalty for waiting. In the case without waiting it is the cost of serving a customer to early.
- c_d, the cost of the probability of serving a customer later than the end of its Time window.

4.5 Schedule variables

Finally the decision variables which indicate what our solution schedule will look like:

- x_{ijv}, a decision variable, 1 if vehicle v goes from customer i to customer j consecutively, 0 otherwise.

4.6 Transportation costs

We will now use the variables introduced in the previous sections to model the transportation costs within the objective function. Naturally, we would like to keep the transportation costs as low as possible, as driving inefficient routes costs gas money, work time, and potentially even having to use additional vehicles. To do so we add the following variables to the objective function for the transportation costs:

- $c_t \sum_{i \in N} \sum_{j \in N} \sum_{v \in V} d_{ij} x_{ijv}$, The cost of driving, multiplied by the sum of the distance over all chosen routes

- $c_f \sum_{j \in N} \sum_{v \in V} x_{ojv}$, The cost of hiring a vehicle, multiplied by the sum of vehicles that leave the depot.
- $c_o \sum_{v \in V} O_v$, the cost of overtime, multiplied by the sum of the probability of overtime over all vehicles. The total expected overtime is found by calculating for each scheduled route the probability that the vehicle makes overtime and adding them. The calculation of the expected overtime for one single route will be discussed later.

4.7 Service costs

However, we also have to have a good level of customer satisfaction and stay within the time windows as much as possible. An additional service cost is therefore added to our objective function depending on how likely a customer will be served too early or too late. While we will expand further on how we calculate this likeliness in the next section, for now we will simply define:

- D_{jv}, the probability of delay of vehicle v at customer j.
- $\sum_{i \in N} \sum_{j \in N} c_d D_{jv} x_{ijv}$, the cost not being served on time, the sum of expected delay of vehicle v at all its customers j.
- E_{jv}, the expected earliness of vehicle v at customer j.
- $\sum_{i \in N} \sum_{j \in N} c_e E_{jv} x_{ijv}$, the cost of being too early, multiplied by the expected earliness of vehicle v at customer j.

4.8 Objective function

The objective function is based on the objective function of Tas et al. [18]. Combining the service and the transportation costs in our objective function we get:

$$\sum_{v \in V} \sum_{i \in N} \sum_{j \in N} (c_d D_{jv} + c_e E_{jv}) x_{ijv} + c_o \sum_{v \in V} O_v + c_t \sum_{i \in N} \sum_{j \in N} \sum_{v \in V} d_{ij} x_{ijv} + c_f \sum_{j \in N} \sum_{v \in V} x_{ojv}$$

The relative importance of the transportation costs compared to the different service cost components can be managed by increasing or decreasing the penalties. The calculation of D_{jv} and E_{jv} are described in detail in the next chapter.

4.9 Constraints

Any candidate solution schedule has to not violate any of the following constraints:

- $\sum_{i \in N} x_{ikv} - \sum_{j \in N} x_{kjv} = 0 \; k \in N \backslash \{o\}, v \in V$
  Make sure that each route visiting a customer also leaves that customer, conservation of flow.
- $\sum_{j \in N} \sum_{v \in V} x_{ijv} = 1, i \in N \backslash \{0\}$
  Make sure that each customer is visited by exactly one route.

- $\sum_{j \in N} x_{0jv} = 1, v \in V$
  Make sure that each route leaves the depot to exactly one customer.
- $\sum_{i \in N} x_{i0v} = 1, v \in V$
  Make sure that each route finishes at the depot.
- $x_{ijv} \in \{0,1\}$, $i \in N$, $j \in N$, $v \in V$.
  x_{ijv} needs to be 0 or 1 because we cannot serve customer partially by a vehicle.
- $\sum_{i \in N} D_{jv} x_{ijv} < \delta$, $\forall j \in N$ and $\forall v \in V$.
  This constraint is to make sure that the expectation of arriving to late at a customer in a selected route is less than a certain threshold value, δ.

By minimizing the objective function given these constraints we get an optimal solution to our problem. Which satisfies that at least every customer should be served on time with a probability greater than $1 - \delta$.

5 Calculations expected on time deliveries and expected overtime

The main focus of this thesis will be on the calculation of D_{jr} and O_r (the probability of a customer j being served later than his time window by vehicle r, and the probability of the vehicle having overtime) and to a lesser extend on E_{jr} (the probability of a customer being served before his time window).

We prioritize E_{jr} much less, because in cases where a customer would be served early, it would generally be possible to wait near the customer until the time window starts, thus serving the customer on time and avoiding any associated penalties. In these cases the extra waiting time can be considered part of the travel time from the previous customer, in most cases without any loss of problem modeling specificity. This has the benefit that we only have to consider the right tail (being late) of continuous distributions, simplifying modeling travel times stochastically more accurately.

We will investigate whether using stochastic distribution functions to model travel times directly into our objective function can predict the arrival time more accurately and therefore give a better result than using a sampling based method.

Using a distribution function could potentially increase the accuracy and robustness of schedules, however it should not increase the running time of the algorithm by too much to become unpractical.

This is where sampling based methods potentially have an extra edge as they can make a trade-off between running time and accuracy; using more samples increases the accuracy, at the cost of some running time. This gives the user direct control to fit his specific needs. Because such a trade-off does not exist when using distribution functions directly, the solution should already fit all the user's needs.

On top of that, the accuracy of using distribution functions relies entirely on the variance of the underlying data and our ability to find a distribution that matches this data close enough. If the variance in the data is relatively low then modeling and solving the problem deterministically (ignoring any possible variation from the mean) might give an already qualitative satisfying result in much less computing time. In this thesis we will examine the overall performance of modeling the travel times using normal- and gamma distributions directly in the objective function.

While we will mainly focus on using the stochastic travel time functions we will compare our results to deterministically found solutions.

To model different levels of road congestion at different time of the day, days are divided into different intervals. For each route between two customers i an j there exists a stochastic distribution for the travel time of the route in time period p, T_{ij}^p.

5.1 Normal distribution

In this subsection we will first show how to calculate the travel times T_{ij}^d and the expected probabilities of D_{jv} and O_v given that the travel times are normally distributed. In later subsections we will also show calculations given that the travel times have a gamma distributions which perhaps is more realistic for modeling travel times because they can capture skewness of the data.

We define:

- $T_{ij}^p \sim N(\mu_{ij}^p, \sigma_{ij}^p)$

the travel time from customer i to j at day part p is normally distributed with mean μ_{ij}^p and variance σ_{ij}^p. For obvious reasons, the travel time can never be negative.

5.1.1 Travel times

For the travel times we define the variables

- $E(T_{ij}^p)$, the expected travel time from customer i to j in time period p
- $var(T_{ij}^p)$, the variance in travel time traveling from i to j in time period p

Having different time periods introduces the problem of crossing from one time period into another during a single trip, changing the congestion on the road and therefore the expectation and variance in the travel time.

In case a trip has a longer expected travel time (x) than there is time left (y) until the next period starts ($x > y$),

- The first part of the trip will be taken using the fraction of how much can still be traveled within the first period ($\frac{y}{x}$), for which we will use the expected travel time and variance of the first period.
- The remainder of the trip ($\frac{x-y}{x}$) then needs to be computed using the expected time and variance of the next period.

This gives an expected travel time and variance for trips crossing over period bounds of:

- $E(T_{ij}^d) = \frac{y}{x}E(T_{ij}^p) + \frac{x-y}{x}E(T_{ij}^{p+1})$
- $var(T_{ij}^d) = (\frac{y}{x})^2 var(T_{ij}^p) + (\frac{x-y}{x})^2 var(T_{ij}^{p+1})$

Where d is the departure time at customer i. The above formulas give the case where the trip does fit into two periods. If the time windows are smaller it is possible that multiple periods are crossed during a single trip, in which case the same principle applies recursively.

Because starting a trip in a period -regardless of how long a trip takes in that period- always reduces how much of the trip needs to be traveled in the consecutive period, the FIFO property for expected arrival time is satisfied. It is not possible that waiting at a customer until a later period with more beneficial travel times, can reduce the expected arrival time at the next customer.

5.1.2 Arrival times without waiting

Because the arrival time at a customer depends on the arrival times at the customers visited before this customer, we will define the arrival time, A_{cr}, in a recursive way:

$$A_{cr} = A_{hr} + T_{hc}^{d(A_{hr})}$$

- Where A_{cr} is the arrival time at customer c given a route r
- $d(A_{cr})$ is a function returning the day part given a specific time
- h is the customer on route r visited before arriving in c

Or in words: the arrival time at a customer is the arrival time at the previous customer plus the travel time from the previous customer to the current customer in the current time period.

Given the recursive nature of this we can therefore rewrite A_{cr} to

$$A_{cr} = \sum_{(i,j) \in Arcs(c,r)} T_{ij}^{d(A_{ir})}$$

where

- the function $Arcs(c, r)$ returns the arcs taken on the route r until the customer c is reached.

Because we are interested in the reliability of a route, we need to know the mean and the variance of the arrival time of a route:

$$\mu_{cr} = E[A_{cr}] = E[A_{hr}] + E[T_{hc}^{E[A_{hr}]}]$$
$$\sigma_{cr}^2 = Var(A_{cr}) = Var(A_{hr}) + Var(T_{hc}^{E[A_{hr}]})$$

Because two independent random normal variables sum to another random normal variable: $A_{cr} \sim N(\mu_{cr}, \sigma_{cr}^2)$.

- To calculate the probability that a customer is served at a specific time on a route the probability density function (PDF) is used.
- To calculate the probability that a customer is served in a period (continuous period of time) the cumulative density function (CDF) is used, which is the integral of the PDF from $-\infty$ to a lower bound.

We need to calculate the probability of a customer being served within its time window,

- The arrival time must be smaller than the upper bound of the time window ($A_{cr} < u_c$)
- but not smaller than the lower bound ($A_{cr} < l_c$)

The probability of a customer not being served within the time window is therefore:

$$1 - (P(A_{cr} < u_c) - P(A_{cr} < l_c)) \tag{1}$$

The probability density function (PDF) in the case of a normal distribution with mean μ and deviation σ is given by

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

We need the probability that the variable takes a value less than or equal to x as we can see in equation 1, this is given by the cumulative distribution function, $F(x, \mu, \sigma)$:

$$F(x, \mu, \sigma) = \int_{-\infty}^{x} f(z, \mu, \sigma) dz$$

$$F(x, \mu, \sigma) = \Phi(\frac{x-\mu}{\sigma}) = \frac{1}{2}[1 + erf(\frac{x-\mu}{\sigma\sqrt{2}})]$$

Where $erf(x)$ is the so-called error function.

We can thus rewrite $P(A_{cr} < x)$ by its cumulative equivalent:

- $P(A_{cr} < u_c) = F(u_c, \mu_{cr}, \sigma_{cr})$,
- $P(A_{cr} < l_c) = F(l_c, \mu_{cr}, \sigma_{cr})$

This gives the probability of customer c not being served within his time window given route r

$$D_{cr} = 1 - (F(u_c, \mu_{cr}, \sigma_{cr}) - F(l_c, \mu_{cr}, \sigma_{cr}))$$

The probability of overtime is thus arriving past the end of the time window of the depot d

$$O_r = 1 - F(u_d, \mu_{dr}, \sigma_{dr})$$

5.1.3 Arrival times with waiting

For the sake of simplification we have until now considered that arriving too early was just as bad as arriving too late. In cases where a vehicle arrives too early, it is however possible to wait until the start of the lower bound of the next service window. In the next section we will see how we adjust our calculations to take into account waiting until the start of the customers time window. We will still estimate the values with a single normal distribution function in which we adjust the mean and variance for the waiting.

Adding the waiting until the lower bound of the time window if the vehicle arrives too early changes the arrival time at customer c given route r to:

$$A_{cr} = \max(A_{hr} + T_{hc}^{A_{hr}}, l_c)$$

The expected arrival time and variance given a route for that customer is now no longer given by the simple summation we had before.

Although these travel times are clearly not normal distributed, in our calculations we will just assume that the maximum of the normal variable and the constant is again normal distributed to get an good approximation. This makes it easier to reuse A_{cr} to calculate the arrival time of the next customer and the according probabilities. Ehmke et al. [5] come to the conclusion in their paper that this is indeed a good approximation of the maximum and works well in practice. This approach of approximating the max function for two normal distributed variables is common in previous literature.

We will substitute the left side of what is in the maximum function by X, which is normally distributed thus we get

- $X = A_{hr} + T_{hc}^{A_{hr}}$ with $X \sim N(\mu_1, \sigma_1)$

We will also treat the constant l_c as a normally distributed variable without any variance:

- $l_C \sim N(l_C, 0)$

Because there is no correlation between X and l_c we can calculate $E[A_{cr}]$ and $\sigma^2(A_{cr})$ easily and directly with the method of Nadarajah and Kotz (2008) [15]. Their method states that if we have two normal distributed variables, X_1 and X_2, with $X_1 \sim N(\mu_1, \sigma_1)$ and $X_2 = N(\mu_2, \sigma_2)$ and correlation ρ, then the first two moments of $X = max(X_1, X_2)$ can be calculated with the following two formulas:

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

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

Where $\theta = \sqrt{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2}$. Φ and ϕ are the CDF and PDF of the standard normal distribution, this is the case when $\mu = 0$ and $\sigma = 1$.

