The Research into the Vehicle Routing Problem with Self-Imposed Time Windows

Bachelor Thesis
Econometrics and Operational Research

Author:
Jaron Davelaar
433691

Supervisor:
Naut Bulten
Second Assessor:
Dr. Twan Dollevoet

July 8, 2018

ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Abstract

Small-package carrier companies transport goods to customers. In this article we will focus on companies who provide a time window for delivery to its customers. The company strives to respect this selected time windows as well as possible. The self-imposed time window (SITW) has an important aspect that it is selected by the company and not by the customer. We include this self-imposed time window into the Vehicle Routing Problem (VRP). The VRP-SITW is different from the VRP with time windows (VRPTW), because in the latter problem the time windows are chosen by the customer itself. Another main aspect of this article is the incorporation of uncertainty in the travel times. Disruptions may occur in the routes and we account for this by allocating buffers. We used a two-stage solution approach with a tabu search heuristic and a linear programming problem as introduced by Jabali et al. (2015). In the first stage the routing and the ordering of the customers within a route is done by using the tabu search procedure. The second stage generates the schedules of the time windows using the linear programming problem. In this article we test the algorithm on a number of benchmark instances. We indicate the costs involved in including the SITW in the VRP and in addition to this, highlight the advantages of the VRP-SITW over the VRPTW. Lastly, we make some alterations in the algorithm to test the influence on the final obtained costs.

Keywords: Vehicle Routing Problem · Self-Imposed Time Windows · Disruptions · Buffers ·Tabu Search · Linear Programming

Contents

1 Introduction

Over the last few decades the purchasing of goods via the Internet has played a tremendous role in the world economy. In the United Kingdom, the highest online shopping rate is obtained. In 2016, 85% of the Internet users in the United Kingdom purchased goods and services online, as investigated by UNCTAD (2016). The number of people in the world buying online was equal to approximately 1.52 billion in 2016 and this number is still increasing. The expected number of online buyers in 2021 will transcend the 2 billion (Statista (2017)). Most of the goods purchased online have to be transported to the customers. The small-package carrier companies can provide this service.

The arrival time of a good is important to the customer. For some goods, the customer needs to stay at home in order to sign for the product. Some of the carrier companies let decide their customers when they want to be served. The allocation of vehicles and the routing of this company is based on the exogenous decisions of the time windows of his customers. Other small-package carrier companies provide their customers with time windows for delivery. These time windows are endogenous decisions of the company. One example of such a company, where the time windows are chosen by the company itself, is UPS. Once the time window is chosen for a specific customer, the company aims to serve the client within the specified time window.

In this article we will focus on the endogenous time windows, or in other words, the self-imposed time windows. These time windows are selected by the company. The location of the customer will influence the time window, but the preference of the customer for a certain time window will not be taken into account. The time windows are endogenous to the routing problem. Three steps have to be taken to solve the routing problem with these self-imposed time windows. First of all, the carrier company has to assign customers to different vehicles. Thereafter the company has to sequence the customers assigned to one vehicle. When both decisions are made, the scheduling of the time windows for the customers can be done. This described problem is the Vehicle Routing Problem with Self-Imposed Time Windows (VRP-SITW), like also discussed in Jabali et al. (2015).

The carrier companies have to deal with uncertainties in their daily planning. A vehicle breakdown or a traffic jam can damage the planning of the company. In this article we consider delays in travel time due to disruptions. These delays in a route may let the actual arrival time deviate from the selected time window. In order to avoid this phenomenon, we can include time buffers in our schedules, as done by for example Yeo and Ning (2006). We will use a buffer allocation model in order to account for the disruptions in travel time.

Our aim is to construct routes and schedules of the time windows for the VRP-SITW who will deal best with disruptions and will try to minimize the operational and customer service costs. A solution to the problem is generated before the start of a certain planning horizon and can not be altered during this horizon. In order to obtain the routes for the problem a few assumptions have to be made. First, we assume that the service of a customer cannot start before the scheduled time window. When an early arrival occur, the driver of the vehicle has to wait until the start time of the time window is reached to serve the customer. On top of this, late arrivals are penalized relatively to their tardiness. One important remark is the fact that early arrivals are not penalized. Lastly, a driver of a vehicle has a fixed shift length and has to be paid a constant amount of money per day.

In order to get a good solution for the VRP-SITW, we will use a two-stage solution approach, introduced by Jabali et al. (2015). In the first stage the assigning of customers to routes and the sequencing in each route is done by using a tabu search heuristic. The second stage will

schedule the time windows by solving a linear programming problem, given a certain set of routes. In the tabu search heuristic of stage 1 an initial set of routes is needed. In this article we will use different procedures to find the initial solution and explore the effect of the initial solution on the final obtained costs. On top of this, we will set different penalty settings for not respecting the time windows and explore the effects on the acquired costs. Besides these alterations, we will evaluate the solutions of the VRP-SITW by comparing these to the solutions of the VRP and the VRP with time windows set by the customer (VRPTW). The main contributions of this article are twofold. First of all, we analyze the results from the article of Jabali et al. (2015). Furthermore, we explore the effect of the initial solution on the final obtained costs.

This article is organized as follows. In Section 2 the literature regarding the VRP-SITW is discussed. In Section 3 the problem is formalized and the different aspects of the problem are outlined. In Section 4 the methodology in order to solve this problem is explained and Section 5 describes the data. Section 6 discusses the results and finally, in Section 7 the conclusion of our findings is given.

2 Literature Review

In the scientific literature of the last decades the Vehicle Routing Problems are studied quite frequently. The Vehicle Routing Problem (VRP) was introduced over 50 years ago by Dantzig and Ramser (1959). Dantzig and Ramser called it the Truck Dispatching Problem, but nowadays the Vehicle Routing Problem is the most commonly used name. The VRP is considered as a generalization of the Traveling Salesmen Problem. Due to the capacity constraints of the vehicles, the VRP may use multiple vehicles in order to satisfy all the demand of the customers. In the Traveling Salesmen Problem only one vehicle without a limited capacity is used.

The tabu search procedure is commonly used in order to solve Vehicle Routing Problems. The procedure is used by Gendreau et al. (1994) for example. The algorithm of this article considers a sequence of routes obtained by removing a vertex from the current route and add this vertex in another route. Routes that were acquired recently, are forbidden and added into the tabu list to prevent the phenomenon of cycling.

The classical VRP considers deterministic travel times. But over the years many variations of the VRP have been studied. One variation is the incorporation of stochastic travel times and service times, as done by for example Laporte et al. (1992). One reason for stochasticity in travel time is the weather condition. On a rainy or snowy day, driving is likely to be more difficult and will take more time compared to driving on a sunny day. When vehicles with a fixed capacity must collect random quantities from customers, stochasticity in service times is obtained.

Another variation of the VRP is the inclusion of time windows. The service of a customer can only begin within the time window imposed by the customer. This problem is called the Vehicle Routing Problem with Time Windows (VRPTW) and has been researched by for example Desrochers et al. (1992) and Homberger and Gehring (2005). The time windows naturally arise in business organizations that work on fixed time schedules and therefore have to be included in this specific problem. In this VRPTW the customer itself can set his time window, independently of the company. The company strives to satisfy the time windows specified by the customers as well as possible.

Another form of stochasticity is incorporated in the VRP with Stochastic Demands (VRPSD). In this problem the demand is not known in advance but announced upon the arrival of the

vehicle at the customer. The VRPSD is for example investigated by Laporte et al. (2002). In some cases, the vehicle is unable to serve the amount of goods a customer wants and this situation will be seen as a failure. The vehicle has to return to the depot, in order to collect another amount of goods. Penalties are charged for failures corresponding to a return trip to the depot.

In our specified VRP-SITW from the article of Jabali et al. (2015), a few differences are indicated compared to the above named problems. First of all, the demand of the customers is known in advance. A second difference is the fact that the time windows are not imposed by the customer but by the company itself. Apart from Jabali et al. (2015), the VRP-SITW has never been studied before in scientific research.

3 Problem Description

In this section we discuss the different aspects of the VRP-SITW. First, we give a general description of the VRP-SITW. Thereafter the objective function is introduced, together with its different components. Then the self-imposed time windows are discussed and the modeling of disruptions. And lastly, the tardiness and overtime penalties are introduced and explained.

3.1 General Description of the VRP-SITW

The solution of the VRP-SITW gives the routes and the schedules of the time windows. To solve the problem some notation has to be introduced. The set N is the set of customers who have to be served. The set K is the set of available vehicles. All the customers have a certain location where they have to be visited. The directed graph $G = (V, A)$ represents the network with $V = \{0, \ldots, |N|\}$ the set of vertices, or in other words, the customers and A the set of arcs. These arcs indicate the connection between the different customers. The vertex 0 represents the depot. The Eucledian distance is used in order to determine the distances between the vertices, as in Danielsson (1980). The parameter d_{ij} gives the Eucledian distance between vertex i and j.

Some specific requirements are needed to solve the problem. First, the routes have to start and end at the depot. On top of that, the total demand of the customers on one route may not exceed the vehicle capacity, which is introduced as Q. Lastly each customer has to be visited exactly once, in order to satisfy his demand. A company has to determine a set of routes to satisfy all the demand of its customers. The set Z is a set of possible routes, where $Z = \{R_1, \ldots, R_{|Z|}\}$. Because the availability of a limited number of vehicles, the number of routes $|Z|$ may not exceed the number of available vehicles in K. Each route R_r is a vector with the first and last element equal to zero. A route is indicated as follows: $(0,\ i,\ j,\ \ldots,\ 0)$ with $i,\ j \in V$. To be more convenient in the remaining part of this article, the customers of a route will be displayed in an ascending order as follows: $R_r = (0, 1, \ldots, n_r, n_r + 1)$, where $n_r + 1$ corresponds with the depot (vertex 0). The distance between two consecutive customers in a route, $d_{i,i+1}$, is for sake of simplicity written as d_i in the remaining part of the article.