The formulas of Nadarajah and Kotz (2008) [15] can now be used for $A_{cr} = \max(X, l_c)$, to calculate $E[A_{cr}]$ and$E[A_{cr}^2]$ by

$$E[A_{cr}] = \mu_1\Phi(\frac{\mu_1 - l_c}{\sigma_1}) + l_c\Phi(\frac{l_c - \mu_1}{\sigma_1}) + \sigma_1\phi(\frac{\mu_1 - l_c}{\sigma_1})$$

$$E[A_{cr}^2] = (\sigma_1^2 + \mu_1^2)\Phi(\frac{\mu_1 - l_c}{\sigma_1}) + l_c^2\Phi(\frac{l_c - \mu_1}{\sigma_1}) + (\mu_1 + l_c)\sigma_1\phi(\frac{\mu_1 - l_c}{\sigma_1})$$

Note that in our case $\theta = \sigma_1$, since $\sigma_2 = 0$. Now we can use the general rule in statistics that $Var(x) = E[x^2] - E[x]^2$, to calculate the variance of the arrival time A_{cr} with

$$\sigma^2(A_{cr}) = E[A_{cr}^2] - E[A_{cr}]^2$$

With the newly calculated expectation and variance of the arrival times we can again use the cumulative distribution function for the normal distribution to calculate the probabilities for serving outside of the customers time window or having overtime on a route.

Using the CDF with the newly calculate parameters the function to calculate the probability of a customer c being served too late is now changed to no longer include the probability of being served too early.

$$D_{cr} = 1 - F(u_c, \mu_{cr}, \sigma_{cr})$$

And the probability of the a vehicle being too late at the final depot d remains

$$O_r = 1 - F(u_d, \mu_{dr}, \sigma_{dr})$$

In the next section we will consider similar calculations but then for travel times that are gamma distributed.

5.2 Gamma distribution

In this section we describe the calculation of the probability of arriving too late at a customer and the probability of having overtime for a route given that the travel times are gamma distributed. We use the parametrization with a shape parameter α and a scale parameter β. For each arc (i, j) in the graph the travel time in each time period is gamma distributed and therefore we have a shape and scale parameter for each arc in each time period.

We assume to have the same β for each arc, as this simplifies the calculations of the distribution of the sum of gamma distributed variables In the case where the data has different values for β we will adjust them to the same beta for all arcs, this is done by using a constant to maintain the distribution, its variance and mean. First we choose which β to take and then we calculate the according constant, c, by which the mean needs to be adjusted. We will do this such that the constant and the new mean always sum up to be equal to the actual mean of the travel time for that arc. Observe that for a gamma distribution $\beta = \frac{\text{var}}{\text{mean}}$. Thus the following equations show the calculation of the constant, c, we need for the distribution of one arc which has mean u and variance σ^2

$$\beta_f = \frac{\sigma^2}{\mu - c}$$

In which b_f is the chosen beta that is the same for all arcs, σ^2 and μ are the variance and mean obtained from the data. We can rearrange this equation to

$$c = \mu - \frac{\sigma^2}{\beta_f}$$

By calculating the specific c for each gamma distribution, the scale parameter can be same for every arc and every time period, β_f. c needs to always be added to the travel time of the arc. The mean of an arc is set to $\mu_{new} = \mu - c$ Which is used to calculate the new value of the shape parameter α for the distribution:

$$\alpha_{new} = \frac{\mu_{new}^2}{\sigma^2}$$

.

While we could have also chosen to adjust the variance rather than the mean, this would reduce the level of variance, which is the main complexity of our problem and is what we would like to keep at a high enough level in our model to investigate how well we can deal with it.

We use these parameters to calculate the expected travel time and travel time distribution between two customers, i and j, given a departure time d at customer i. First we do this without considering waiting before early arrivals, next we will consider waiting before early arrivals, further modifying the travel time distributions.

We can use a distribution's CDF to calculate the probabilities D_{ir} and O_i.

5.2.1 Travel times

The general rule for two gamma distributed variables $Y \sim Gamma(\alpha_1, \beta)$ and $X \sim Gamma(\alpha_2, \beta)$, is that the summation of the two is a again a gamma distributed variable with $X + Y \sim Gamma(\alpha_1 + \alpha_2, \beta)$. We will use this to calculate the travel times.

For the travel times we have the variables

- α_{ij}^p, the shape parameter for the travel time from customer i to j in period p.
- β, the scale parameter for the travel time from customer i to j in period p.
- $E[T_{ij}^p] = \alpha_{ij}^p \beta$, the expected travel time from customer i to j in period p.

Because a single trip between two customers does not necessarily fit into a single time period, similar to with the normal distributions, the travel time distributions from different periods need to be combined in a smooth way.

In case a trip has a longer expected travel time (x) than there is time left until the next period starts $(x > y)$,

- The first part of the trip will be taken using the fraction of how much can still be traveled withing the first period $(\frac{y}{x})$, for which we will will use the shape parameter of the first period. Thus for this part we have $\frac{y}{x}\alpha_{ij}^p$
- For the remainder of the trip $(\frac{x-y}{x})$ we use the scale parameter of the next period, $p + 1$.

This gives us the scale parameter for the trip crossing over period bounds:

- $\alpha_{ij}^d = \frac{y}{x}\alpha_{ij}^p + \frac{x-y}{x}\alpha_{ij}^{p+1}$

The shape parameter stays the same. Similar as with the normally distributed case it can happen that the trip does not fit into two periods, but we apply the same idea for more then two periods.

5.2.2 Arrival times without waiting

First we consider a gamma distributed arrival time where we do not adjust the distribution for that the vehicle has to wait if it arrives early. Like for the normally distributed arrival times, the arrival time is expressed as:

$$A_{cr} = A_{ir} + T_{ic}^{A_{ir}}$$

A_{cr} is gamma distributed as it is a summation of gamma distributed variables. Define that $A_{cr} \sim Gamma(\alpha_{cr}, \beta)$, where

$$\alpha_{cr} = \alpha_{ir} + \alpha_{ic}^{A_{ir}}$$

The probability density function for the gamma distribution, is defined as:

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

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

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

Where $\gamma(\alpha, \beta x) = \int_0^{\beta x} t^{\alpha-1} e^{-t} dt$ is the lower incomplete gamma function. The probability of not serving customer c on time in route r is

$$D_{cr} = 1 - (F(u_c, \alpha_{cr}, \beta) - F(l_c, \alpha_{cr}, \beta))$$

The probability of having overtime at route r is given by

$$O_r = 1 - F(u_d, \alpha_{dr}, \beta)$$

With u_d the end of a work day. In the next section we will extend this with adjusting the gamma distribution to include waiting time at a customer if a vehicle arrives to early.

5.2.3 Arrival times with waiting

In this section we will modify the gamma distributions to take into account waiting for the start of a customer's time window. The arrival times are given by the maximum function over the calculated arrival time at customer c and the lower bound of the serving window of customer c,

$$A_{cr} = \max(A_{ir} + T_{ic}^{A_{ir}}, l_c)$$

The travel times in this case are gamma distributed. To calculate the expected delay and overtime the distribution of A_{cr} should be known. The maximum of a gamma distributed variable and a constant does not return a gamma distributed arrival time, but we will treat it like it is gamma distributed to keep the calculations simple. We do approximate the shape and scale parameter of the maximum function over a gamma distributed variable and a constant.

5.2.3.1 Single distribution estimation

Given the shape parameter α and scale parameter β for a gamma distributed variable, X, we can calculate the mean and variance as follows:

$$E[X] = \alpha\beta$$

$$Var(X) = \alpha\beta^2$$

Now we want to calculate the expected value of the maximum of the gamma distributed variable X and a constant c, $Y = max(X, c)$. With $X \sim G(\alpha, \beta)$. The formula for the expected value of Y is

$$E[Y] = cP_c + \int_c^\infty zf(z)dz \tag{2}$$

P_c is the probability that X is smaller than c. This can be calculated using the cumulative density function of X. The cumulative probability function for a gamma distributed variable is given by

$$P_c = \frac{\gamma(\alpha, \frac{c}{\beta})}{\Gamma(\alpha)}$$

$f(z)$ is the probability density function for X, given by

$$f(z) = \frac{\beta^{-\alpha} z^{\alpha-1} e^{-\frac{z}{\beta}}}{\Gamma(\alpha)}$$

We have that $c \geq 0$ since a negative lower bound on the time windows is not allowed. Using these two things, we can write formula 2 as the following

$$E[Y] = c\frac{\gamma(\alpha, \frac{c}{\beta})}{\Gamma(\alpha)} + \int_c^\infty z\frac{\beta^{-\alpha} z^{\alpha-1} e^{-\frac{z}{\beta}}}{\Gamma(\alpha)}dz$$

Because of linearity we can write it as

$$E[Y] = c\frac{\gamma(\alpha, \frac{c}{\beta})}{\Gamma(\alpha)} + \frac{\beta^{-\alpha}}{\Gamma(\alpha)}\int_c^\infty z^\alpha e^{-\frac{z}{\beta}}dz \tag{3}$$

Note that $\gamma(\alpha, x)$ is the lower incomplete gamma function. $\Gamma(\alpha, x)$ is the upper incomplete gamma function and $\gamma(\alpha, x) + \Gamma(\alpha, x) = \Gamma(\alpha)$

To calculate this we need to integrate $\int z^\alpha e^{-\frac{z}{\beta}}dz$. First we apply the substitution rule, in general form the substitution rule is given by

$$\int_{\phi(a)}^{\phi(b)} f(x)dx = \int_a^b f(\phi(t))\phi'(t)dt$$

We substitute $u = \frac{z}{\beta}$. This gives $dz = \beta\, du$ and we have that $z^\alpha = \beta^\alpha u^\alpha$. Filling these in we write the integral as

$$\int_c^\infty z^\alpha e^{-\frac{z}{\beta}} dz = \beta^{\alpha+1} \int_{\frac{c}{\beta}}^\infty u^\alpha e^{-u} du \tag{4}$$

The formal definition of the upper incomplete gamma function is $\Gamma(s,x) = \int_x^\infty t^{s-1} e^{-t} dt$. This looks exactly like the integral of formula 4 if we take $s = \alpha - 1$ and $t = u$. The solution of the integral is thus

$$\beta^{\alpha+1} \int_{\frac{c}{\beta}}^\infty u^\alpha e^{-u} du = \beta^{\alpha+1} \Gamma(\alpha + 1, \frac{c}{\beta})$$

Plug this into formula 3 to get

$$E[Y] = c\frac{\gamma(\alpha, \frac{c}{\beta})}{\Gamma(\alpha)} + \frac{\beta^{-\alpha}}{\Gamma(\alpha)} \int_c^\infty z^\alpha e^{-\frac{z}{\beta}} dz \tag{5}$$

$$= c\frac{\gamma(\alpha, \frac{c}{\beta})}{\Gamma(\alpha)} + \frac{\beta^{-\alpha}}{\Gamma(\alpha)} \beta^{\alpha+1} \Gamma(\alpha + 1, \frac{c}{\beta}) \tag{6}$$

$$= c\frac{\gamma(\alpha, \frac{c}{\beta})}{\Gamma(\alpha)} + \beta\frac{\Gamma(\alpha + 1, \frac{c}{\beta})}{\Gamma(\alpha)} \tag{7}$$

Which is

$$E[Y] = c - \frac{c\Gamma(\alpha, \frac{c}{\beta})}{\Gamma(\alpha)} + \frac{\beta\Gamma(\alpha + 1, \frac{c}{\beta})}{\Gamma(\alpha)}$$

and to calculate the variance we need to calculate the expectation of Y^2, which we will do with the following formulas

$$E[Y^2] = c^2\frac{\gamma(\alpha, \frac{c}{\beta})}{\Gamma(\alpha)} + \int_c^\infty z^2 \frac{\beta^{-\alpha} z^{\alpha-1} e^{-\frac{z}{\beta}}}{\Gamma(\alpha)} dz$$

Because of linearity we write it as

$$E[Y^2] = c^2\frac{\gamma(\alpha, \frac{c}{\beta})}{\Gamma(\alpha)} + \frac{\beta^{-\alpha}}{\Gamma(\alpha)} \int_c^\infty z^{\alpha+1} e^{-\frac{z}{\beta}} dz$$

Again we use the substitution of $u = \frac{z}{\beta}$, with $dz = \beta\, du$. Substitution gives

$$E[Y^2] = c^2\frac{\gamma(\alpha, \frac{c}{\beta})}{\Gamma(\alpha)} + \frac{\beta^2}{\Gamma(\alpha)} \int_{\frac{c}{\beta}}^\infty u^{\alpha+1} e^{-u} du$$

The integral is the upper incomplete gamma function $\Gamma(\alpha + 2, \frac{c}{\beta})$, thus we get

$$E[Y^2] = c^2 + \frac{-c^2\Gamma(\alpha, \frac{c}{\beta})}{\Gamma(\alpha)} + \frac{\beta^2\Gamma(\alpha + 2, \frac{c}{\beta})}{\Gamma(\alpha)}$$

With these the formulas for $E[Y]$ and $E[Y^2]$ we can calculate the variance of Y, by the general rule for a random variable that

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

Now that we have calculated the mean and variance of Y we use them to approximate Y with a gamma distribution. This we will do by calculating an α and a β parameter for gamma distribution, for a gamma distribution the general formulas hold for calculation of the α and β from the mean and variance as follows:

$$\alpha = \frac{u^2}{variance}$$
$$\beta = \frac{variance}{u}$$

With $u = E[Y]$ and variance $= Var(Y)$. Thus we will treat Y as a gamma distributed variable, $Y \sim G(\alpha, \beta)$. The drawback of this is that the β can change as well which makes the summation of two gamma distributed variables harder. If we than want to add the next travel time to the arrival time we have to sum two gamma distributed variables, say $X \sim G(\alpha_1, \beta_1)$ and $Y \sim G(\alpha_2, \beta_2)$, they can have different values for beta. We can approximate the sum of the two gamma distributed variables by approximating the α and β of the sum. This is based on the Welch–Satterthwaite equation [16] and will be done with the following formula.

$$\alpha = \frac{(\alpha_1\beta_1 + \alpha_2\beta_2)^2}{\alpha_1^2\beta_1 + \alpha_2^2\beta_2}$$
$$\beta = \frac{\alpha_1\beta_1 + \alpha_2\beta_2}{\alpha}$$

5.3 Tree distribution approximation

Rather than estimating the distribution of maximum of the constant lower bound and some distribution with a single new distribution, we can also describe the distribution more precisely using a tree of distributions. The approach is quite flexible in that the sole requirement on making a tree with some type of distribution is that it is possible to sum two distributions (which at least applies for both the normal and gamma distributions).