3.2 Objective Function

The objective function of the VRP-SITW accounts for two different categories of costs. First it considers the operational costs. These costs are the costs for operating a vehicle, consisting of for example the salary of the driver and the fuel costs. The second cost category is the customer service costs. These costs consist of two components. The first component is a penalty for not respecting the imposed time window, called the tardiness penalty. When the vehicle arrives at the customer after the imposed time window, the penalty is incurred. A late arrival will lead to a decrease in customer satisfaction. One important remark is the fact that an arrival before the time window is not penalized, but the service can start only after the beginning of

the time window. Besides this tardiness penalty, the objective function of the VRP-SITW also imposes an overtime penalty. The drivers of the vehicles have a specific shift length and when a certain driver has a route duration longer than the shift length, an overtime penalty has to be paid. Both penalties are discussed in more detail in Section 3.5. Considering this different categories of costs, the objective function of the VRP-SITW, introduced by Jabali et al. (2015), is as follows:

$$F(Z) = c \sum_{R_r \in Z} \sum_{(i,j) \in R_r} d_{ij} + \sum_{R_r \in Z} \Theta(R_r). \tag{1}$$

In this equation the parameter c is the cost of traveling one unit of distance and therefore the first part of this equation specifies the operational costs. The function $\Theta(R_r)$ represents the tardiness and overtime penalties of the route R_r, which will be formulated in Section 3.5.

3.3 The Self-Imposed Time Windows

As mentioned in Section 1, the VRP-SITW is a problem where the company itself decides on the time windows for delivery of his customers. Given a specific route and the corresponding distances between the customers, a schedule of this route can be made. The departure time s_i for each customer $i \in R_r$ is given in the schedule vector $\mathbf{s} = (s_0, s_1, \ldots, s_{n_r+1})$. A driver of a vehicle has a certain shift length and this shift length is indicated as $[s_s, s_e]$ where s_s is the start time and s_e the end time of a shift. A vehicle cannot departure from the depot before the start time of the shift and therefore $s_s \leq s_0$ holds. For each customer, except for the depot, a time window length W_i is given in which the arrival of the vehicle is requested. The time window is based on the schedule of the departure times of the customers ($\mathbf{s}$). The company will communicate these time windows to his customers.

Each customer has to be served and this service is time costly. The service time of a customer is u_i and it is known before the planning horizon. This means the company knows the service time of a customer before the arrival of the vehicle at the customer. The service times u_0 and u_{n_r+1} are equal to zero because these vertices are the depot in a route, as indicated in Section 3.1. We assume that a vehicle never leaves a customer before the scheduled departure time. Therefore the service cannot start before $s_i - u_i$, for each customer $i \in R_r$. If the service starts before this specified time, the service ends before the scheduled departure time, which gives a contradiction with the assumption. Hence, the left bound of the time window equals $s_i - u_i$. Considering this left bound of the time window and the time window length W_i, the right bound of the time window is equal to $s_i - u_i + W_i$.

3.4 Disruptions

In the described VRP-SITW we will incorporate stochasticity in the traveling times, as also done by Jabali et al. (2015). There may occur disruptions in the routes, leading to an increase in the actual traveling time. The length L_i of the delay on arc (i, i+1) is a random variable and is added to the travel time d_i. The variable L_i can take some discrete values, as also done by for example Kouvelis et al. (2000). The probability of having a delay on arc (i, i+1) is equal to p_i. The variable L_i has a probability-mass function $g_i(\cdot)$, which gives probabilities to the positive delay values $l_{ik} \in \Psi_i$. The set Ψ_i is the set of disruption scenarios for d_i. Each scenario has a specific increase in traveling time. As a result the following holds:

$$\sum_{k \in \Psi_i} g_i(l_{ik}) = 1. \tag{2}$$

We have to make an assumption in order to model the disruptions. We assume that a route can suffer at most one failure or delay. So only one arc of a route has an increase in travel time. A similar assumption is made by Jabali et al. (2009). However, we will consider different delay scenarios, due to the set Ψ_i. For each arc, we consider a set of different disruption lengths because a vehicle breakdown may take longer than the delay due to, for instance, bad weather conditions.

Due to the disruptions, the actual departure time may differ from the scheduled departure time. The actual departure time ($s_i^a(\mathbf{s})$) is a function of $\mathbf{s}$, because the scheduled departure time influences the actual departure time. Because it is known that the service cannot start before $s_i - u_i$, the scheduled departure time does not exceed the actual departure time. When no disruptions occur, s_i is equal to $s_i^a(\mathbf{s})$, $\forall i \in R_r$. But when disruptions occur, the scheduled departure time may deviate from the actual departure time. At the beginning of the route, in the depot, no disruptions have occur and therefore the actual departure time is equal to the scheduled one. But after leaving the depot disruptions may appear and the departure times are likely to differ. The following two equations give the actual departure time as a function of the scheduled time:

$$s_0^a(\mathbf{s}) = s_0, \tag{3}$$
$$s_i^a(\mathbf{s}) = \max\{s_i; s_{i-1}^a(\mathbf{s}) + d_{i-1} + L_{i-1} + u_i\}. \tag{4}$$

3.5 Tardiness and Overtime Penalty

The tardiness penalty is introduced as extra costs due to a decrease in customer satisfaction. An arrival before $s_i - u_i$ is not penalized, the driver of the vehicle only has to wait until he can serve his client. However, an arrival after the right bound of the time window ($s_i - u_i + W_i$) is penalized. The non-negative integer t_i is the penalty of one time-unit delay at customer $i \in R_r$.

Besides the tardiness penalty, the overtime penalty is also incorporated in the objective of the VRP-SITW. A driver has a certain shift length which he gets paid for. When he arrives at the depot later than the end time of the shift, a penalty has to be paid. This overtime penalty b is the cost of arriving one time-unit too late at the depot. So the penalty function $\Theta(R_r)$, given in Section 3.2, consists of two components namely the costs of the expected delay at the customers and the overtime costs. Therefore, the function can be written as follows:

$$\Theta(R_r) = \sum_{i \in R_r \setminus \{0\}} t_i \mathrm{E}[\max\{0; s_i^a(\mathbf{s}) - (s_i - u_i + W_i)\}] + b\mathrm{E}[\max\{0; s_{n_r+1}^a(\mathbf{s}) - s_e\}]. \tag{5}$$

The first part of the summation is the expected delay at the customers and the second part is the overtime at the end of the route. In order to model the penalty function we have to write the function in another form, like also done in Jabali et al. (2015). We distinguish two situations namely the situation where no arc is disturbed and where one arc is disrupted. There could be routes who have an overtime even if no disruptions arise. This overtime is incorporated in the ζ-variable which is the overtime of route R_r with no disrupted arcs. The penalty function can be written as follows:

$$\Theta(R_r) = \min \sum_{i=0}^{n_r} \sum_{j=i+1}^{n_r+1} \sum_{k=1}^{|\Psi_i|} p_i g_{ik} t_j \Delta_{ijk} + b \sum_{i=0}^{n_r} \sum_{k=1}^{|\Psi_i|} p_i g_{ik} \Lambda_{ik} + b\Big(1 - \sum_{i=0}^{n_r} p_i\Big)\zeta, \tag{6}$$

with the following three variables included:

$$\Delta_{ijk} = \max\left\{0; s_i + d_i + l_{ik} + \sum_{m=i+1}^{j-1}(u_m + d_m) - s_j + u_j - W_j\right\}, \quad (7)$$
$$i \in R_r \backslash \{n_r + 1\}; j \in R_r \backslash \{0\}; i < j; k \in \Psi_i,$$
$$\Lambda_{ik} = \max\{0; s_{n_r+1} + \Delta_{i,n_r+1,k} - s_e\}, \quad i \in R_r \backslash \{n_r + 1\}; k \in \Psi_i, \quad (8)$$
$$\zeta = \max\{0; s_{n_r+1} - s_e\}. \quad (9)$$

The first summation of Equation (6) represents the expected delay costs at the customers. The second summation is the expected overtime penalty for arriving late at the depot due to disruptions. The last summation represents the expected overtime penalty in case of no disruptions. The variable Δ_{ijk} gives the tardiness at customer j due to a disruption pursuant to scenario k of d_i. The variable is the maximum of zero and the disruption length of i minus the buffer size between the customers i and j. One important remark is that the term $\sum_{m=i+1}^{j-1}(u_m + d_m)$ is zero when i and j are two consecutive vertices in a route. Λ_{ik} is the overtime at the depot due to a disruption according to scenario k of d_i. The variable ζ is the overtime at the end of the route if no disruptions occur. If the arrival time at the depot at the end of the route is smaller than the end time of a shift, the variable ζ will be zero. If the arrival time is higher than the end time of the shift, ζ will be the difference between the arrival time at the depot and the end time of the shift.