For each customer a vehicle may either:

- Arrive before the customer's time window's lower bound (l_c) this means it may again leave at exactly that lower bound. This resets the variance and distribution of the time he leaves.
- Not arrive before this lower bound, which means the arrival time is described by the probability distribution of the arrival time at the previous customer, combined with the distribution for travel time towards this customer.

This means that the probability of a specific arrival time at a customer is the sum of the probabilities of each of the potential ways to reach the customer multiplied by the probability of the arrival time happening within the distribution corresponding to that way of reaching the customer.

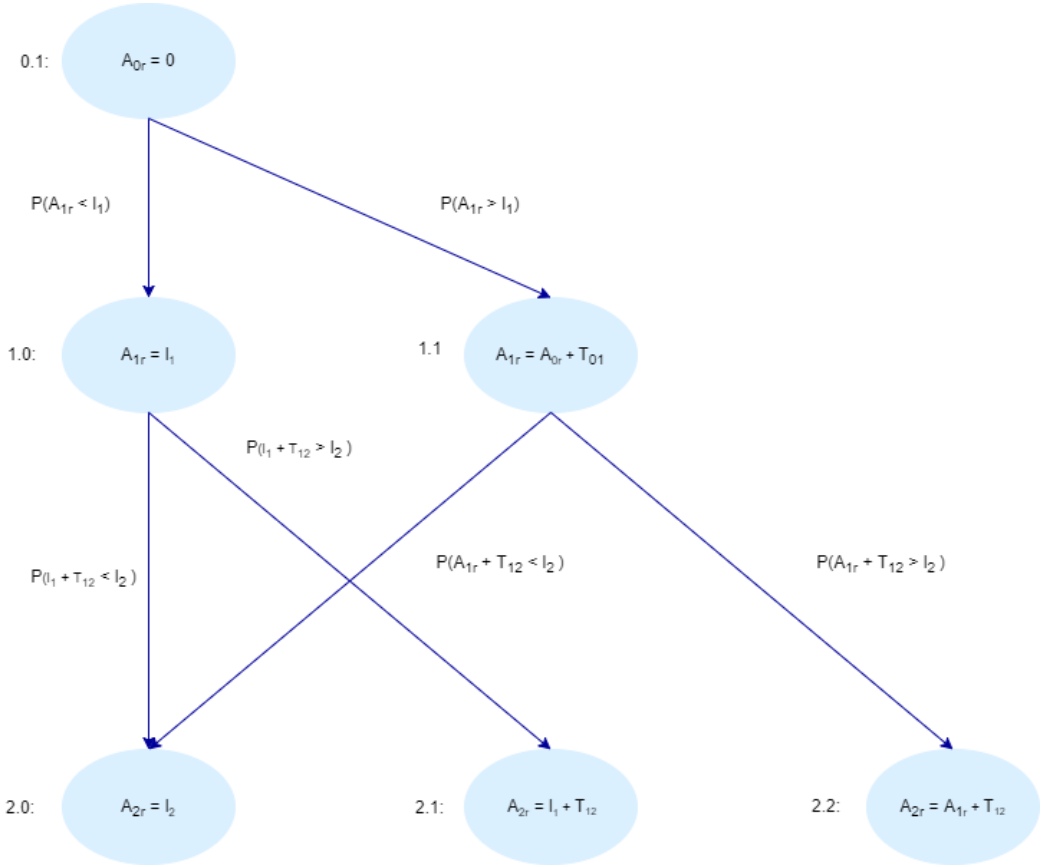

Figure 1: Systematic overview for the calculations of arrival times.

In figure 1 we can see a systematic overview of the tree. This tree can be used to calculate the probability of delay at a customer.

Some observations about this tree:

- Each row in the tree corresponds to another customer added to the route, with each node on this customer's row corresponding to a distribution function describing a path of serving the previous customers before him either before, or after their lower bounds (l_c).
- For example: the distribution in node 2:2 would describe the path where neither customer 1 or customer 2 was served before l_1 and l_2.
- For each node where a customer is served before l_c, the distribution of the probability can be set to the constant l_c (without variance). We will refer to these nodes as "simple nodes" or "left nodes" in the rest of this thesis
- For each non constant l_c node, there exists a single path from one constant l_c node towards this node, for which the distribution can be written recursively up till the constant node, for which the distribution is simple. The probability of the non constant node is therefore the probability of the simple node multiplied by the probability of the summation of distributions up to this node. We will refer to these nodes as "non simple nodes" or "right nodes" in the rest of this thesis.
- The "left node", is the absolute lowest bound on a level; none of the "right nodes" on its level is able to produce an arrival time before the "left node".
- Newer (or more left) "right nodes" should always have a lower (or at best equal) expected travel time compared to older "right nodes"; the newer nodes are built from a more recent "left node", which by definition had the lowest expected travel time on its customer level, plus the same distritions that were added to the older "right nodes'.
- The probabilities of the nodes at any level of the tree sum up to 1.0.
- For the nth customer on a route, there exist at most n different probability density functions which together describe the probability density function of this customer.
- If a customer's l_c is less than the arrival time at the previous customer then the tree does not expand at all, and at most extends each branch at the previous level. This completely rules out the possibility of arriving before l_c
- A lot of the branches of the tree will at some point fall to a probability of 0 or very close to it. A branch having no possibility of existing allows the closing of the branch, which will shrink the tree.

5.3.1 Growing the tree

For every level of the tree we can grow a next level by adding an addition customer. In this case we have to consider each leaf of the branch of the previous customer.

For the left node:

- Calculate the probabilty of the "split" between the next level's potential "left" and "new right distribution". The probability of this split is calculated from the previous level's left node for customer c_{-1}, with l_{c-1} as a constant value, which is added to the CDF of the distribution of the travel time from c_{-1} to c
- If $l_{c-1} >= l_c$ then there is no "left node" possible, and the probability of the new right node is the probability of c_{-1}
- if $l_{c-1} < l_c$ then the "new right distribution" has a probability $P_{right} = (1-$ CDF(l_c)) $P(c_{-1})$, with a distribution of the traveltime from c_{-1} to c.
- if $l_{c-1} < l_c$ then P_{left} is at least CDF(l_c)P_{c-1}
- in addition to this any probability of any of the "right nodes" arriving before l_c is added to the probability of P_{left}

For each right node:

- Calculate the split from this "right node" towards the new "left node".
- If the probability $>= 0$, then the probability is added to the one single "left node" in the row.
- The remaining probability is multiplied by this "right node's" probability, and the distribution from c_{-1} to c is added to the distribution stored for the "right node".
- The way two distributions are added to eachother depends on the type of distribution.

5.3.2 Cumulative Distribution Function

To calculate the probability of a customer c arriving within his customer window, the probability of the different branches at the level of c all need to be summed together

- The probability of the "left node" is the probability of the left node itself for any value (time) less than than the lower bound corresponding to this node's lower bound, or 0 otherwise.
- The probability of each "right node" is the CDF of its distribution multiplied by the probability of the node. If the value is less than the node's lower bound, then the node's lower bound is used in the CDF of the distribution. If the value is bigger, then the parameter is used in the CDF of the distribution.
- For each of the nodes the node's probability can be computed from the previous node, which means it can be done while building up a route. This means that as soon as a customer is added which makes the route infeasible, the route can be discarded.

5.3.3 Expected arrival time

To calculate the expected arrival time at a customer c, the expected arrival time of each of the brances at the row of c is multiplied by the probability of the brance. The total expected arrival time is the sum of these arrival times.

While this tree method is more complex and requires more storage space, by being able to prune many branches of the tree early on, the actual costs is likely rarely near the worst case where n distributions need to be stored for a route of length n.

Moreover, although the storage space and computing time increase in the worst case only linearly in the route length, typical routes are rarely very long. This makes it likely that the potential increase in precision worth the slight increase in storage and running time.

Because the only requirement for a distribution in the tree is that they need to be able to be summed, we coul make a tree for the normal, the gamma, the max normal and the max gamma distribution, as well as any other potential future distributions.

6 Column Generation

To solve our main linear programming problem we will use column generation, this is a technique to solve large linear programs. The idea is that linear programs are often too large to consider all variables explicitly. There are too many potential variable combinations, with only a small subset of the variables ending with a non zero value in the optimal solution. Column generation reduces this number, by only considering the variables which have the potential to improve the object function.

With column generation the problem is split into the following (iterative) steps:

- First the restricted master problem (RMP) is defined as a set partitioning problem. The RMP only considers routes which were added to the set because they were considered to have potential. This has the advantage that not all possible routes need to be enumerated, but only those of high potential. We choose set partitioning because of its flexibility with constraints and the objective function. It is more flexible than other approaches because it allows adding constraints affecting a single route more easily. Solving the RMP assigns a dual value for each customer. This value corresponds to how much the total solution could be improved by adding alternative better routes including this customer.
- The pricing problem is a sub-problem that follows from the RMP to identify new variables - in our case routes - that can enter the set of variables for the RMP. RMP assigns these so called "dual values" to customers, which help identify new routes involving the customers that have most potential to improve the total solution. These new routes which are found by solving the pricing problem are then added to the RMP, which can then be solved to again find new dual values.

  During our experiments we will try to solve the pricing problem using different techniques.
- Finally we consider using post optimization methods to improve the obtained solution.

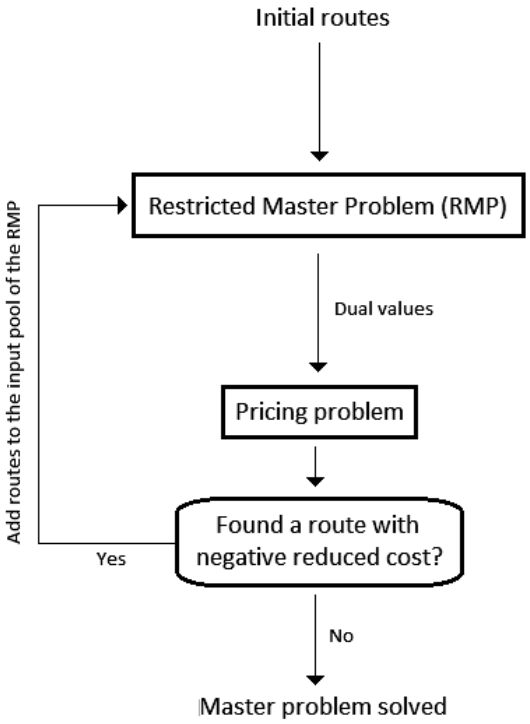

Figure 2: Overview of the column generation procedure

6.1 Restricted Master Problem

The objective function is rewritten to a set partitioning problem, for which the following variables are introduced:

$$x_r = \begin{Bmatrix} 1 & \text{if route r is chosen} \\ 0 & \text{otherwise} \end{Bmatrix}$$

$$a_{ir} = \begin{Bmatrix} 1 & \text{if route r visits customer i} \\ 0 & \text{otherwise} \end{Bmatrix}$$

C_r is the total costs of driving a route, including the fixed cost, the penalties for early and late arrivals and the cost of overtime. The objective is formulated as

$$\min \sum_{r \in R} C_r x_r \tag{8}$$

Subject to

$$\sum_{r \in R} a_{ir} x_r = 1, \forall i \in N \tag{9}$$

$$x_r \in \{0,1\}, \forall r \in R \tag{10}$$

The idea is to pick a set of routes from a restricted set of routes (R), which will be produced by the second step, the pricing problem, that serves all customers and

minimizes the objective function. Constraint (9) ensures that each customer is visited exactly once. In the final solution we can not include a partial route, so x_r should either be zero or one, therefore constraint (10) is added. Because the pricing problem will only add feasible routes subject to the constraints of section 4, the solution to the RMP will also satisfy these constraints on all of the routes.

6.2 Relaxed Master Problem

In the column generation approach the constraints of the RMP are relaxed which helps with finding the dual values needed to identify what kind of new routes should be included in the set R to find a better solution for both the restricted and relaxed master problem. The relaxed master problem is defined as

$$\min \sum_{r \in R} C_r x_r \tag{11}$$

Subject to

$$\sum_{r \in R} a_{ir} x_r \geq 1, \qquad \forall i \in N \tag{12}$$

$$x_r \geq 0, \qquad \forall r \in R \tag{13}$$

Thus the solution for the RMP allows partial routes.

In the first iteration of solving the RMP and pricing problem only some initial simple routes are included, the initial routes make sure that there is at least a feasible solution of the RRMP since each customer is included on at least one route. Solving the restricted relaxed master problem (RRMP) gives dual values for each of the constraints corresponding with specific customers which are then used to solve the pricing subproblem to find better routes. If in the pricing problem a new route is found with negative reduced cost, the RRMP solution will be improved by the route and it is added to the the RRMP. After solving the pricing problem the RRMP can be solved again to obtain new dual values. Once the pricing problem can no longer find any routes with negative reduced cost, then we have found the optimal solution to the RRMP.

6.3 Calculation of the reduced route cost

In the pricing problem we want to find the route which reduces the cost most, since this route is most likely to improve the RRMP solution.

It is important that any added routes satisfy the constraints defined in section 4. The routes should start and end at the depot but not visit the depot in the mean time and keep the conservation of flow. By only considering and therefore adding routes which are feasible, the solution to our RRMP will also be a feasible solution

that satisfies all of the constraints.