4 Methodology

4.1 Two-Stage Solution Approach

The Vehicle Routing Problem (VRP) is an NP-hard problem. The VRP-SITW is a similar problem as the VRP, but with the inclusion of overtime and tardiness penalties. The VRP-SITW with only travel costs is equivalent to the VRP and is thus also NP-hard. Therefore with the considering of the tardiness and overtime penalties, we need a heuristic to solve this problem. We introduce the two-stage solution approach of Jabali et al. (2015) where in the first stage the routing and in the second stage the scheduling of the time windows is done. In the first stage the routes are specified and on top of that, the customers are sequenced within the routes. We use a tabu search heuristic to get the routes. For a given route, the second stage will schedule the time windows using a linear programming program. First of all, we describe the linear programming problem (second stage) to make the schedules. We discuss the second stage first, because this linear programming problem is required in each iteration of the first stage. Thereafter the tabu search heuristic (first stage) is explained, with the different procedures of the initial solution.

4.2 Scheduling

The tabu search procedure provides a set of routes, $Z = \{R_1, \ldots, R_{|Z|}\}$, to satisfy all the demand of the customers. For one given route, R_r, the linear programming problem will make the schedule of the time windows. One important remark is that the schedule is based on the earlier explained assumption of exactly one disrupted arc in a route. The objective function considers only the tardiness and overtime penalties and is therefore as follows:

$$\Theta(R_r) = \min \sum_{i=0}^{n_r} \sum_{j=i+1}^{n_r+1} \sum_{k=1}^{|\Psi_i|} p_i g_{ik} t_j \Delta_{ijk} + b \sum_{i=0}^{n_r} \sum_{k=1}^{|\Psi_i|} p_i g_{ik} \Lambda_{ik} + b\Big(1 - \sum_{i=0}^{n_r} p_i\Big)\zeta. \quad (10)$$

The objective function minimizes the tardiness and overtime penalties of a specific route R_r, as discussed in Section 3.5. The constraints of the linear programming problem are the following:

$$s_{i-1} + d_{i-1} + u_i \leq s_i, \qquad i \in R_r \backslash \{0\}, \tag{11}$$

$$s_0 \geq s_s, \tag{12}$$

$$s_i + d_i + l_{ik} + \sum_{m=i+1}^{j-1} (u_m + d_m) \leq s_j - u_j + W_j + \Delta_{ijk}, \tag{13}$$

$$i \in R_r \backslash \{n_r + 1\}; j \in R_r \backslash \{0\}; i < j; k \in \Psi_i,$$

$$s_{n_r+1} + \Delta_{i,n_r+1,k} - s_e \leq \Lambda_{ik}, \qquad i \in R_r \backslash \{n_r + 1\}; k \in \Psi_i, \tag{14}$$

$$\zeta \geq s_{n_r+1} - s_e, \tag{15}$$

$$\text{all } \Delta_{ijk} \geq 0; \text{ all } s_i \geq 0; \text{ all } \Lambda_{ik} \geq 0; \ \zeta \geq 0. \tag{16}$$

Constraints (11) ensure that the departure time of customer i is at least equal to the sum of the departure time of his predecessor, the distance from the predecessor of i to customer i itself and the service time of customer i. The buffer size between customer $i-1$ and i is equal to $s_i - (s_{i-1} + d_{i-1} + u_i)$. These buffers will make a route more robust against disruptions. Constraint (12) requires that the scheduled departure time from the depot is higher than or equal to the start time of the shift. Constraints (13) specify the terms Δ_{ijk}. The derivation of these constraints can be obtained from Equation (7) in Section 3.5. Constraints (14) and (15) specify the terms Λ_{ik} and ζ respectively. The derivation of these constraints can be obtained from Equations (8) and (9).

4.3 Tabu Search Heuristic

The tabu search procedure is commonly used for solving Vehicle Routing Problems. That is why we adopted this procedure also in the VRP-SITW. The tabu search procedure will generate a set of routes and then the linear programming problem, explained in Section 4.2, will make the schedule. In the tabu search algorithm the neighbors of the current solution are investigated to find an improvement. A specific solution can be tabu for the next certain number of iterations, in order to avoid local minima. In our case, the move will be tabu for the next κ number of iterations.

A problem arises with the computation time of this two-stage procedure. Each potential move evaluation requires a linear programming run and the total computation time will be enormous. In order to avoid these large computations times, an approximation of the tardiness and overtime penalties is made. These approximations will try to lead the tabu search to the best move in the current neighborhood. Three different criteria, C_1, C_2 and C_3, will estimate the effect of the moves in the neighborhood. These criteria are explained in Section 4.4. The best move in the neighborhood is based on the criteria and the two-stage procedure is given in Algorithm 1.

Algorithm 1: Tabu Search

```
   Input: The instance with all the customers, distances and service times
   Output: The best solution for criterion C_1, C_2 and C_3
1  construct initial solution Z_0 and compute F(Z_0)
2  function Tabu Search
3      for ξ = 1 to 3 do
4          set Z = Z_0 and F(Z) = F(Z_0)
5          generate the neighborhood of Z using the 2-opt* and Or-Opt exchanges
6          evaluate all neighbors on criterion C_ξ and retain the best non-tabu move as new solution Z
7          evaluate F(Z) and update the tabu list to include Z
8          if Z is feasible and is better than the current best solution then
9              update the best feasible solution for C_ξ to Z
10         update excess demand penalty
11         if no improvement in η_max iterations then
12             store best solution for C_ξ
13         else
14             go to line 5
15     return the best solution from ξ = 1, 2 and 3
```

The initial solution Z_0 is an important step in our heuristic. In Section 4.5 the different procedures in order to find the initial solution will be explained. Both the excess demand penalty in line 10 and the different criteria will be discussed in Section 4.4.

In the article of Jabali et al. (2015) the η_{max} parameter is not specified. Therefore we estimate a suitable value for this parameter using the benchmark instances. We will randomly select a number of instances and test different values of η_{max}. The test values are sequenced in an ascending order. If there is a test value for which no reduction in the objective values $F(Z)$ of the randomly chosen instances is realized compared to the previous test value of η_{max}, we set η_{max} to this value.

Another main task of this procedure is the definition of the neighborhood. For each customer $i \in V$, the neighbors are created by making the closest η customers available for a move. In order to generate the neighbors, two exchange procedures are used. The first procedure is the 2-opt* neighborhood, as introduced by Potvin and Rousseau (1995). They argued that classical k-opt exchange heuristics are not well adapted to problems with time windows. However, the 2-opt* exchange is powerful for problems with these time windows because it introduces the last customers of a certain route at the end of the first customers of another route. The algorithm of creating the neighborhood of a current solution Z using 2-opt* exchanges is given in Appendix A Algorithm 5.

The second neighborhood is found by an Or-Opt exchange, as used by Potvin and Rousseau (1995). They introduced an Or-Opt-1 exchange procedure where a customer is inserted at another location to try to improve the current solution. The Or-Opt exchange considers not only one customer, but inserts also two and three adjacent customers at another location. The procedure is represented in Appendix A Algorithm 6.

4.4 Criteria

Another important ingredient of the tabu search procedure are the three criteria in order to avoid large computation times. Before introducing these criteria, another concept has to be explained. The total travel costs associated with the set of routes Z are the following:

$$\Omega(Z) = c \sum_{R_r \in Z} \sum_{(i,j) \in R_r} d_{ij}. \tag{17}$$

Gendreau et al. (1994) create a concept to diverse the search in the heuristic. This diversification is achieved by allowing demand-infeasible solutions. In these solutions the total demand of the customers in a specific route may exceed the vehicle capacity. The excess demand is penalized and the objective function in Equation (17) is replaced by the following equation:

$$\Omega_2(Z) = \Omega(Z) + w \sum_{R_r \in Z} \left[\left(\sum_{i \in R_r} q_i - Q \right) \right]^+. \tag{18}$$

In this above equation each unit of excess demand in a route is penalized by a parameter w. In order to make the search even more diverse, the demand-infeasible penalty parameter is not a constant. The penalty is decreased by multiplication with a constant v after ϕ consecutive feasible iterations. The search will be guided to infeasible solutions due to this decrease in the penalty parameter. On the other hand, the penalty is increased by multiplication with the inverse of v after ϕ consecutive infeasible iterations. When the algorithm realized a sequence of infeasible solutions, the increase in the penalty parameter will guide the heuristic to feasible solutions.

After introducing this concept of diversification of the search we are able to introduce the three different criteria. These criteria avoid the heuristic to execute a substantial number of linear programming runs and estimate the objective values of the possible moves.

The first criterion is the C_1-distance based criterion. This criterion does not take into account the time windows and the overtime and tardiness penalties, but focuses purely on the minimization of the traveled distance. Let Z' be a neighbor of the solution Z and define $\Delta_2(Z') = \Omega_2(Z) - \Omega_2(Z')$. The chosen move is the one that is not tabu and maximizes the expression $\Delta_2(Z')$.

The second criterion is the C_2-distance based criterion with marginal penalties. This criterion is the same as C_1 but with an added assessment of the penalty component $\sum_{R_r \in Z} \Theta(R_r)$. For a specific solution Z, the marginal penalty of route R_r is $\frac{\Theta(R_r)}{n_r+1}$. Let there be an exchange of customers between two routes, R_1 and R_2, and define the new solution as Z'. The number of customers in these routes possibly changed and therefore let n_1 be the number of customers in route R_1 and n_2 in route R_2 of solution Z. Let n_1' and n_2' be the number of customers in R_1' and R_2' of the new solution Z' respectively. C_2 chooses the move which maximizes the following equation:

$$\Delta_2(Z') = \Omega_2(Z) - \Omega_2(Z') + \rho \left[\Theta(R_1) + \Theta(R_2) - \frac{\Theta(R_1)}{n_1+1}(n_1'+1) - \frac{\Theta(R_2)}{n_2+1}(n_2'+1) \right]. \tag{19}$$

When a given route has an increase in the number of customers, the penalties are more likely to be high. Similarly, when a certain route has a decrease in the number of customers, the penalties are likely to decrease and therefore this criteria accounts for the delay penalties. When there is an exchange of customers within one route, the last term of Equation (19) cancels out and the C_2 criterion is exactly the same as the C_1 criterion.