The objective function for the pricing problem is defined as

$$\min_{r \in R} \hat{C}_r \tag{14}$$

$\hat{C}_r$ is the reduced cost of route r and is calculated as

$$\hat{C}_r = C_r - \sum_{i \in N \setminus 0} a_{ir} \pi_i \tag{15}$$

Where $\sum_{i \in N \setminus 0} a_{ir} \pi_i$ sums all dual values (π_i) of the customers on the route r and

$$C_r = \sum_{i \in N} \sum_{j \in N} x_{ijr} d_{ij} + \sum_{i \in N} c_d D_{ir} + \sum_{i \in N} c_e E_{ir} + c_o O_r + c_f \tag{16}$$

which was described in detail in chapters 4 and 5.

While the most negative solution is most likely to improve the RRMP's solution most, any route with a negative reduced cost is already a good candidate to improve the solution.

6.4 Initialization

In the initialization we generate one route for each of the customers, containing solely this single customer. These routes are the initial input of the master problem which makes sure that there is always a feasible solution.

6.5 Pricing problem

To solve the pricing problem we will use a local search and a dynamic programming approach. In Lekkerkerker et al. [13] a mixed integer programming and an incremental mixed integer programming method were also described and tested on small instances. However as Lekkerkerker et al. [13] already concluded that both methods already performed poorly on small instances, because of the number of variables, we will not consider these methods.

6.5.1 Pricing using Local Search

Local Search is a relatively easy heuristic method to solve difficult optimization problems. In the pricing problem we try to find a solution that minimizes the reduced cost. Local search algorithms start with an initial solution which is improved by looking at neighboring solutions, which can be reached by applying local changes like adding or removing customers from the route.

The number of iterations must be big enough to make that the solution converges, because when only local changes are considered the method is likely to get stuck in a local optimum, this is why different starting points are considered to increase the odds of finding a global optimum.

Because the goal of the pricing problem is to find candidate routes that will be part of a bigger solution, even if only local optimums are found they can already very useful towards improving the RMP's solution.

We use the hill climbing technique, every iteration we change an element of the best solution so far and if this change is better than the previous best solution we accept it as the new best solution. We use different methods to go to a neighbor solution, adding or removing customers in an increment, applying 2-opt or do an improvement based on finding the best position for a random customer. This is described further in section 6.5.3.

6.5.2 Initial solution

As the initial solution to the pricing problem with local search we will use the approach described in Solomon [17] this is a Sequential Insertion Heuristic. Solomon concluded that this method from the 5 heuristics he evaluated proved to be the best in quality and in required computation time to find a good initial solution.

The approach is modified to find a good starting solution for our variation of the problem. We build the solution route for route. First we start by finding the customer with the highest dual value. This value gives the cost of meeting the minimum requirement for that customer. Thus finding a new route for this customer is most likely to improve the solution. This customer is added to a new route.

Subsequently the method iterates over the remaining customers, checking for each customer what the cheapest feasible position in the route would be. A position is only feasible for a customer when the probability that the customer and all subsequent customers on that route are visited on time at that route and position is above a threshold value and when the probability of having overtime for the route is staying below the threshold value. If there is no feasible position for a customer on this route this customer can not be placed into the route and will be set aside for now.

After determining the best position, if any, for each customer the algorithm finds the best customer to be inserted in the route. This is the customer for which the route costs are the lowest after inserting. We thus have checked for each customer checked for the best position to be inserted and then which customer is the best to add on the route according to its best position.

We continue until there are no more feasible customers to add to this route. Then with the customers that were unfeasible and thus not yet placed we repeat these steps until all customers are placed on a route. If all customers are placed on a route the initial solution is finished and we have obtained a set of routes.

6.5.3 Solution improvement operators

In the previous section we obtained an initial solution which consists out of a set of routes. For each of these routes a local search algorithm is run to improve the routes.

Given a single route r out of the set of routes it is set to be the current best route with a reduced cost $\hat{c}_r$. Next a neighboring solution r' is selected with one of local operators further specified below. For r' the reduced cost is calculated, $\hat{c}_{r'}$. If $\hat{c}_{r'} < \hat{c}_r$ then it is an improvement and r' will replace r as the current best route, this method is called hill climber. This is done until a termination criteria is met, which can either be a time limit or he current best solution did not change for a specific number of iterations.

The following operators are used:

- Randomly add a certain number of customers that are not in the route yet at random positions.
- Randomly remove a certain number of customers from the current route.
- 2-Opt; reorders the customer on a route. This could for example swap the edges (c_i, c_{i+1}) and (c_j, c_{j+1}) for the edges (c_i, c_j) and (c_{i+1}, c_{j+1}).
- Random Customer Repositioning; we select a customer at random and find the best place for this customer in the route.

6.6 Pricing problem with Dynamic Programming

A dynamic programming approach based on Held and Karp [9] is used to solve the shortest path problem to find the route with the least reduced cost. The approach enumerates all the possible solutions in a clever way and returns a feasible route with the least reduced cost. The method solves the problem by breaking it down in small subproblems and for each of the subproblems the solution is stored so that it does not need to be recomputed. This leads to less computation time but an increased use of storage space.

By reducing a problem to reusable subproblems and saving the solutions to these subproblems, many otherwise recursive calls can be looked up in constant time. The solution is build bottom up, we start with the smallest subproblems, routes with one customer, and extend these by adding customers one by one. The subproblems are called a state, in each state we store the following information:

- The set of customers visited in that state so far (in no particular order).
- Last customer visited in the state.
- Reduced cost of the cheapest route that visits all the customers in the state and ends in the last customer.
- The travel time distribution for the sum of the travel times for the best route visiting all customers in the state and ending in the last customer. This contains the mean of the distribution which is used as departure time from the last customer.

6.6.1 Expanding states

The algorithm starts with defining basic states which are states that correspond to a route with only one customer c. Next we create a next layer of all routes of length two by creating a new state for all unvisited neighbors next to each basic state. In each new state, an unvisited customer is added to the set of visited customers of the new state, with the unvisited customer being added as the last visited customer. This process can be repeated for n levels bottom up, where n is the maximum length of a route and will exhaustively compute every potential route of length up to n.

At a first glance this may seem inefficient, but because the order in which previous customers are visited does not matter, but merely what the minimum cost is of visiting them, many potential routes share the same previous customers and therefore states. This means that we can do a dictionary lookup for the minimum cost as soon as it was computed once for a state.

Once we have calculated n levels bottom up, we can select the b number of best routes of each length that have the most negative reduced cost and are feasible. We do not only add the routes of maximum length that may reduce the reduces cost

most, because once the RMP calculates an integer solution it often requires some "filler routes" of lesser size to get an optimal a good solution.

6.6.2 Arrival times

An unfortunate aspect of our problem is that we do not merely have to deal with the minimum cost, but also with arrival time; at certain arrival times, certain customers can no longer be visited because of time windows. Because of this it is possible that some state has a minimum cost associated with an arrival time very late into its last customer's time window. The result is that this state can only be expanded towards customers that have an upper bound on their time window that is even later, thus not being able to reach a number of customers that would have resulted in a lower cost down the road.

As a potential solution to this problem we may divide a customers time window into a fixed number of slices. Each slice represents an arrival time into the nth part of the customer time window. This makes it possible to also save a worse minimum cost at a customer, as long as it has an arrival time in an earlier time window slice than the state with a better minimum cost. This could partially prevent the previously stated problem, but has the downside that generally the main bottleneck of Dynamic Programming is memory space, by saving a fixed number of window slices, this becomes even worse. Regardless it might potentially improve some routes.

6.6.3 Delayed travel time costs

Another difficult aspect we may have to deal with are delayed costs: If we choose a route with a high variance, this does not just impact the cost right now, but the cost for every customer that is added later on, as it still has this increased uncertainty in travel time. What makes this difficult, is that it is not known before hand how many customers the added uncertainty affects, as the ending length of a route is not known at the time of adding a customer.

Again a potential solution, is to make the desired route length part of the state, the cost can then be added based on the desired route length. In the end only states where the desired route length matches the actual length should be considered candidate routes. Again the downside is the memory bottleneck would in most cases make this impossible.

6.6.4 Pruning routes

Because we keep track of the travel time distribution while maintaining the minimum cost, when adding a new customer we can already calculate the probability of being later than the customer's time window upper bound, potentially rejecting the customer for making the route unfeasible, if the probability is bigger than some chosen hard constraint.

6.6.5 Reducing memory problems

To deal with memory problems, rather than expanding every state at each level, we can only expand the m states with the lowest reduced cost. To prevent that some far away customers are largely excluded, we will use a variant of this where we expand for each customer the m states of lowest cost that include the customer. This makes sure that we do not look too narrowly and end with very local maximums.

To further deal with memory problems, in a lot of cases every customer is connected to each other customer. This results in a large number of states that have a very low likelihood of resulting in a good route. To deal with this we may take only the k nearest neighbors as actual neighbors.

6.7 Column management method

The columns grow very rapidly during the column generation approach, which then leads to an increasing compution time of the master problem. Therefore we will apply column management to limit the time increase of the master problem.

First we can limit the number of columns in the input pool by a fixed number. We only use the n columns with the most negative reduced cost. Where n is a chosen value. Secondly we check each time we want to add a route if it is not a duplicate.

7 Restricted Integer Linear Program

After having solved the master problem to optimality we have a solution for the relaxation. This means that the solution can be fractional, thus that a route is partially included in the solution. Of course we can not select partial routes in the final solution, so we need to convert the solution of the master problem to an integer solution. One common approach is to make use of branch-and-price, but this is normally computationally expensive. Another option is that if we have found the solution of the master problem we can use the LP-Solver again using the same columns as generated with the column generation process, but than restrict the decision variables to being integers. This however gives a solution from low quality.

Therefore we use the method of Diepen et al. [4] to get an approximation solution. This method uses second-choice columns. The columns generated in the original pricing problem are the first-choice columns. Now we decide to create a set of additional columns each time we solve the pricing problem. When the pricing problem is solved we obtain a solution, a route with customers, we then take out the customers one by one and solve the pricing problem again. The extra generated columns are added to a separate column pool. If the master problem is solved optimal we filter this pool such that it exists out of unique routes. These routes are than added to the input pool of the master problem and with all these columns together we solve the problem where the decisions variables are redistricted to be integers with the LP-Solver.

Another method is that during the pricing problem we do not only select columns with a strictly negative reduced cost ≤ 0, but we add same space to create second rank columns by selecting columns with a reduced cost $\leq \delta$ where δ is a threshold value. Each time the pricing problem is solved we add the columns r with $\hat{C}_r < 0$ to the column pool of the master problem and the columns with $\hat{C}_r \in (0, \delta)$ are added to a second rank column pool. As soon as the master problem is solved to optimality we add these column to the input pool and solve the problem as a Integer Linear Program problem.

8 Data

To analyze the performance of the model, multiple experiments are performed with existing data sets. The experiments are conducted on the data sets used in Lekkerkerker et al. [13]. Lekkerkerker et al. [13] got real life data from the company Veldhuizen. The data consists of a number of travel periods, a number of customers with their time window, so a lower bound and an upper bound for servicing. The travel cost between each two customers and from and to each customer from the depot. And finally the travel time between each two customers for each given time period for 100 sample worlds.

The data Lekkerkerker et al. [13] obtained from the company Veldhuizen consisted of two test cases, one with data from a particular day in 2015 and one with data at a particular day in 2016. The average size of time windows in these test cases where 6h30m and 7h17m respectively. Since the influence the size of the time windows might have on the performance of the methods, quality of the solution and running time, Lekkerkerker et al. made variations on the size of the time windows. For both test cases the time windows were reduced to sizes of 10, 30, 60, 120 and 240 minutes. We obtained these test cases and use them for the experiments.

There are data sets indicated as tiny(T), small(S), medium(M) and large(L) according to the number of customers in the data set. In the tiny data set there are around 10 customers, in the large according to 100 customers. The data obtained is from two days 18 September 2015 and 12 January 2016 which we refer to as V2015 and v2016 respectively. And for example V2015-M T60 refers to the dataset from 2015 from medium size and with time windows of 60 minutes for each customer.

We want to work with a stochastic objective function, but the travel times of the data from Veldhuizen do not follow a specific distribution with any significants. Therefore we take the mean and standard deviation of the travel times for each of the arcs in each period to estimate the parameters. To test the method we then generate our own samples to test on. We create samples for the normal distribution and for the gamma distribution by drawing random numbers according to these distributions and their parameters.

9 Implementation

For this thesis we have built a framework in Java to test the different distribution methods with both of the pricing problems. For the formulated linear programming problems, we use Gurobi [8] optimization to find the solution and the corresponding dual variables of the LP.

The framework was set up in a way that makes it easy to swap between distribution functions. Any distribution needs to just satisfy the properties that it can be added to another distribution, that is has an expected value and that a CDF can be calculated. For each of the distributions described in 5 a class is implemented can be used by the framework.

For the implementation of the DP we added options for:

- When exploring states, to at least explore the n most promising states for each customer, that involve that customer.
- The maximum number of potential routes to add to the column generation method of each length.
- The minimum reduced cost threshold the routes need to satisfy to be considered a potential solution route.
- The number of nearest neighbors of a customer that need to be considered. As regular DP is an exhaustive method, pruning the most unlikely customers greatly reduces computation time.

For the implementation of the local search we added options for:

- The number of iterations of the local search.
- The number of iterations a single route will be tried to be improved with local operations.
- The minimum reduced cost which the routes need to satisfy to enter the column pool.

To test how well our solutions perform a simulated ground truth is constructed:

- For each arc between customers for every period of the day, the arc is sampled individually.
- Using these samples, the actual cost of the tested solution is computed.
- This process is repeated n times to get an average of the solution's actual cost.

The implementation reads a problem description from a file, computes a solution to the problem using the selected settings and finally tests this solution against the simulated ground truth.