The last criterion is the C_3-distance and buffer based criterion. The buffer size between customer i and customer *i+1* in a route R_r is equal to $bu(i) = s_{i+1} - (s_i + d_i + u_{i+1})$. The C_3

criterion regards it beneficial to take moves with small buffers. When the buffer sizes are small, an improvement in travel times is likely to decrease the penalties as well. Hence, for a solution Z' where a move between customer i and customer j is realized, the following quantity will be computed:

$$\Delta_3(Z') = \Omega_2(Z) - \Omega_2(Z') - \gamma[bu(i) + bu(j)]. \tag{20}$$

The best neighbor of the current solution Z based on the C_3 criterion will be the solution that maximizes $\Delta_3(Z')$.

4.5 Initial Solution

In a heuristic approach the initial solution is of great importance to the final solution. The initial solution will guide a heuristic to a certain search area and another starting point is likely to have another search area. In Algorithm 1 line 1 we use different initial solutions, in order to investigate the influence of the initial solution.

The first initial solution is obtained by the nearest neighbor procedure. In this procedure the routes are generated by searching for the nearest neighbor of a specific customer and add this customer to the current route, if this will generate a feasible route. If the route is not feasible, the route will return to the depot and a new route will be created. The exact procedure is as follows:

Algorithm 2: Nearest Neighbor Heuristic

```
   Input: The set of N customers, the distances between the customers and the demand of each customer
   Output: A set of routes Z
 1 function Nearest Neighbor Heuristic
 2     initialize an empty set Z
 3     initialize an empty set L
 4     initialize a list R and add zero to R
 5     initialize the variable edit to zero
 6     while the size of set L is smaller than the number of customers do
 7         Find the nearest neighbor of the customer edit who is not in set L
 8         if the route R obtained by adding the nearest neighbor is feasible then
 9             add the nearest neighbor to R and set edit to the nearest neighbor
10             add the nearest neighbor to the set L
11         else
12             add zero at the end of R
13             add R to the set Z
14             clear R and add zero to the list
15             set variable edit to zero
16     return the set of routes Z
```

The second initial solution is obtained by the savings heuristic, used by for example Laporte (2009). This procedure starts with only back-and-forth routes. In each iteration, a merge of routes is tried by maximizing the saving $s_{ij} = c_{0i} + c_{0j} - c_{ij}$. Algorithm 3 shows the exact procedure.

Algorithm 3: Savings Heuristic

Input: The set of N customers, the distances between customers and the demand of each customer
Output: A set of routes Z

function Savings Heuristic
  take Z the set of $|N|$ initial routes with $C_i = \{0, i, 0\}$, $i \in N$
  initialize an empty set L
  for *all customers* $i \in N$ **do**
    for *all customers* $j \in N$*:* $j > i$ **do**
      compute the saving $s_{ij} = d_{0i} + d_{0j} - d_{ij}$
      add the saving s_{ij} to the list L
  sort the list L in a nonincreasing order
  for *all savings in the list* L **do**
    extract and remove from the top of list L the saving s_{ij}
    if *customers* i *and* j *belong to two separate routes and both are directed linked to the depot* **then**
      if *route obtained by replacing (0. i) and (0,j) with (i,j) is feasible* **then**
        remove the routes of i and j from Z
        merge the routes of customer i and j and add the route to Z
  return *the set of routes* Z

The last procedure in order to find an initial solution is the sweep heuristic, as introduced by Gillet and Miller (1974). In this approach first the polar-coordinate angle for each customer is determined. Thereafter the routing is based on these polar-coordinates. This is done by rotating a half-line rooted at the depot, adding customers to the route until the route is not feasible anymore. The procedure is given in Algorithm 4.

Algorithm 4: Sweep Heuristic

Input: The set of N customers, the distances between customers and the demand of each customer
Output: A set of routes Z

function Sweep Heuristic
  initialize an empty set Z
  initialize an empty list P
  for *all customers* $i \in N$ **do**
    calculate the polar-coordinate angle of customer i
    add the polar-coordinate angle to the P
  sort the list P in nondecreasing order
  initialize a list R and add zero to R
  while *the size of list* P *is greater than zero* **do**
    extract from the top of P the polar-coordinate angle p and find corresponding customer j
    if *the route* R *obtained by adding customer* j *is feasible* **then**
      add customer j to R
      remove polar-coordinate angle p from P
    else
      add zero at the end of R
      add R to the set Z
      clear R and add zero to the list
  return *the set of routes* Z

5 Data

To asses the computational results of the VRP-SITW we have run a number of experiments on benchmark instances. The first dataset contains 27 instances from Augerat et al. (1998). The number of customers is varying between 31 and 79 in these instances and the coordinates of the locations of the customers are given. The vehicle capacity Q is equal to 100 units and the service times u_i are the same for each customer, namely 10 minutes. The time window length W_i is equal to 60 minutes for each customer and the start time and end time of a shift are 0 and 200, respectively. The second dataset contains 29 instances from Solomon (1987). Some of these instances are random (R), some are clustered (C) and the remaining instances are random and clustered (RC). All the instances contain 100 customers and the service times, window lengths and the coordinates of the customers are given in the datasets of these instances. The opening hours of the depot are used as the start and end time of a shift and the vehicle capacity Q equals 200 units.

In order to get a good solution for the VRP-SITW, we need to specify the used parameters. The travel cost of traveling one-unit of distance, c, is equal to one and therefore the travel time is equivalent to the distance. The number of customers in a specific benchmark instance is equal to N and for each customer, the closest $\eta = \lceil 0.3N \rceil$ customers are allowed to make a move in the neighborhood search. The tenure size κ of the tabu search equals 20. The initial tardiness penalty t_i equals the value 5 for each customer $i \in N$. The overtime penalty b is equal to 2. The infeasibility penalty w is equal to 12 and can be altered by the parameters $\phi = 5$ and $v = \frac{3}{4}$. The penalty parameters for the C_2 and C_3 criterion are respectively $\rho = 1$ and $\gamma = 0.1$. The parameter η_{max} is not specified in the article of Jabali et al. (2015). We estimate a suitable value for this parameter, as discussed in Section 4.3.

To model the disruptions the probability of having a delay on arc $(i, i+1)$, with $i, i+1 \in R_r$, have to be specified. The probability p_i is equal to one over the number of customers in a route and the number of vehicles needed to satisfy all the demand of the customers, thus $p_i = \frac{1}{N+k}$. Hence, the probability of having a delay is identical for each arc. One important remark is the fact that k may not exceed the available number of vehicles, so $k \leq K$. We introduce four different delay scenarios, so $|\Psi_i| = 4$. The probabilities of having a specific scenario are equal to $g_{i1} = 0.5$, $g_{i2} = 0.3$, $g_{i3} = 0.1$ and $g_{i4} = 0.1$. Hence, the probability of having a specific delay are also the same for each arc. Lastly, the disruption lengths of the different scenarios are based on the distance between the customers, namely $l_{i1} = 0.1d_{ij}$, $l_{i2} = 0.2d_{ij}$, $l_{i3} = 0.5d_{ij}$ and $l_{i4} = d_{ij}$.

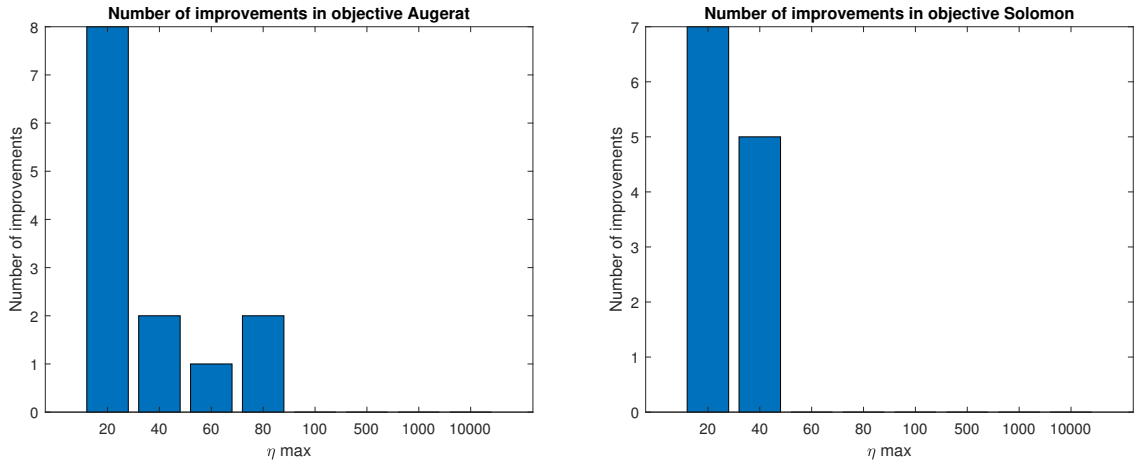
6 Computational Results

The experiments are all performed on an Intel Core with 2.0 GHz and 4 GB of RAM. The software used in order to solve the tabu search procedure is *JAVA* and the linear programming problems to schedule the time windows are solved by using *JAVA CPLEX*.

First we estimate the parameter η_{max} using the benchmark instances. On top of that, we execute some experiments to look into the effect of using the different criteria. Thereafter experiments are executed to explore the tardiness choices. Furthermore, the effect of the initial solution on the obtained costs is measured. Finally, the solutions of the VRP-SITW are compared to the solutions of the VRP and in addition to this, a comparison is made between the solutions of the VRP-SITW and the VRPTW. In all these experiments the initial solution is specified by the nearest neighbor procedure, unless stated otherwise.

6.1 Number of Consecutive Iterations without Update

In order to determine a suitable value for η_{max}, the number of consecutive iterations without update, we randomly took 10 Augerat instances and 10 Solomon instances. For each instance one of the three criteria is randomly chosen to obtain the set of routes. We tested different values for η_{max} in an increasing order. The sequence of test values for η_{max} is as follows: (1, 20, 40, 60, 80, 100, 500, 1,000, 10,000). The objective $F(Z)$ of each instance using a certain test value is compared to the objective value realized by using the previous test value in the sequence. When there is an improvement in $F(Z)$, it means that increasing the number of consecutive iterations without update leads to an improvement in the objective value. Figure 1a indicates the number of improvements in the objectives of the randomly chosen Augerat instances with comparing the current test value of η_{max} to the previous one. Figure 1b gives the same comparison for the Solomon instances. One important remark is the fact that the objectives obtained by $\eta_{max} = 20$ are compared to the objectives realized by the test value of $\eta_{max} = 1$. In the Augerat instances for example, there are 8 improvements in the objective using $\eta_{max} = 20$ instead of $\eta_{max} = 1$.



(a) Augerat instances

(b) Solomon instances

Figure 1: Number of improvements in the objectives

The Figures 1a en 1b indicate that the last improvements in objective are obtained with the test value of 80. Since we do not check every instance, we set $\eta_{max} = 100$ in order to get more certainty. We continue with this value in the upcoming experiments.

6.2 Move Selection

Table 1 shows the results of the Augerat instances by using only one criterion. Either the C_1, C_2 or C_3 criterion is used in these experiments. The left hand side of the table displays the obtained objective values $F(Z)$, as introduced in Equation (1) in Section 3.2. The right hand side of the table gives the running times. The first important observation is the fact that using C_3 leads in 18 out of the 27 instances to the best objective value $F(Z)$. For C_1 this is also the case in 18 out of the 27 instances and for C_2 in 10 out of the 27 instances. One important remark is the fact that for some instances the same objectives are obtained by using different criteria. The table indicates that the objective values obtained by using the first and third criterion are the same in a substantial number of instances. A reason for this could be small buffer sizes for customer i and j in Equation (20), which makes criteria C_1 and C_3 similar. Besides this observation, the same objective values are also obtained by using C_1 and C_2 in a certain number of instances. When exchanges occur in the same route, the C_1 and C_2 criteria are equivalent and this could be the reason for the same objectives for these two criteria. Another reason could be the dominant influence of the distance in Equation (19) of criterion C_2. The last term of this equation, which considers the penalties, can be dominated by the distance term and therefore no differences between the costs of the two criteria are obtained. The running times of the different criteria are approximately the same. Because the running times of the different criteria are roughly similar and no criterion obtains clearly better results for the given benchmark instances of Augerat, there is no clear preference for one of the criteria. However, the results obtained by criteria C_1 and C_3 are somewhat better than the results of criterion C_2 for the Augerat instances.

Table 1: Comparison of the three move selection criteria for the Augerat instances

Instance	Objective Value			CPU time (s)			Total
	C_1	C_2	C_3	C_1	C_2	C_3	
32k5	**1405.4**	1427.3	**1405.4**	5.4	5.2	5.3	16.0
33k5	**888.6**	**888.6**	**888.6**	4.3	3.9	4.0	12.2
33k6	**985.0**	1011.4	**985.0**	4.9	5.0	4.1	14.1
34k5	**1074.6**	**1074.6**	**1074.6**	3.8	3.5	3.7	11.0
36k5	**1411.3**	1412.3	1465.7	5.5	5.9	5.1	16.5
37k5	**972.0**	**972.0**	**972.0**	4.1	3.9	4.7	12.7
37k6	1478.0	**1416.2**	1548.5	6.1	6.0	6.3	18.3
38k5	**1065.2**	**1065.2**	**1065.2**	4.3	4.2	4.4	12.9
39k5	1519.4	**1471.0**	1519.4	4.8	7.8	4.8	17.3
39k6	1340.0	**1259.5**	1340.0	4.7	5.6	5.0	15.3
44k6	**1558.8**	**1558.8**	**1558.8**	5.5	5.6	5.6	16.6
45k6	**1386.6**	1448.0	**1386.6**	6.8	7.9	7.2	21.9
45k7	2003.7	**1893.7**	1959.0	7.5	8.9	10.7	27.1
46k7	**1326.1**	1330.8	**1326.1**	6.8	7.5	6.6	21.0
48k7	1957.3	1852.9	**1836.9**	8.6	21.3	9.0	38.9
53k7	**1570.6**	1663.1	**1570.6**	8.7	7.2	8.7	24.5
54k7	2004.3	2003.8	**1986.7**	10.0	12.7	11.8	34.5
55k9	**1234.8**	1239.9	1253.8	7.3	7.5	7.4	22.2
60k9	2215.0	**2102.3**	2105.7	12.0	31.9	20.2	64.2
61k9	**1312.2**	1319.3	**1312.2**	9.6	10.0	9.7	29.3
62k8	**2270.7**	2332.7	2279.2	16.6	16.9	27.1	60.7
63k10	**1881.5**	2013.0	1941.4	13.3	23.9	13.1	50.3
63k9	3086.8	3020.5	**2989.2**	20.4	24.9	25.4	70.7
64k9	**2374.3**	2545.2	**2374.3**	22.0	13.8	12.2	48.0
65k9	**1547.0**	1592.2	**1547.0**	21.7	11.9	13.8	47.3
69k9	**1681.9**	1744.9	**1681.9**	17.7	10.2	16.6	44.5
80k10	3616.8	3552.9	**3494.3**	28.9	54.9	34.6	118.4
Arithmetic Average				10.0	12.1	10.6	32.8
Geometric Average				8.3	9.2	8.6	26.2

This table represents the objective values and running times of the Augerat instances using the C_1, C_2 and C_3 criterion.

Table 2 contains the objective values $F(Z)$ and the running time of the Solomon instances. Again the instances are tested with each criterion separately. The left hand side of the table gives the objective values obtained by criteria C_1, C_2 and C_3 respectively and the right hand side gives the running time. Criterion C_1 outperforms the other criteria in 26 out of the 29 instances and C_3 in 24 out of the 29 instances. Criterion C_2 realizes the best objective in 9 of the instances. The different criteria give in a substantial number of benchmark instances the same objective value. This may be due to the earlier discussed small buffer sizes and the exchanging of customers on a single route. The running times of the Solomon instances are

relatively high compared to the running times of the Augerat instances. The reason for this is the higher number of customers. For the Solomon instances, criteria C_1 and C_3 are beneficial because these criteria outperform criterion C_2 in almost all the instances and the running times of the different criteria are similar.

Table 2: Comparison of the three move selection criteria for the Solomon instances

Instance	Objective Value			CPU time (s)			Total
	C_1	C_2	C_3	C_1	C_2	C_3	
R101	1113.7	1122.8	**1103.3**	24.3	25.1	25.6	74.9
R102	1113.6	1122.7	**1097.9**	23.9	25.6	25.8	75.3
R103	1113.2	1122.2	**1087.0**	34.0	24.6	28.9	87.4
R104	**1112.8**	1121.8	1113.3	23.5	30.0	30.8	84.4
R105	**1112.5**	1121.5	**1112.5**	25.1	24.0	37.1	86.2
R106	**1112.5**	1121.5	**1112.5**	26.7	26.6	25.3	78.5
R107	**1112.5**	1121.5	1113.0	25.0	25.8	22.0	72.8
R108	**1112.5**	1121.5	1113.0	22.1	21.6	22.3	66.0
R109	**1112.5**	1121.5	**1112.5**	29.0	22.1	22.1	73.2
R110	**1112.5**	1121.5	**1112.5**	23.9	23.1	28.2	75.2
R111	**1112.5**	1121.5	1113.0	27.8	22.4	23.4	73.6
R112	**1112.5**	1121.5	**1112.5**	25.2	20.4	21.8	67.4
C101	**834.2**	**834.2**	**834.2**	24.8	31.3	31.5	87.6
C102	**834.2**	**834.2**	**834.2**	29.7	32.6	31.1	93.4
C103	**834.2**	**834.2**	**834.2**	28.0	29.7	29.0	86.7
C104	**834.2**	**834.2**	**834.2**	34.1	29.6	31.1	94.8
C105	**834.2**	**834.2**	**834.2**	34.9	32.0	32.4	99.3
C106	**834.2**	**834.2**	**834.2**	35.3	32.2	38.0	105.5
C107	**834.2**	**834.2**	**834.2**	28.4	27.9	31.5	87.7
C108	**834.2**	**834.2**	**834.2**	28.8	31.7	31.6	92.1
C109	**834.2**	**834.2**	**834.2**	31.7	38.8	38.0	108.5
RC101	**1105.7**	1230.7	**1105.7**	34.6	40.8	37.9	113.3
RC102	**1105.7**	1230.7	**1105.7**	27.6	32.7	35.0	95.3
RC103	**1105.7**	1230.7	**1105.7**	33.9	53.5	34.6	122.0
RC104	**1105.7**	1230.7	**1105.7**	34.3	43.7	33.4	111.4
RC105	**1106.2**	1231.1	1131.6	28.0	47.5	41.5	117.1
RC106	**1105.7**	1230.7	**1105.7**	29.0	42.1	31.6	102.7
RC107	**1105.7**	1230.7	**1105.7**	33.5	33.3	33.7	100.5
RC108	**1105.7**	1230.7	**1105.7**	29.6	42.7	32.8	105.0
Arithmetic Average				28.8	31.5	30.6	91.0
Geometric Average				28.6	30.5	30.2	89.3

This table represents the objective values and running times of the Solomon instances using the C_1, C_2 and C_3 criterion.