10 Experiments

In this section we will present our experiments. We had a number of things to investigate:

For each of the presented probability distributions:

- the accuracy of predicted lateness, thus how well each distribution predicts the actual overtime.
- the robustness to high variance in the data
- the robustness to poor data fitting
- the computation time

To investigate these characteristics we will use the dynamic programming version of our pricing problem as some pre-experimental tests seemed to indicate a better performance and computational speed. Regardless the version of the pricing problem should not affect the results we find regarding these distributions.

Once these characteristics have been established we will investigate the influence of our choice of pricing problem for the column generation. For this we will use both our local search and dynamic programming approach.

We run the experiments using our own implementation. Every computation is performed on a computer with an Intel(R) Core(TM) i7-6700HQ (2.60 GHz) 16 GB Ram.

10.1 Experimental set up

Our experiments consist of the following phases:

- A route generating phase, where we run our column generation program with some specific settings, this generates a best found solution consisting of a number of routes.
- Each route in the solution is assigned a "predicted objective cost", which ideally matches the actual objective cost as close as possible, as incorrect evaluations of the objective cost make finding the best or even a good solution impossible.
- A testing phase, where we test the returned solution by traversing each route, sampling each individual distribution T_{ij}^p on the route and thus simulating performing each route in a test world. T_{ij}^p is the distribution of the arc from customer i to customer j in time period p. In case we arrive at a customer before the start of his time window, we continue the route from the lower bound of its time window. To get a good average, We repeat this process for each solution 10000 times and return the average cost. The average traveling cost

and the cost of penalties for being late and we return the average percentage of customers served on time in one simulation.

- Finally we compare the predicted solution to the actual solution and come to some conclusions.

10.2 Probability Distributions

The most important aspect which the probability distributions need to satisfy is an accurate prediction of the service costs for a given route; If the predicted service costs are too low and the route is chosen, then the actual objective cost may explode to multiple times the predicted cost. However if it overestimates the service cost, an inefficient route or an additional vehicle may be used to unnecessarily try to decrease the service costs. On top of that it may completely disregard routes that currently only have little potential, that might later prove to be part of an optimal route.

To test the accuracy of each of the distributions we performed the following experiment: We run our column generation program on the "small dataset" using dynamic programming to obtain a solution for each of following distributions:

- The Simple distribution. A deterministic method that completely disregards any potential variance. This will be used as a base case to show what happens if we do nothing to predict lateness, or do it very poorly.
- The Normal distribution. The regular normal distribution with a mean at least as big as the lower bound of the last customers it has passed (to never serve customers before their serving window's lower bound).
- The Gamma distribution. A gamma distribution plus a constant, for which the μ plus the constant is at least as big as the lower bound of each of the customers that is has passed
- The MaxNormal distribution. The normal distribution that is updated to approximate the distribution of the maximum of itself and the lower bound of each of the customers that it passes.
- The NormalTree distribution. A tree structure consisting of a constant and multiple distributions, each with their own probability, Uses normal distributions for the distributions.
- The GammaTree distribution. The same tree structure using gamma distributions.
- The MaxNormalTree distribution. The same tree structure using max normal distributions.

We will choose the following values for the objective function $c_d = 10000$, $c_o = 10000$, $c_f = 50$ $c_e = 0$, by prioritizing lateness and overtime, we should get a good idea of how well each distribution estimates the probabilities associated with these

parameters.

After calculating a solution for each of the distributions, each of the other distributions will evaluate the routes of the other distributions. We do this to check whether the distributions judge the performance of not chosen solutions correctly. If some type of distribution would only evaluate some types of routes correctly as good routes, then it might incorrectly disregard actually better solutions. By letting the distributions cross evaluate each other we can get a good idea of whether this is the case.

This cross evaluation gives a predicted objective value for every distribution for every found solution. Next, we will simulate 10000 test worlds where each edge is sampled individually to simulate traversing the route, We will do this once using normal distributions, and once using gamma distributions, since the different methods are based on travel times that are normal distributed or travel times that are gamma distributed.

Additionally, as we would like to see how different levels of variance affects the predictive rates of the methods, we will run the experiment several times on the same dataset, where we however multiply the standard deviation by a different number each time.

10.2.1 Max Gamma and Max Gamma Tree

Despite the theoretical support for the max gamma distribution approach, it was not practically possible for this thesis to implement a satisfactory version into the column generation frame work. Due to the extensive calculation with the gamma functions with both very large and small values of α and β the size of the Java primitives easily overflowed or turned into (very near) 0 values.

In a following attempt BigDecimal was used for the calculations of the max gamma but, however less often, it still resulted in the same problems. Attempts at shutting down "infeasible values" resulted in very poor results (seemingly incorrect), that still took much more time than any of the other distributions. This is why we have decided not to include these distributions into the experiment.

10.2.2 Prediction service cost per distribution method

In table 1 we see the results for each of the solutions generated by different stochastic methods. The title of each of the charts tells which distribution was used to find the solution. The x-axis indicates how many times the regular standard deviation was multiplied (1, 2, 3, 5, 10, 20).

The y-axis shows the service cost of a route on a logarithmic scale. This is done because especially when the service cost is low, it is important to be accurate. A

route of high cost is rejected regardless of exactly how high its cost is; any route of lower cost will be used instead. Which is why it is especially important to valuate low cost routes accurately, it has more potential to be chosen as the best route.

Mostly overlapping are the Gamma- (dark red) and Normal- (dark blue) results from the simulated worlds. While we will not present a quantitative measure, a distribution can be considered a better predictor if its line is closer to the normal- or gamma result lines, as we consider these the true objective values.

Some observations that we can make here:

- The routes generated by using the Simple distribution are judged as "high service cost" across the board.
- The GammaTree (blue-purple) seems to be the best predictor, following the actual results closely in most cases.
- The MaxNormalTree (orange) also seems to be a good predictor, although it fails to accurately classify some of the routes that both the Gamma distributions correctly classify as good, it still performs well on all the other cases.
- The Gamma (light blue) and Normal (red-purple) distributions seem to be the worst predictors, way overestimating the service costs especially when the actual service costs are very small.
- The MaxNormal(dark green) and the NormalTree (light green) seem to perform neither very good nor very bad.
- Overall, even for large standard deviations most stochastic methods seem to do well.

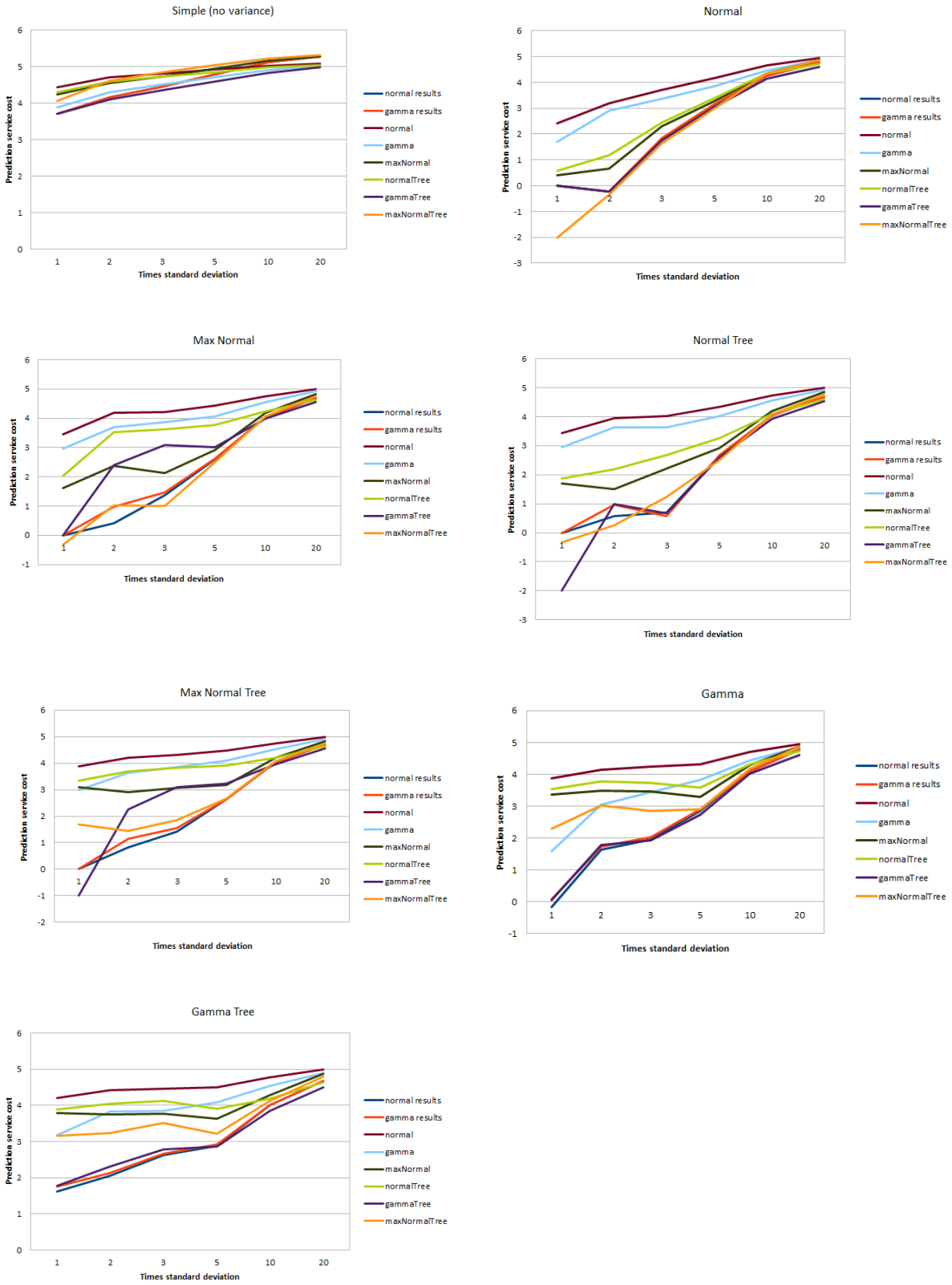

Table 1: In each chart we have on the x-axis the standard deviation multiplication factor. On the y-as we have the prediction of the service cost on a logarithmic scale. A closer line distance to the normal- and gamma results indicates better predictive power. All results were obtained from the small test case.

10.3 Prediction versus actual performance - small dataset

Now that we have some indication that some distributions are indeed better at service cost prediction, we will perform an experiment to establish how well the distributions do at finding solutions of low objective cost. Moreover, we will investigate how increasing the standard deviation affects the ability to find good solutions.

- We will again solve the small dataset using the same parameters used in the previous experiment.
- We will let each distribution find its best solution.
- We will multiply the standard deviation by (1, 2, 3, 5, 10, 20)
- We test the solution against 10000 simulated worlds, using both normal and gamma distributions.
- We will also test the 1 standard deviation against 10000 simulated worlds, where we uniformly sample from the 100 travel times that were provided in the test set.

In table 2 the results of the experiment are shown, in the title of each chart the multiplication factor used to multiply the standard deviation is shown. On the y-axis the objective value is shown (linearly). Blue bars indicate the predicted objective value, with the other bars showing the results of the simulated worlds.

Some observations about the results:

- For the 1 standard deviation, the Simple distribution manages to barely get the lowest predicted objective score. The GammaTree despite having to take service costs into account is however very close to its predicted score.
- When we look at the actual objective values, it becomes very apparent that the Simple distribution will not work well for the given parameter settings. The GammaTree on the other hand, seems to have actual values very close to what was predicted (with the exception of the uniformly sampled objective function).
- As we increase the standard deviation, it becomes increasingly more apparent that the normal and gamma functions way overestimate the service costs. In the case of the normal distribution this also leads to increasingly worse results in the actual objective functions compared to the other distributions. In the case of the gamma function this effect is less apparent.
- For the 10 and 20 standard deviations we solely show the best performers, MaxNormal, GammaTree and MaxNormalTree to highlight the differences: where the MaxNormal and MaxNormalTree seem to start overestimating the service costs, the GammaTree actually starts underestimating it.
- Overall, it seems like the GammaTree finds the best solutions for different standard deviations, followed by the MaxNormalTree and the MaxNormal.