Considering Table 1 and 2, the obtained results are not in line with the article of Jabali et al. (2015). The obtained objectives $F(Z)$ of the Augerat and the Solomon instances are on average 36.7% and 12.3% higher. The obtained differences are not caused by the specification of the η_{max}, the number of consecutive iterations without update, because a higher value of η_{max} will not lead to a reduction in the objective. A possible explanation could be another implementation of for example the neighborhood search or the nearest neighbor heuristic.

The implementations of the nearest neighbor heuristic and the neighborhood search are not given in the article of Jabali et al. (2015). The implementation of these procedures could have a significant effect on the obtained objectives. Therefore we added the concept of a maximum number of customers per route in the nearest neighbor heuristic of Section 4.5 and explore the effects on the obtained costs. A maximum number of customers per route could avoid long routes and thus the delay penalties. We used the Solomon instances and set the maximum number of customers per route equal to 10, because routes with more than 10 customers are more likely to have substantial penalties. The results are represented in Appendix B Table 6.

For the R and C instances, the average objective $F(Z)$ remains approximately the same. In the R instance a mild increase is acquired, whereas in the C instance a small decrease is achieved. However, the average objective of the C instances increased with 12.5% compared to the nearest neighbor heuristic without the concept of the maximum number of customers per route. (Algorithm 2). These results indicate that the implementation of the nearest neighbor heuristic has a significant effect on the obtained costs. These results do not necessarily proof that the differences between this article and the article of Jabali et al. (2015) are due to the implementation of the heuristic. However, the implementation of the nearest neighbor heuristic could be one cause for the differences.

6.3 Choice of the Tardiness Penalty

To get some intuition about the effects of the tardiness penalty, we will make some variations in the penalty costs t_i. Four different penalty costs are introduced. The first penalty cost setting is P5 and the costs are the same as in the previous experiments in Section 6.2. Therefore, in P5 the penalty cost is $t_i = 5$, $\forall i \in V\backslash\{0\}$, where in P10 the penalty cost t_i is equal to 10. In the third cost setting Prop, the delay cost of a customer is equal to the quantity of his order, thus $t_i = q_i$, $\forall i \in V\backslash\{0\}$. In this setting the penalty is proportional to the demand of the customer. In the last penalty setting, 1.3dist, the penalty cost is again $t_i = 5$. However, the distances between the customers are increased by 30%. This leads to smaller buffer sizes between the customers and less flexibility in the schedules.

Table 3 indicates the results for the two-stage procedure with the different penalty settings. The left hand side provides the objective values $F(Z)$ for the instances of Augerat with the different penalty settings. The value $M(C_i)$ indicates the number of times that criterion C_i has the best objective value for the given penalty cost setting. For the cost settings P10 and 1.3dist criterion C_3 performs the best, whereas in cost setting Prop criterion C_1 gives the best results. For the cost setting P5 the criteria C_1 and C_3 are the best performing criteria. Overall criterion C_3 indicates the best results and a possible explanation could be the assessment of the buffer sizes in this criterion.

In the cost setting P10, the tardiness penalty is doubled compared to the setting P5. However, the average objective value of P10 is only 0.3% higher than the average objective of P5. The average objective value of P5 and Prop are approximately the same. Considering these results we can conclude that varying the penalty costs does not influence the objective values significantly. In the last cost setting, 1.3dist, the average objective increase is 54.1% compared to the cost setting P5, whereas the distance increase is only 30%. A possible explanation for this phenomenon is the fact that when the distances increase, the buffer sizes will decrease and therefore higher penalties are obtained due to the possible disruptions.

The right hand side of the table represent the penalty ratio, which is defined as follows:

$$\sum_{R_r \in Z} \frac{\Theta(R_r)}{F(Z)}. \tag{21}$$

The penalty ratio is the proportion of the objective value corresponding to the overtime and tardiness penalties. Table 3 indicates that the first three penalty settings have approximately the same average penalty ratio, while the average penalty ratio of the 1.3dist setting is by far the largest. Therefore, we conclude that an increase in distance will lead to an increase in the delay penalties.

Table 3: Results for the Augerat instances with four different penalty settings

Instance	Objective				Penalty Ratio			
	P5	P10	Prop	1.3dist	P5 (%)	P10 (%)	Prop (%)	1.3dist (%)
32k5	1405.4	1407.8	1403.0	2084.9	40.0	40.1	39.9	46.3
33k5	888.6	888.9	889.1	1448.6	22.1	22.2	22.2	38.2
33k6	985.0	985.0	985.0	1525.4	20.1	20.1	20.1	29.7
34k5	1074.6	1075.4	1075.5	1683.2	9.0	9.1	9.1	38.8
36k5	1411.3	1412.8	1410.1	2181.0	35.5	34.8	35.4	48.4
37k5	972.0	972.0	972.0	1672.7	5.3	5.3	5.3	40.4
37k6	1416.2	1417.8	1414.9	2247.6	27.6	27.7	27.6	37.7
38k5	1065.2	1065.5	1065.6	1564.2	17.5	17.5	17.5	34.9
39k5	1471.0	1471.4	1471.2	2219.0	40.2	40.2	40.2	48.6
39k6	1259.5	1259.5	1259.4	2040.4	26.3	26.3	26.3	41.1
44k6	1558.8	1561.4	1558.3	2310.4	17.1	17.2	17.0	44.9
45k6	1386.6	1387.1	1386.1	2085.4	26.7	26.7	26.6	34.9
45k7	1893.7	1964.9	1890.8	2922.2	36.7	39.1	37.5	46.5
46k7	1326.1	1326.2	1326.3	2063.3	26.1	26.2	26.2	38.5
48k7	1836.9	1838.7	1836.4	2824.2	32.7	32.8	32.7	43.8
53k7	1570.6	1571.6	1569.7	2467.7	30.8	30.9	30.8	41.9
54k7	1986.7	1990.4	1983.6	3102.7	39.1	39.3	39.1	49.6
55k9	1234.8	1235.1	1234.4	1994.4	6.6	6.6	6.5	22.4
60k9	2102.3	2106.3	2098.4	3227.7	33.8	33.9	33.7	44.1
61k9	1312.2	1312.2	1312.2	1991.1	16.6	16.6	16.6	28.7
62k8	2270.7	2275.6	2266.4	3455.3	40.4	40.6	40.3	49.3
63k10	1881.5	1882.8	1881.9	2929.4	27.4	27.4	27.4	38.2
63k9	2989.2	2998.5	3012.0	4507.2	42.7	42.9	42.9	51.0
64k9	2374.3	2377.5	2371.4	3522.1	38.7	38.8	38.7	46.7
65k9	1547.0	1547.0	1547.0	2555.1	21.3	21.3	21.3	36.8
69k9	1681.9	1681.9	1681.9	2556.4	27.2	27.2	27.2	36.3
80k10	3494.3	3502.1	3486.7	5249.6	45.6	45.7	45.4	53.0
Average Penalty (%)					30.8	30.9	30.8	42.8
M(C_1)	18	17	18	10				
M(C_2)	10	10	10	10				
M(C_3)	18	18	17	13				

This table displays the objective values and penalty ratios of the Augerat instances using different penalty settings, as explained in Section 6.3.

6.4 Initial Solution Choice

Table 4 represents the obtained results for the Solomon instances using three different procedures for the initial solution of Algorithm 1. The left hand side of the table represents the objective values $F(Z)$ obtained by using the nearest neighbor, savings and sweep heuristic for the initial solution. The savings heuristic outperforms the other heuristics in the R1 and RC1 instances, whereas the nearest neighbor heuristic outperforms the other heuristics in the C1 instances. Another important observation is the fact that the C_2 criterion performs best in the savings heuristic. However, in the other heuristics the C_2 criterion performs worst.

The right hand side of the table indicates the running times of the heuristics. On average, the savings heuristic has the lowest running time. The running time of the nearest neighbor heuristic is on average only 11% higher. The sweep heuristic has a substantially higher running time than the savings heuristic, namely an increase of 72%. Considering the objective values and the running times, we can conclude that the procedure for the initial solution and the initial solution itself have a significant effect on the obtained objective value and the running time.

Table 4: Results of the Solomon instances with different initial solution procedures

Instance	Objective Value			CPU time (s)		
	Nearest Neighbor	Savings	Sweep	Nearest Neighbor	Savings	Sweep
R101	1103.3	**977.7**	1104.4	25.6	20.4	43.1
R102	1097.9	**977.2**	1077.8	25.8	19.3	38.9
R103	1087.0	**975.8**	1102.4	28.9	27.3	45.2
R104	1112.8	**974.7**	1179.9	23.5	17.7	49.0
R105	1112.5	**974.1**	1232.4	31.1	20.9	49.1
R106	1112.5	**974.1**	1263.0	26.0	25.9	45.6
R107	1112.5	**974.1**	1263.0	25.0	17.3	48.1
R108	1112.5	**974.1**	1263.0	22.1	18.6	38.0
R109	1112.5	**974.1**	1263.0	25.5	21.6	47.0
R110	1112.5	**974.1**	1158.8	26.0	22.3	45.7
R111	1112.5	**974.1**	1150.5	27.8	21.9	46.2
R112	1112.5	**974.1**	1263.0	23.5	21.1	44.9
C101	**834.2**	850.2	840.9	29.2	20.8	38.8
C102	**834.2**	850.2	840.9	31.1	31.0	39.8
C103	**834.2**	850.2	840.9	28.9	22.4	42.0
C104	**834.2**	850.2	840.9	31.6	27.8	42.5
C105	**834.2**	850.2	840.9	33.1	22.1	42.8
C106	**834.2**	850.2	840.9	35.2	29.6	47.3
C107	**834.2**	850.2	840.9	29.2	21.3	44.6
C108	**834.2**	850.2	840.9	30.7	20.9	45.7
C109	**834.2**	850.2	840.9	36.2	28.1	40.7
RC101	1105.7	**1096.5**	1106.4	36.3	34.8	36.5
RC102	1105.7	**1096.2**	1106.4	31.3	37.0	63.1
RC103	1105.7	**1096.0**	1099.9	34.3	35.8	74.1
RC104	1105.7	**1095.7**	1104.1	33.9	38.2	64.4
RC105	1106.2	**1097.4**	1107.0	28.0	36.7	41.7
RC106	1105.7	**1095.6**	1106.4	30.3	38.4	40.5
RC107	1105.7	**1095.6**	1106.4	33.6	33.0	43.4
RC108	1105.7	**1095.6**	1106.4	31.2	35.7	34.6
Average CPU time (s)				29.5	26.5	45.6
M(C_1)	26	13	18			
M(C_2)	9	29	0			
M(C_3)	24	14	20			

This table represents the objective values obtained by using the nearest neighbor, savings and sweep heuristic in order to find the initial solution of the tabu search procedure. In addition to this, the running times are given.

6.5 VRP-SITW versus VRP

The introduction of self-imposed time windows into the vehicle routing problem may influence the total distance traveled. Therefore, we compare the distances obtained by the VRP-SITW with the distances obtained by the VRP. This experiment is executed on the Augerat instances and the VRP solutions are obtained from Ralphs (2010). The results are represented in Table 7 in Appendix B. The incorporation of the self-imposed time windows increased the average traveled distance with approximately 9% in the cost settings P5, P10 and Prop. For the 1.3dist setting the VRP distances are increased by 30% and the VRP-SITW distances are approximately 7% higher. Hence, the incorporation of self-imposed time windows does influence the traveled distance mildly. A carrier company has to decide whether it is beneficial to incorporate self-imposed time windows. The company has to weigh the costs of traveling additional distance against an increase in customer service level due to the self-imposed time windows.

6.6 VRP-SITW versus VRPTW

A company can choose to let their customers decide on the time windows for delivery. In this section the benefits of self-imposed time windows over the exogenous time windows will be evaluated. We compare the results of the VRP-SITW Solomon instances with the results of the VRPTW obtained by Solomon (2010). In the VRPTW the time windows are set by the customer. Table 5 represents the results. T_F is the travel time or distance of the VRPTW and T_S for the VRP-SITW. The number of vehicles used in the VRPTW is represented by K_F and in the VRP-SITW by K_S. The third column indicates that the travel time of the R1 and RC1 instances reduced using the self-imposed time windows. These instances have tight time windows, as visible from the penalty ratios and therefore the self-imposing of these time windows reduces the travel time. For the C1 instances the travel time slightly increases. The last column of the table indicates that these instances have a penalty ratio of zero. These instances have more flexibility in their time windows and the self-imposing does not influence the travel time

drastically. Column 4 of the table displays the decrease in the number of vehicles. The R1 and RC1 instances have a substantial reduction in the number of used vehicles, whereas in the C1 instances no reductions are achieved. In conclusion, the instances that allow a substantial reduction in travel time are also suitable for improvements in the number of used vehicles. And on average, the traveled distance will decrease with using self-imposed time windows compared to time windows set by the customers.

Table 5: Comparison of VRP-SITW with the VRPTW solutions for the Solomon instances

Instance	T_F	T_S / T_F (%)	K_F	K_F - K_S	$\sum_{R_r \in Z} \Theta(R_r)$ / $F(Z)$ (%)
R101	1637.7	62.5	20	10	7.3
R102	1466.6	69.4	18	8	7.3
R103	1208.7	81.7	14	4	9.2
R104	971.5	106.5	11	1	7.1
R105	1355.3	76.3	15	5	7.0
R106	1252.0	82.6	12	2	7.0
R107	1064.6	97.1	11	1	7.0
R108	960.9	107.6	9	-1	7.0
R109	1146.9	90.2	13	3	7.0
R110	1068.0	96.8	12	2	7.0
R111	1048.7	98.6	12	2	7.0
R112	982.1	105.3	9	-1	7.0
C101	827.3	100.8	10	0	0.0
C102	827.3	100.8	10	0	0.0
C103	826.3	101.0	10	0	0.0
C104	822.9	101.4	10	0	0.0
C105	827.3	100.8	10	0	0.0
C106	827.3	100.8	10	0	0.0
C107	827.3	100.8	10	0	0.0
C108	827.3	100.8	10	0	0.0
C109	827.3	100.8	10	0	0.0
RC101	1619.8	62.3	15	6	8.8
RC102	1457.4	69.2	14	5	8.8
RC103	1258.0	80.2	13	4	8.8
RC104	1261.7	80.0	11	2	8.8
RC105	1513.7	66.7	15	6	8.8
RC106	1424.7	70.8	11	2	8.8
RC107	1207.8	83.5	12	3	8.8
Average		89.1			5.3

This table displays the comparison between the VRP-SITW and the VRPTW solutions for the Solomon instances.

7 Conclusion

In this article we analyzed the routing of the small-package carrier companies. The routing decisions are combined with the scheduling of self-imposed time windows. The companies set the time windows of delivery for its customers. The so-called VRP-SITW is an extension of the classical VRP. Firstly, by including the self-imposed time windows and secondly by adding stochasticity into the travel times. Due to possible disruptions, a delay can occur between specific customers. The VRP-SITW is a relaxed variation of the VRPTW. In the VRPTW the time windows are exogenous constraints imposed by the customer itself.

The VRP-SITW is an NP-hard problem and we introduced the two-stage solution approach used by Jabali et al. (2015). In the first stage the routing and sequencing of the customers within a route is done by using a tabu search procedure. In the second stage the scheduling of the time windows is done by using a linear programming model that inserts buffers into the schedules. This buffer allocation model has an assumption of only one disrupted arc per route. To avoid large computation times, we used three different criteria in order to guide the tabu search heuristic in the right direction.

To test our algorithm we run some experiments on instances with 31 to 100 customers. A first result was that the criterion based on the distance and the criterion based on distance and buffer sizes are performing relatively better than the criterion based on the distance and marginal penalties. However, there are instances where the criterion based on the distance and the marginal penalties is outperforming the other criteria.

In addition to this result, we explored the influence of the penalty costs. The result is that the value of the penalty costs is less relevant for the penalty ratio compared to an increase in distance. When we increase the distance, the penalty ratio increases substantially. However, by varying the value of the penalty costs, the penalty ratios remain approximately similar.

Furthermore, the effect of the initial solution of our algorithm on the obtained operational and customer service costs is explored. Using different procedures we obtain a substantial effect of the initial solution on the obtained costs and running times. This is a limitation of our research, because it would be ideal to get similar costs for a given instance, irrespectively of the initial solution.

Another main aspect of this article is the comparison of the VRP-SITW distances to the VRP distances. The results indicate that the VRP-SITW requires only a mild average increase in distance. Another comparison is made between the VRP-SITW and the VRPTW. The results of these two problems show substantial differences. The VRP-SITW uses significantly less vehicles and even more important significantly less distance. This is due to the flexibility in setting time windows in the VRP-SITW.

The VRP-SITW and its two-stage solution approach may be beneficial to the small-package carrier companies. A first benefit is for companies using a business model without time windows. These companies could increase its customer service level by incorporating the self-imposed time windows. The increase in operational costs can be weighted against the increase in the customer service level. A second possible benefit is for the companies who let decide their customers on the time window. These companies have a high customer service level but on the other hand also high operational cost. It could be beneficial for these companies to incorporate the VRP-SITW model in order to decrease these operational costs. The traveled distance and number of vehicles will decrease substantially and this can be weighted with the possible decrease in customer service level. For companies who already set the time windows by itself, this article could be beneficial as well. The influence of the penalty cost, the initial solution of the procedure and the different disruption scenarios can be investigated and could give some insight in the strategic decisions of the company.

Further research could make a setting where a subset of the customers has fixed or exogenous time windows. So the company lets decide the customer whether they want to set their time window by itself, against extra costs, or let the company decide on the time window. This gives a combination of the VRP-SITW and VRPTW and could give a reduction in operational costs and an increase in the customer service level. Furthermore, the research into time-dependent travel times may be interesting. Traveling in the early morning during rush hours will take more time than for example traveling during the night. Accounting for this phenomenon in the routing and scheduling decisions could be beneficial for the small-package carrier companies. Another extension can focus on adding stochasticity in the demand, like done in the Vehicle Routing Problem with Stochastic Demands. The demand of a certain customer is announced upon the arrival of the vehicle and this can lead to a return trip to the depot and therefore extra costs. All these possible further research topics may be relevant for companies, depending on its specific requirements and specifications.