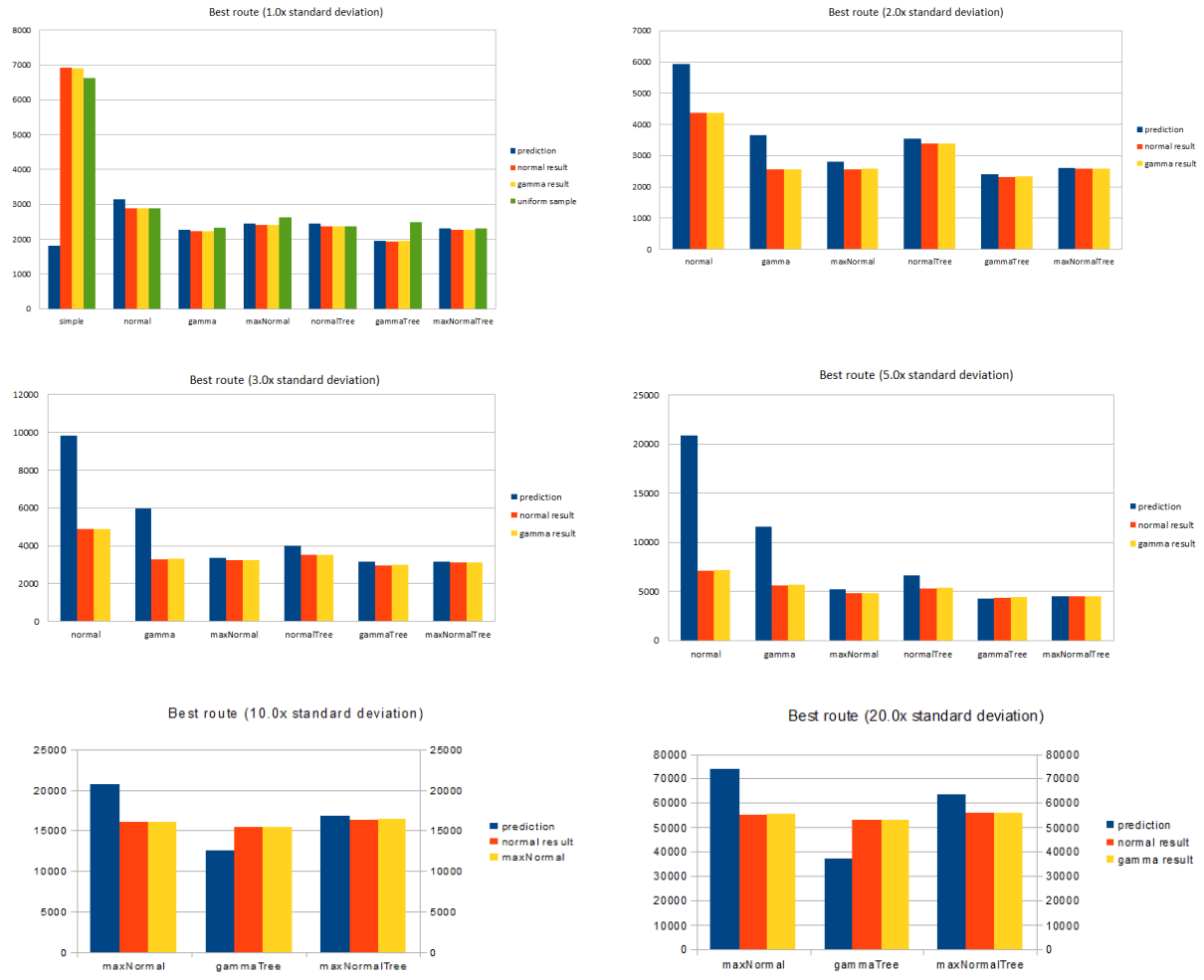

Table 2: The prediction of the objective values and the actual results for the different distribution methods. On the y-axis is the objective value. Tested on the dataset with 26 customers.

10.4 Prediction versus actual performance - big dataset

In this section we will repeat the previous experiment on the large dataset with 100 customers. To reduce the running time and memory complexity a bit, we use the following dynamic programming parameters:

- We only consider the edges to the 7 nearest neighbors for each customer.
- We expand only the 40 most promising states for each customer, for each route length.

The results are very similar as the results for the small dataset which further confirms our previous observations, and were likely not a fluke resulting from using a specific dataset.

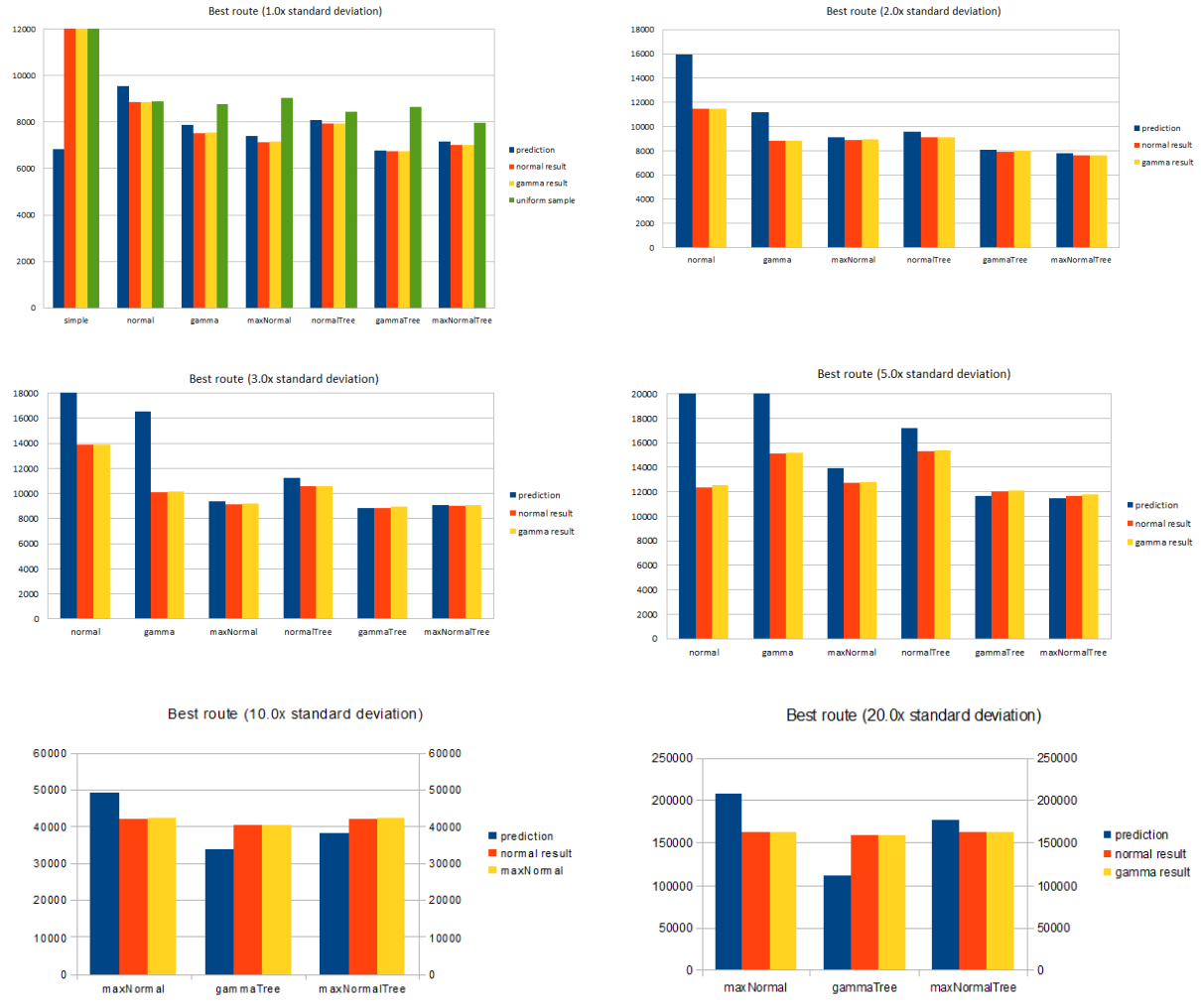

Table 3: The prediction of the objective values and the actual results for the different distribution methods. On the y-axis is the objective value. Tested on the dataset with 100 customers.

10.4.1 Performance different actual standard deviation

Now that we have an idea how the distributions rank up against each other in terms of accuracy assuming we know the standard deviation, we will perform an experiment to see how the distributions perform when the distribution is unknown and we may estimate the standard deviation incorrectly.

- We again test on the large test set with the same parameters.
- For each distribution we calculate a solution
- We test this solution against a simulated world using Normal distributions, however these test standard deviations are multiplied by some factor (0.0, 0.25, 0.5, 0.8, 1.0, 1.1, 1.2, 1.3,1.5, 2.0, 3.0, 5.0).
- Because the normal and gamma function have proved to be almost identical for testing in previous experiments, we will only use the normal distribution.

In table 4 we can see the results of this experiment. All charts have the same y-scale cutting off most of the higher values.

Some observations:

- In the first chart we see the simple method, and as expected, without any standard deviation it performs perfectly, at the introduction of the slightest bit of standard deviation it however falls directly off the charts.
- The Normal function performs better, but by far the worst out of the remaining distributions.
- The better the "base line' (0 standard deviation, objective function) performs, the less room there seems to be for unaccounted standard deviation. Especially the MaxNormalTree and GammaTree rise rapidly as soon as the standard deviation approaches 1.0.
- Regardless, the MaxNormalTree and GammaTree still perform best of all distributions up till a standard deviation multiplied between 1.1 and 1.2.

Table 4: Performance of the methods when the actual standard deviation is higher than the standard deviation used to predict the routes. The x-axis states how many times higher. Results obtained from the big test case.

10.5 Comparing pricing problems

Thus far we performed the tests with the dynamic programming approach for the pricing problem. However we also implemented the local search approach and in this section we will compare the two methods. First we start with finding the proper settings for the dynamic programming and the local search approach. Next, we follow this up with a comparison in running time and accuracy for the two methods.

10.5.1 Settings Dynamic programming

In the implementation of the dynamic programming a few settings are required. The following variables should be set

- Number of nearest neighbors, only these neigbors will be considered to create a next state with one customer added to the current state.
- The number of states that we at least expand per customer when we extend our route by another level.
- The maximum reduced cost for which we still want to add a new route. By adding routes that may not have a negative reduced cost, the best found solution may still improve, because it can help bridge the gap between the RRMP and the RMP. If no new, better routes are found during an iteration, the DP algorithm stops.

The settings are examined using the GammaTree method, as it has proven to be at the very least one of the most accurate distributions, for almost every setting, regardless of the pricing problem.

In figure 3 the objective value of the relaxed restricted master problem is plotted against the number of iterations. The number of iterations is how often the pricing problem, followed by the rrmp are iteratively solved. After 40 iterations the objective value of the RRMP does not seem to improve much anymore. The corresponding version of the restricted master problem is show in 4 which also shows to be converging after roughly 50 iterations.

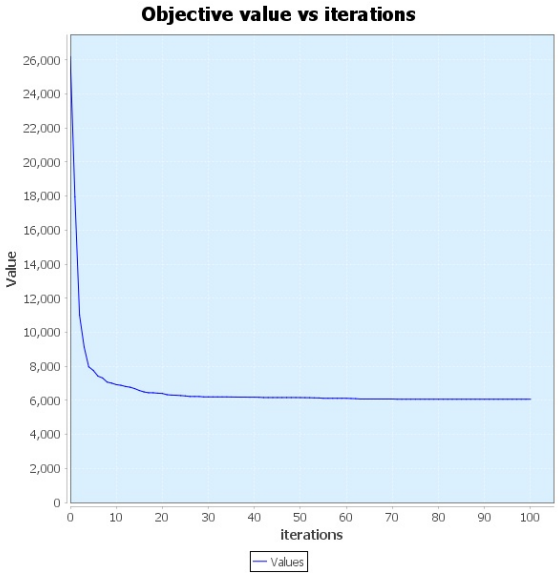

Figure 3: Value of RRMP

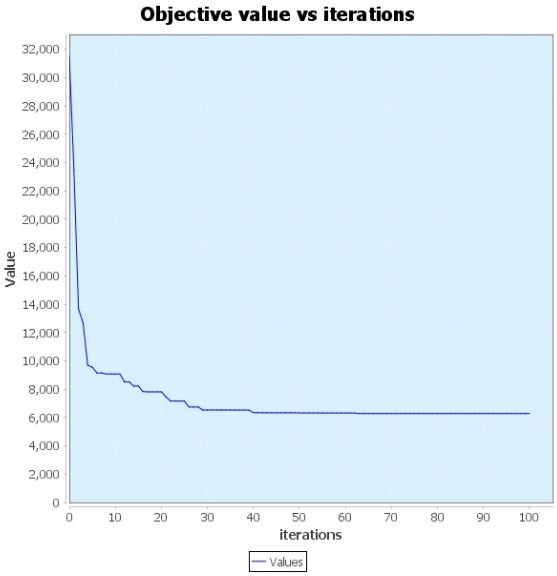

Figure 4: Value RMP

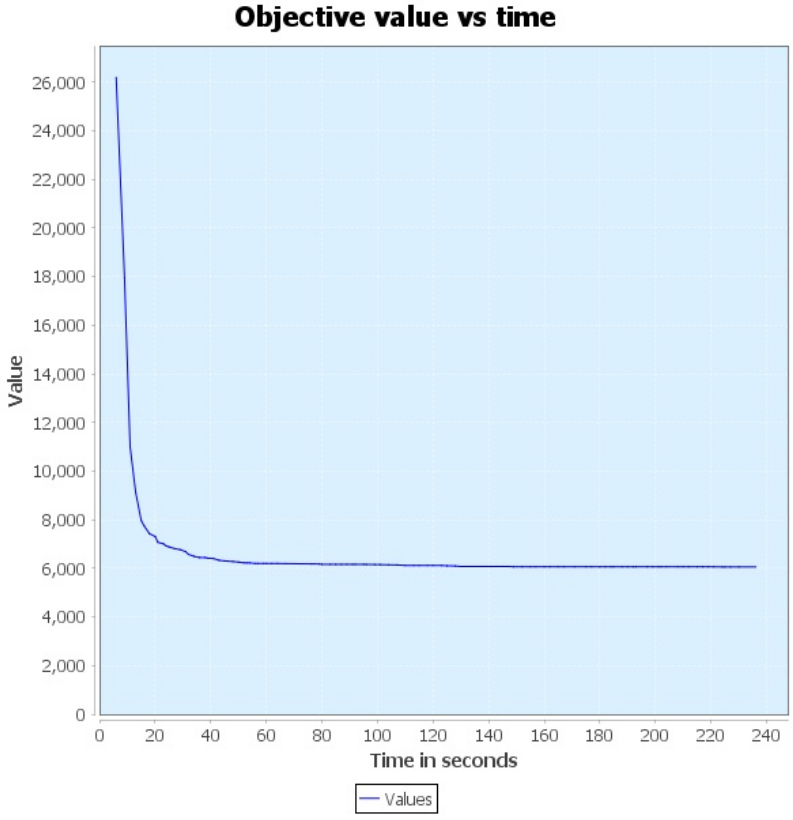

Figure 5: Calculation time versus RRMP objective value

In the next experiment we vary the number of closest neighbors and present the results in table 5. The results clearly show that the CPU processing time increases rapidly by every extra nearest neighbor we consider. Initially the objective value cost decreases steadily, but after roughly 9 neighbors the objective function starts

rising again. This is due to the number of states considered to be explored. If we consider a lot of nearest neighbors but the number of states is limited, some neighbors that would later prove valuable are cut early in favor of a different neighbor.