References

Augerat, P., Belenguer, J., Benavent, E., Corber, A., and Naddef, D. (1998). Separating capacity constraints in the cvrp using tabu search. *European Journal of Operational Research*, 106:546–557.

Danielsson, P. (1980). Euclidean distance mapping. *Department of Electrical Engineering, Linko ping University*, 14(3):227–248.

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

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354.

Gendreau, M., Hertz, A., and Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290.

Gillet, B. and Miller, L. (1974). A heuristic algorithm for the vehicle dispatch problem. *Operational Research*, 22:340–349.

Homberger, J. and Gehring, H. (2005). A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162(1):220–238.

Jabali, O., Leus, R., Van Woensel, T., and de Kok, T. (2015). Self-imposed time windows in vehicle routing problems. *OR Spectrum*, 37(2):331–352.

Jabali, O., Van Woensel, T., de Kok, A. G., Lecluyse, C., and Peremans, H. (2009). Time-dependent vehicle routing subject to time delay perturbations. *IIE Transactions*, 41(12):1049–1066.

Kouvelis, P., Daniels, R., and Vairaktarakis, G. (2000). Robust scheduling of a two-machine flowshop with uncertain processing times. *IIE Transactions*, 32:421–432.

Laporte, G. (2009). Fifty years of vehicle routing. *Transport Science*, 43(4):408–416.

Laporte, G., Louveaux, F., and Mercure, H. (1992). The vehicle routing problem with stochastic travel times. *Transport Science*, 26(3):161–170.

Laporte, G., Louveaux, F. V., and Van Hamme, L. (2002). An integer l-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operational Research*, 50(3):415–423.

Potvin, J. Y. and Rousseau, J. M. (1995). An exchange heuristic for routing problems with time windows. *Operations Research Society*, 46(12):1433–1446.

Ralphs, T. (2010). Branch cut and price resource web. *http://www.branchandcut.org*, [Online; Accessed June 2018].

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

Solomon, M. (2010). Vrptw benchmark problems. *http://web.cba.neu.edu/ msolomon/problems.htm*, [Online; Accessed May 2018].

Statista (2017). Number of digital buyers worldwide from 2014 to 2021 (in billions). *https://www.statista.com/statistics/251666/number-of-digital-buyers-worldwide*, [Online; Accessed June 2018].

UNCTAD (2016). B2c e-commerce index 2016. *http://unctad.org/en/PublicationsLibrary/tn_unctad_ict4d07_en.pdf*, [Online; Accessed May 2018].

Yeo, K. and Ning, J. (2006). Managing uncertainty in major equipment procurement in engineering projects. *European Journal of Operational Research*, 171(1):123–134.

A Appendix

Algorithm 5: 2-opt* neighborhood

Input: The current solution Z and the distances between the customers
Output: The neighbors of Z using a 2-opt* exchange

```
function 2-opt* neigborhood
    initialize an empty set of neighbors B
    for all customers i ∈ V do
        for the η closest customers j to i do
            find the route of customer i and set this route to R1
            find the route of customer j and set this route to R2
            Set Z* = Z
            if R1 is not equal to R2 then
                Delete R1 and R2 from Z*
                find index of customer i in R1
                find index of customer j in R2
                Make a route S1 of R1 from index 0 to the index of customer i - 1
                Make a route S2 of R2 from index 0 to the index of customer j - 1
                Make a route S3 of R1 from index of customer i to the end of the route
                Make a route S4 of R2 from index of customer j to the end of the route
                Merge S1 and S4 and add the route to Z*
                Merge S2 and S3 and add the route to Z*
                Add Z* to B
    return the set B
```

Algorithm 6: Or-Opt neighborhood

Input: The current solution Z and the distances between the customers
Output: The neighbors of Z using a Or-Opt exchange

```
function Or-Opt neigborhood
    initialize an empty set of neighbors B
    for all customers i ∈ V do
        for the η closest customers j to i do
            find the route of customer i and set this route to R1
            find the route of customer j and set this route to R2
            Set Z* = Z
            for k = 1 to 3 do
                if k equals 1 and the exchange of nodes is feasible then
                    delete customer i from R1 and add him after customer j in R2
                    update R1 and R2 in Z*
                if k equals 2 and the exchange of nodes is feasible then
                    delete customer i and his successor from R1 and add them after customer j in R2
                    update R1 and R2 in Z*
                if k equals 3 and the exchange of nodes is feasible then
                    delete customer i and his two successors from R1 and add them after customer j in
                      R2
                    update R1 and R2 in Z*
                if Z is not equal to Z* then
                    add Z* to B and set Z* = Z
    return the set B
```

B Appendix

Table 6: Comparison of the three move selection criteria for the Solomon instances

Instance	Objective Value			CPU time (s)			Total
	C_1	C_2	C_3	C_1	C_2	C_3	
R101	**1094.6**	1167.5	**1094.6**	26.5	24.7	29.8	81.1
R102	**1094.1**	1167.4	**1094.1**	28.0	27.0	35.7	90.8
R103	**1093.0**	1166.5	**1093.0**	26.1	22.8	26.0	75.0
R104	**1091.8**	1165.6	**1091.8**	26.9	22.9	25.9	75.7
R105	**1091.0**	1165.1	**1091.0**	22.4	20.6	28.0	70.9
R106	**1091.0**	1165.1	**1091.0**	25.3	21.4	27.8	74.4
R107	**1091.0**	1165.1	**1091.0**	28.3	20.8	26.0	75.1
R108	**1091.0**	1165.1	**1091.0**	23.8	28.5	30.0	82.4
R109	**1091.0**	1165.1	**1091.0**	26.1	21.4	31.6	79.2
R110	**1091.0**	1165.1	**1091.0**	26.0	21.1	25.5	72.7
R111	**1091.0**	1165.1	**1091.0**	24.5	21.8	25.7	72.1
R112	**1091.0**	1165.1	**1091.0**	24.9	22.3	27.3	74.4
C101	**916.9**	980.3	**916.9**	36.7	24.5	34.9	96.1
C102	**916.9**	980.3	**916.9**	32.6	24.0	36.7	93.3
C103	**916.9**	980.3	**916.9**	31.4	22.8	38.9	93.1
C104	**916.9**	980.3	**916.9**	35.3	24.5	44.0	103.8
C105	**916.9**	980.3	**916.9**	42.3	26.5	41.1	109.8
C106	**916.9**	980.3	**916.9**	40.2	23.2	38.7	102.1
C107	**916.9**	980.3	**916.9**	34.4	22.5	43.1	100.1
C108	**916.9**	980.3	**916.9**	35.6	25.6	38.5	99.8
C109	**916.9**	980.3	**916.9**	37.4	26.8	35.0	99.2
RC101	**1091.1**	1226.3	**1091.1**	45.3	23.6	38.3	107.2
RC102	**1091.1**	1226.0	1128.2	41.4	23.9	39.5	104.8
RC103	**1091.1**	1225.6	1127.9	36.9	23.4	37.9	98.3
RC104	**1091.1**	1225.4	1127.7	37.6	24.0	39.6	101.2
RC105	**1091.5**	1225.7	1168.4	43.9	24.6	40.9	109.4
RC106	**1091.1**	1225.4	**1091.1**	36.1	26.4	42.2	104.6
RC107	**1091.1**	1225.4	**1091.1**	43.5	22.5	50.1	116.0
RC108	**1091.1**	1225.4	**1091.1**	45.4	24.2	45.1	114.6

This table represents the objective values and running times of the Solomon instances using the C_1, C_2 and C_3 criterion. The initial solution is obtained by the extended nearest neighbor heuristic.

Table 7: Comparison of VRP-SITW with optimal VRP solutions for the Augerat instances

Instance	Increase in distance			
	P5 (%)	P10 (%)	Prop (%)	1.3dist (%)
32k5	107.6	107.6	107.6	109.8
33k5	104.7	104.7	104.7	104.3
33k6	106.1	106.1	106.1	111.2
34k5	125.7	125.7	125.7	101.8
36k5	113.9	115.3	113.9	108.4
37k5	137.6	137.6	137.6	114.7
37k6	108.0	108.0	108.0	113.5
38k5	120.4	120.4	120.4	107.2
39k5	107.0	107.0	107.0	106.7
39k6	111.7	111.7	111.7	111.2
44k6	138.0	138.0	138.0	104.5
45k6	107.7	107.7	107.7	110.7
45k7	104.7	104.4	103.2	104.9
46k7	107.2	107.2	107.2	106.8
48k7	115.2	115.2	115.2	113.7
53k7	107.6	107.6	107.6	109.1
54k7	103.6	103.6	103.6	103.1
55k9	107.5	107.5	107.5	111.0
60k9	102.8	102.8	102.8	102.4
61k9	105.8	105.8	105.8	105.6
62k8	105.0	105.0	105.0	104.7
63k10	104.0	104.0	104.0	106.0
63k9	105.9	105.9	106.4	105.2
64k9	103.8	103.8	103.8	103.0
65k9	103.7	103.7	103.7	105.8
69k9	105.7	105.7	105.7	108.0
80k10	107.9	107.9	107.9	107.7
Average	109.2	109.3	109.2	107.1

This table indicates the increase in distance of the Augerat instances using the VRP-SITW compared to the VRP.