While the effects of eliminating a number of neighbors likely strongly depends on the specific problem instance, preprocessing a graph to eliminate highly unlikely candidates is a good idea to reduce computing time, allowing more processing time on better candidate solutions.

	Predictions						Performance		
NN	TC	SC	#V	Obj	%OT	CPU	SC	Obj	%OT
3	7482.35	104.41	15	8336.76	0.99898	0m6s	105.78	8338.13	0.99910
4	6502.35	208.48	12	7310.83	0.99796	0m21s	135.72	7238.07	0.99881
5	5536.04	677.56	9	6663.6	0.99337	2m2s	577.04	6563.08	0.99484
6	5349.3	585.12	8	6334.42	0.99429	2m34s	641.1	6390.4	0.99418
7	5626.54	123.5	9	6200.04	0.99879	5m48s	123.46	6200	0.99889
8	5559.83	293.92	8	6253.75	0.99714	6m1s	376.08	6335.91	0.99662
9	5108.83	612.58	7	6071.41	0.99408	8m5s	739.78	6198.61	0.99348
10	5599.69	443.13	8	6442.82	0.99570	9m6s	499.64	6499.33	0.99547
11	6146.74	262.4	10	6909.15	0.99743	24m12s	315.97	6962.71	0.99718
12	5475.36	343.06	8	6218.42	0.99664	14m11s	343.31	6218.67	0.99689
15	5953.83	445.54	9	6849.37	0.99574	27m5s	452.79	6856.62	0.99602

Table 5: Results for different numbers of nearest neighbors(NN)

In the next experiment we vary the minimum reduced cost value routes need to satisfy to be allowed in the column pool of the relaxed restricted master problem. We want to add routes with negative reduced cost to be able to be able to find a good solution to the integer linear programming as well. We see that first if we increase the reduced cost, the objective value decreases. But after a while the objective value stays the same. The margin of adding routes with positive reduced cost is big enough to find enough filler routes.

	Predictions						Performance		
MCR	TC	SC	#V	Obj	%OT	CPU	SC	Obj	%OT
0	6146.58	262.56	11	6959.15	0.99747	8m11s	244.53	6941.11	0.99792
500	5782.33	374.58	9	6606.92	0.99636	7m43s	321.35	6553.68	0.99718
1000	5870.9	302.98	9	6623.88	0.99707	6m57s	298	6618.9	0.99737
1250	5559.83	293.92	8	6253.75	0.99714	6m46s	374.91	6334.74	0.99663
1500	5559.83	293.92	8	6253.75	0.99714	6m1s	376.08	6335.91	0.99662
2000	5559.83	293.92	8	6253.75	0.99714	6m0s	377.01	6336.84	0.99661
2500	5559.83	293.92	8	6253.75	0.99714	6m2s	371.18	6331.01	0.99666

Table 6: Minimum value of the pricing problem (MC), if none of the new routes reach the value, we stop iterating.

In the next experiment we vary the number of states per customer that are expanded for consideration when extending the route using Dynamic Programming. The computation time increases but the objective value is more or less constant.

	Predictions						Performance		
#States	TC	SC	#V	Obj	%OT	CPU	SC	Obj	%OT
10	5401.6	321.3	9	6172.9	0.99685	9m4s	337.84	6189.44	0.99696
20	5626.54	123.5	9	6200.04	0.99879	5m58s	124.2	6200.74	0.99888
30	5488.18	183.45	8	6071.63	0.99820	4m38s	191.26	6079.44	0.99826
40	5522.39	502.23	8	6424.63	0.99511	5m59s	362.69	6285.08	0.99675
50	5269.61	534.47	8	6204.08	0.99477	8m43s	561.51	6231.12	0.99490
60	5500.85	583.71	8	6484.56	0.99431	15m9s	776.12	6676.97	0.99297
70	5699.3	320.98	9	6470.28	0.99689	11m49s	365.78	6515.08	0.99677

Table 7: Performance using different numbers of states.

10.5.2 Dynamic programming results

The column generation method with the dynamic programming solution for the pricing method is tested on all the test cases from Lekkerkerer et al. [13] for each distribution approach.The results can be found in Appendix A.

What we can see from the results is that the MaxnormalTree and the MaxNormal give the best results for normal distributed travel times according to actual objective value. This is what we expected after the results in 10.2, 10.3 and 10.4

With the settings we picked out we can see that it works good for the test cases it was tested on. However if time windows get very small the number of nearest neighbors should be increased because it is harder to find a good feasible solution with only considering the few best customers. If the time windows are bigger than selecting only the few best customers is fine since the actual travel cost between customers is more import on reducing the objective value. Using less nearest neighbors will reduced the running time significantly. Also if time windows are smaller there are less feasible options so many branches are pruned more early because of infeasibility, thus even though needing to increase the number of nearest neighbors the running time is not necessarily increased. This means that it is hard to find a setting that works well on all test cases. It is more realistic to tune the settings to the data you work with, to the average size of the time windows and the number of customers.

For the datasets with the original large time windows we use the setting found in the previous section. However for the test cases with the adjusted time window we adjusted the settings of the dynamic programming algorithm. In the case of smaller time windows we also adjusted the penalty of on lateness, since its likely that the expected lateness varies a lot if we consider time windows of 6 hours or time windows of 10 minutes. To overcome that we in case of small time windows create very few routes we decrease the penalty on lateness. To test the performance on different time window sizes, we took dataset V2015-M and reduced the time window of each customer to, 60, 120, 240 or 480 minutes, named V2015-M-T60, V2015-M-T120, V2015-M-T240 and V2015-M-T480 respectively. The performance of the GammaTree and MaxNormalTree are tested on these adjusted sets. The results are in Appendix B.

In these tests we used other settings, (nearest neighbors = 25, number of states per customer = 70), to find an optimal solution with more specific time windows. In the tables for the gammaTree and the maxNormalTree we can clearly see that if we have smaller time windows the optimal number of vehicles increases. The objective value also increases because there are less possible feasible route and thus less options to limit the travel cost. We see that the reliability of the schedule remains high.

10.5.3 Settings Local Search

First we conducted some tests to determine appropriate values for the parameters used in the Local Search method. To calibrate the number of iterations of solving the RMP and the pricing problem and the number of iterations in the Local Search method to solve the pricing problem one time. To determine these values the number of iterations versus the objective value of the relaxed restricted master problem are plotted in figure 6 and the objective value versus the time are plotted in figure 7. The number of iterations needed will however strongly depend on the size of the dataset. Figure 6 shows that the solution value of the relaxed restricted master problem decreases quickly in the first 40 iterations, in the next 160 iterations it decreases slowly for the dataset V2016-M.

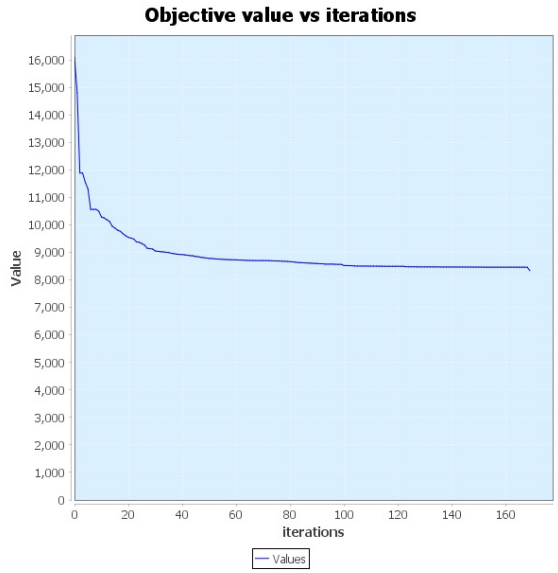

Figure 6: Objective value vs iterations

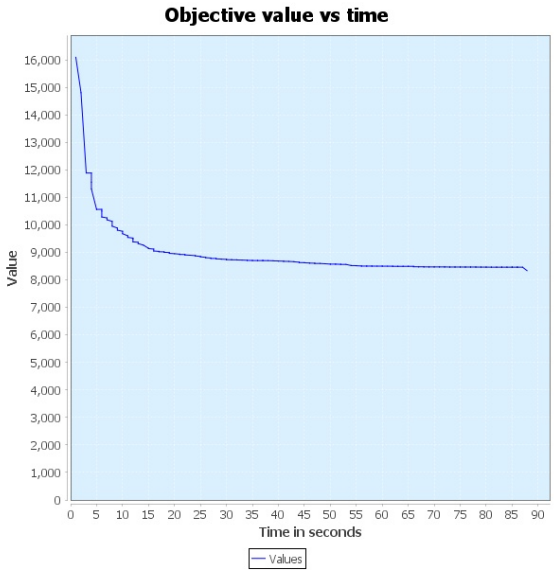

Figure 7: Objective value vs time

Like in the dynamic programming approach we also have the possibility to set the maximum reduced cost for which we allow a new route to be added to the column pool. Thus again to have to possibility to set it to a value bigger to zero to close the gap between the relaxed problem and the integer linear programming problem. We set this setting to the same value as we used for the dynamic programming approach.

We also added the possibility to have second rank columns, created with the method of Diepen et al. [4]. These extra columns are generated in every iterations but are only added to the column pool just before we solve the RMP.

Since the 2-opt and reordering operation are relatively costly compared to adding or deleting a customer, we checked how often a new solution was accepted if it was created with one of these too. It appeared that 2-opt almost never improved the solution even in the case with the original time windows, thus is decided to leave this operation out. Since the smaller the time windows will get, the less likely a 2-opt operation even gives a feasible solution let alone an improvement. The reordering operation does give improvements, but less frequently than the two other operations. It is still used but the probability of using the reorder operator is lower than the probability to get one of the other two operators.

10.5.4 Local search results

In Appendix C the results of the local search approach on the test tests V2015-T, V2015-S, V2015-M, V2016-T, V2016-S, V2016-M are shown. We see that for both the gamma distribution and the normal distribution the results for the tiny and small test cases are comparable to the results of the dynamic programming approach. However the calculation time with the local search approach increases rapidly when tested on the large dataset. To improve this we tried to start the algorithm from different starting solutions and to increase the number of iterations, or the number of iterations in the local search algorithm. However the problem seems mainly in the step of converting to an integer solution.

10.5.5 Performance

The dynamic programming approach and the local search approach give in general comparable results. If we look at the data in Appendices A, C and B, we can see the predictive value and the actual result of each test. The difference between this is caused by how well the distribution functions predict the service cost of the routes, since the traveling cost between customers does not depend on the travel times.

Sometimes there is a small gap between the solution value of the dynamic programming and the local search, both are heuristics, but maybe with tweaking the settings even better they could both reach the minimum in a more stable way. However in this thesis the mean focus was on how the different distribution methods perform to generate a good solution and how well they predict the performance of the routes. In Appendix A we see that the predicted value for the percentage on time served is closest to the actual value for the percentage on time for the MaxNormal, MaxNormalTree and the GammaTree.

In Figures 8 and 9 we see the performance of the two pricing methods on the Tiny, Small and Medium data sets. In Figure 8 we see the performance using the

MaxNormalTree method and in Figure 9 when the GammaTree is used. We see that the performance is quite similar. The distributions with the tree structure do cost more time as the cases are getting bigger. There was a huge increase in running time when using local search on the large data set, but with dynamic programming it is still solved in a reasonable time.

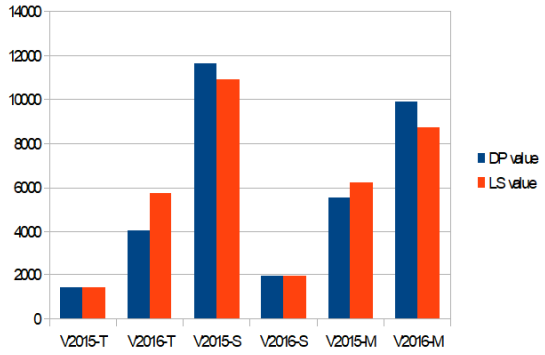

Figure 8: Performance MaxNormalTree

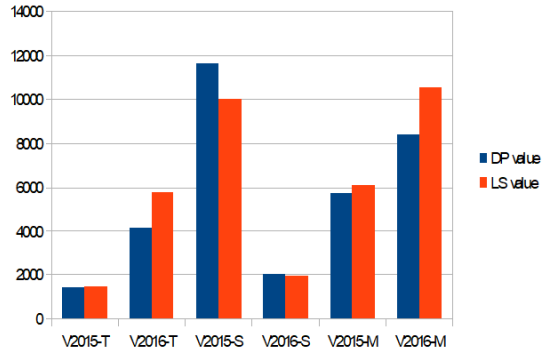

Figure 9: Performance GammaTree

11 Conclusion

We presented several stochastic distributions that can be used to model travel times and calculate the probability of delay at the customer, or overtime of the vehicle on a route. These probabilities were used in the objective function to penalize a high probability of delay.

The max normal tree and the gamma tree method performed best, showing how our method of building trees that branch between constant values representing waiting at a customers, and branches with a distribution that represents the sum of uncertainty of traveling from one customer to another, perform better than single distributions. Using this method we manage to accurately estimate the effect of high levels of variance. Even when the variance was not correctly estimated, these distributions still proved robust to a high a degree to higher levels of variance.

Several options to speed up the dynamic programming approach at the cost of less exhaustive searching were tested with good results.

We used both dynamic programming and local search to solve the pricing problem and concluded that the performance was quite similar, in order of time and in order of objective value. However dynamic programming was superior in the large datasets, where local search significantly increased in time when transforming the solution to an integer solution. While the dynamic programming solution worked steadily towards an incrementally better solution, the local search improved more randomly. This makes it more predictable how close we are to an optimal solution, as far as the pricing problem is concerned, thus allowing termination when no big improvements are expected any more.

Unfortunately due to practical limitations of the implementation, the max gamma tree did not result in a solution. Given that the max normal tree and the gamma tree both work well, the max gamma tree would still be promising distribution to try.

11.1 Future research

The problem could further be extended to incorporate pick-up and delivery, service times and vehicle capacity constraints to make it even more realistic for real life cases. The service time and vehicle capacity additions are relatively easy to incorporate, however the pick-up and delivery case only works with the local search pricing problem and may increase the running time significantly.

As the tree structure proved to be a good improvement over "regular" distributions, implementing a max gamma tree would still seem promising. In addition other distributions for the travel times that are used in other research could also be considered to be used in our trees like the shifted gamma or the log normal.

As the solution was only tested using simulations, investigating its performance in real world situations would be interesting. While it performed very well in the simulations, our solution and the test set up rely heavily on the assumption that the travel times between customers are stochastically independent, while in real world applications this is likely not the case. If there is heavy congestion from customer c_1 to c_2, then heavy congestion can also be more likely expected from c_2 to c_3. Further research into the impact of to what extend this assumption holds would be interesting.

For the conversion from RRMP to RMP we have used Van Diepen et al. [4] to convert from the fractional solution to a good integer solution. Potentially other methods like a branch and bound method could also be considered as the results are not always optimal.

While part of the focus of our research was on letting the heuristic methods for the pricing problem perform well in a practical amount of time, it would be interesting to see how they fare against an exact method like an MIP formulation of the pricing problem. Potentially an exact solution could also be used every n iterations of the column generation to check whether the heuristic methods have found all, or most routes with negative reduced cost and we can therefore terminate.

References

[1] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118, 2005.

[2] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[3] G.B. Dantzig and J.H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[4] G. Diepen, J.M. van den Akker, J A. Hoogeveen, and J.W. Smeltink. Finding a robust assignment of flights to gates at amsterdam airport schiphol. *Journal of Scheduling*, 15(6):703–715, 2012.

[5] J.F. Ehmke, A.M. Campbell, and T.L. Urban. Ensuring service levels in routing problems with time windows and stochastic travel times. *European Journal of Operational Research*, 240(2):539–550, 2015.

[6] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[7] M. Gendreau, G. Laporte, and R. Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, 1996.

[8] Inc. Gurobi Optimization. Gurobi optimizer reference manual. 2016.

[9] M. Held and R.M. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962.

[10] S. Ichoua, M. Gendreau, and J. Potvin. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379–396, 2003.

[11] A.L. Kok, E.W. Hans, and J.M.J. Schutten. Vehicle routing under time-dependent travel times: The impact of congestion avoidance. *Computers & Operations Research*, 39(5):910–918, 2012.

[12] G. Laporte. What you should know about the vehicle routing problem. *Naval Research Logistics (NRL)*, 54(8):811–819.

[13] M. Lekkerkerker. Robust scheduling of the vehicle routing problem with time windows. *Thesis presented for the degree of Master in Computer Science*, 2016.

[14] C. Malandraki and M.S. Daskin. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, 26(3):185–200, 1992.

[15] S. Nadarajah and S. Kotz. Exact distribution of the max/min of two gaussian random variables. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(2):210–212, Feb 2008.

[16] F. E. Satterthwaite. An approximate distribution of estimates of variance components. *Biometrics Bulletin,*, 2:110–114, 1946.

[17] M.M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[18] 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. *Comput. Oper. Res.*, 40(1):214–224, January 2013.

[19] 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. *European Journal of Operational Research*, 236(3):789–799, 2014. Vehicle Routing and Distribution Logistics.

Appendices

A Results dynamic programming

Normal	Prediction						Actual result		
Dataset	TC	SC	#V	Obj	%on time	CPU time	SC	Obj	%OT
V2015-T	496.67	355.65	1	1852.32	0.9717960758	0m0s	0.3	1496.97	0.99998
V2015-S	2321.4	1880.62	3	7202.02	0.9529269562	0m3s	2.63	5324.03	0.9999124138
V2015-M	4924.76	2839.82	7	14764.58	0.9673390721	0m37s	0.65	11925.41	0.9999907143
V2015-L	6757.23	4946.23	11	22703.45	0.9656275997	1m2s	5.26	17762.49	0.9999614159
V2016-T	1545.51	188.09	2	3733.6	0.98869	0m0s	0	3545.51	1
V2016-S	3805.75	1779.06	5	10584.81	0.9707832022	0m3s	9.26	8815.01	0.999969375
V2016-M	6502.12	3281.24	8	17783.36	0.9672508887	0m6s	26.02	14528.14	0.9999525424
V2016-L	9416.15	6433.26	13	28849.41	0.9718046868	1m39s	357.3	22773.45	0.9996853719

Table 8: Results of the normal distribution using dynamic programming

MaxNormal	Prediction						Actual result		
Dataset	TC	SC	#V	Obj	%on time	CPU time	SC	Obj	%OT
V2015-T	419.47	146.63	1	1566.09	0.9881864885	0m0s	6.67	1426.14	0.99949
V2015-S	2336.93	195.08	3	5532.01	0.9949355311	0m5s	0.19	5337.12	0.9999934483
V2015-M	4315.44	1216.01	6	11531.45	0.9872992169	0m51s	68.21	10383.65	0.9991053623
V2015-L	6605.57	1366.4	9	16971.96	0.9872014491	3m24s	120.91	15726.48	0.9989107207
V2016-T	1014.07	29.57	1	2043.64	0.99812	0m0s	0.32	2014.39	0.9999709091
V2016-S	2448.45	603.98	3	6052.43	0.9960639575	0m5s	84.53	5532.98	0.9996753333
V2016-M	4563.32	950.36	5	10513.68	0.9941663943	0m37s	98.64	9661.96	0.9995226786
V2016-L	7814.94	2939.24	10	20754.18	0.9791028259	4m4s	243.99	18058.93	0.9980505085

Table 9: Results of the MaxNormal distribution using dynamic programming

NormalTree	Prediction						Actual result		
Dataset	TC	SC	#V	Obj	%on time	CPU time	SC	Obj	%OT
V2015-T	478.12	242.07	1	1720.19	0.9800633028	0m0s	0.64	1478.76	0.99995
V2015-S	2353.26	514.58	3	5867.84	0.9869439326	0m4s	0.21	5353.47	0.9999927586
V2015-M	4378.38	1275.39	6	11653.77	0.9832998255	2m57s	1.93	10380.31	0.999972029
V2015-L	6291.91	1871.26	10	18163.17	0.9847351848	3m24s	3.71	16295.62	0.999966875
V2016-T	995.91	434.81	1	2430.72	0.97420	0m0s	0.35	1996.26	0.9999681818
V2016-S	3269.82	835.76	4	8105.58	0.9845375371	0m2s	82.31	7352.13	0.9977309677
V2016-M	5246.1	1390.01	6	12636.11	0.9845075803	0m13s	28.32	11274.42	0.9997068421
V2016-L	8710.58	2976.13	11	22686.71	0.9828753075	1m49s	123.56	19834.14	0.9989919328

Table 10: Results of the NormalTree distribution using dynamic programming

MaxNormalTree	Prediction						Actual result		
Dataset	TC	SC	#V	Obj	%on time	CPU time	SC	Obj	%OT
V2015-T	418.04	30.94	1	1448.98	0.9974216199	0m0s	7.51	1425.55	0.99942
V2015-S	1953.09	592.89	2	4545.98	0.9773281002	0m3s	58	4011.09	0.997935
V2015-M	4500.46	912.36	7	12412.81	0.9856396624	0m42s	94.3	11594.76	0.9986541429
V2015-L	6343.3	705.38	10	17048.68	0.9931478299	11m39s	96.57	16439.87	0.9991433929
V2016-T	989.28	16.04	1	2005.32	0.99901	0m0s	2.58	1991.86	0.9998963636
V2016-S	2498.63	197.87	3	5696.5	0.9962501964	0m4s	39.48	5538.11	0.99911
V2016-M	4789.67	351.71	5	10141.38	0.9939258137	0m14s	79.29	9868.96	0.9988525
V2016-L	7605.07	857.33	10	20426.88	0.9935723491	2m21s	406.08	18011.15	0.9035640678

Table 11: Results of the MaxNormalTree distribution using dynamic programming

Gamma	Prediction						Actual result		
Dataset	TC	SC	#V	Obj	%on time	CPU time	SC	Obj	%OT
V2015-T	417.91	9.65	1	1427.56	0.9991958295	0m0s	8.18	1426.09	0.99937
V2015-S	1909.83	613.55	2	4523.38	0.976551078	0m7s	140.52	4050.35	0.9950007143
V2015-M	4727.09	1160.25	6	11887.34	0.9818524387	0m55s	225.79	10952.88	0.9967276812
V2015-L	7341.89	2669.98	11	21011.87	0.9744328369	1m23s	2264.53	20606.42	0.9803971681
V2016-T	995.91	27.63	1	2023.54	0.99724	0m0s	0.48	1996.39	0.9999563636
V2016-S	2529.26	933.12	3	6462.38	0.9710372901	0m7s	16.94	5546.2	0.9999033333
V2016-M	4131.86	1643.68	4	9775.54	0.9728658848	0m37s	422.94	8554.8	0.9938909091
V2016-L	8494.47	2793.45	11	22287.91	0.9777622201	8m0s	1405.42	20899.89	0.9893423529

Table 12: Results of the gamma distribution using dynamic programming

GammaTree	Prediction						Actual result		
Dataset	TC	SC	#V	Obj	%on time	CPU time	SC	Obj	%OT
V2015-T	417.91	7.92	1	1425.83	0.9993396952	0m0s	8.33	1426.24	0.99936
V2015-S	1969.58	178.73	2	4148.32	0.9931337435	0m4s	186.84	4156.42	0.9933271429
V2015-M	4777.41	683.76	6	11461.17	0.9891476476	1m7s	835.73	11613.14	0.987887971
V2015-L	5799.4	1245.67	8	15045.07	0.9877926349	4m4s	1486.37	15285.77	0.9865120909
V2016-T	995.57	12.1	1	2007.67	0.99879	0m0s	0.98	1996.55	0.9999109091
V2016-S	2585.91	245.39	3	5831.3	0.995549353	0m6s	129.19	5715.1	0.9989096667
V2016-M	4129.83	678.37	4	8808.2	0.9902183007	0m44s	259.06	8388.89	0.9969278182
V2016-L	8061.66	1768.86	10	19830.52	0.9842844952	4m56s	2027.51	20089.17	0.9831830508

Table 13: Results of the gammaTree distribution using dynamic programming

B Performance with different time window sizes

GammaTree	Prediction						Actual result		
Dataset	TC	SC	#V	Obj	%on time	CPU time	SC	Obj	%OT
V2015-M-T60	6233.77	64.31	9	15298.08	0.8979199244	3m3s	43.4	15277.17	0.9397286111
V2015-M-T120	5486.84	66.51	8	13553.35	0.8944274169	3m59s	41.87	13528.71	0.9410266197
V2015-M-T240	5770.26	79.81	7	12850.08	0.8733105008	4m11s	40.57	12810.84	0.9420391429
V2015-M-T480	4912.2	82.36	7	11994.56	0.8692700516	4m13s	44.2	11956.4	0.9368551429

Table 14: Performance gammaTree on different time window sizes.

MaxNormalTree	Prediction						Actual result		
Dataset	TC	SC	#V	Obj	%on time	CPU time	SC	Obj	%OT
V2015-M-T60	8104.12	29.54	13	21133.66	0.953115431	0m23s	9.21	21113.33	0.9878881579
V2015-M-T120	7138.02	26.93	11	18164.95	0.9572603336	0m36s	13.86	18151.88	0.9812727027
V2015-M-T240	7021.79	15.44	11	18037.23	0.9754906366	0m29s	12.48	18034.27	0.9831351351
V2015-M-T480	6003.14	25.9	9	15029.04	0.9588934869	0m49s	22.12	15025.26	0.9692781944

Table 15: Performance MaxNormalTree on different time window sizes.

C Results Local Search

MaxNormalTree	Prediction						Actual result		
Dataset	TC	SC	#Vehicles	Obj	%on time	CPU time	SC	Obj	%OT
V2015-T	406.42	0.01	1	1406.43	0.9999992035	0m1s	0	1406.42	1.00000
V2015-S	2693.44	96.64	3	5790.08	0.9962875537	0m53s	2.67	5696.11	0.999907931
V2015-M	4820.83	465.18	6	11286.01	0.9926241625	0m49s	62.25	10883.08	0.9990978261
V2016-T	988.3	2.49	1	1990.79	0.99980	0m0s	0	1988.3	1
V2016-S	3216.36	59.3	3	6275.67	0.9978264716	0m32s	3.31	6219.67	0.9998896667
V2016-M	4606.39	926.17	4	9532.56	0.9838387433	0m36s	119.97	8726.36	0.9978547273

Table 16: Results for the MaxNormalTree distribution using local search

GammaTree	Prediction						Actual result		
Dataset	TC	SC	#Vehicles	Obj	%on time	CPU time	SC	Obj	%OT
V2015-T	463.9	0.01	1	1463.91	0.9999990486	0m1s	0.01	1463.91	0.9999992308
V2015-S	2751.92	13.78	3	5765.7	0.9994698144	0m54s	14.17	5766.09	0.9995113793
V2015-M	5099.76	873.18	4	9972.95	0.9861631991	0m55s	883.67	9983.43	0.9868431343
V2016-T	969.50000	0.12	1	1969.62	0.9999883571	0m0s	0	1969.5	1.00000
V2016-S	3038.21	102.64	3	6140.85	0.99620	0m21s	24.35	6062.56	0.9991883333
V2016-M	5380.46	335.97	5	10716.43	0.9937583596	1m37s	192.14	10572.6	0.99668625

Table 17: Results for the GammaTree distribution using local search