A tabu-search heuristic for the vehicle routing problem with self-imposed time windows

ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Bachelor Thesis: Econometrics and Operations Research

Author	Patrick Stokkink (425192)
Supervisor	Naut Bulten
Second Assessor	Twan Dollevoet

Abstract

In this thesis we analyze the vehicle routing problem with self-imposed time windows and the proposed solution procedure by Jabali et al. (2015). We analyze their results and discuss the differences between their results and the ones we found. In addition to this, we evaluate the effect of different initial solutions, showing that incorporation of shift length in the initial solution improves the results. Furthermore, we implement a random-sized tabu list, which proves to perform almost equally well as a tabu list with a fixed size. Lastly, we extend on the original problem by including time preferences and argue that this increases customer satisfaction.

Keywords: Vehicle routing problem, Vehicle routing problem with time windows, Travel time disruptions, Two phase solution method, Tabu search, Linear programming

July 4, 2018

Contents

1 Introduction

Many package carrier companies provide their customers with a time frame during which their package will be delivered. For example, PostNL indicates during which time window the purchase of the customer will be supplied. In such situations, the carrier itself decides in which time frame the package will be delivered, therefore making the time windows self-imposed. More specifically, the carrier assigns each customer to a route and, based on the ordering of it, decides on a time window. We refer to the problem of assigning customers to routes and deciding on time windows as the Vehicle Routing Problem with Self-Imposed Time Windows (VRP-SITW).

This thesis is based on the work of Jabali et al. (2015). They focus on the allocation of customers to routes based on the distance between them. In addition to this, time windows are communicated to the customer, which companies strive to respect as much as possible. In this thesis, we analyze the methods and results of Jabali et al. (2015). Furthermore, we address a couple of alterations to the heuristic they proposed. We consider new initialization methods for the heuristic, with the aim of improving the running time. In addition to this, we focus on a different implementation of both the neighborhood generation method and the tabu list.

Another contribution of this thesis is to include time preferences in the model. Some package carrier companies allow customers to indicate a preferred time window, during which they would like to be served. We assume the time window is not guaranteed but is taken into consideration during the scheduling phase. By including time preferences, the model provides a new option for companies to increase customer satisfaction. In our analysis, we take into account that the expectations of customers might change by giving them the option to indicate their preferences.

The remainder of this thesis is organized as follows. In Section 2, we outline the relevant literature concerning Vehicle Routing Problems (VRP) and tabu search. Section 3 provides the description of the problem that is dealt with in this thesis. The methodology and the computational results are discussed in Sections 4 and 5 respectively. Finally, Section 6 summarizes the results and addresses possible topics of further research.

2 Literature Review

Both the vehicle routing problem and the Vehicle Routing Problem with Time Windows (VRPTW) have often been a topic of research. The VRP-SITW, however, is a less studied problem. Jabali et al. (2015) give a clear description of the problem and describe an efficient solution method based on tabu search. In this section, we reflect on the relevant literature considering both VRP and VRPTW and the different methods that have been opted to solve these problems.

Laporte et al. (2000) discuss various classical and modern heuristics that have been used to solve vehicle routing problems. According to them, the quality of these modern heuristics is usually much higher, with the drawback of increased computation time. Similar to Jabali et al. (2015), they indicate that the computation times of tabu search heuristics are high. When the results are needed in a timely manner, for example when a last minute delivery needs to be scheduled, this is disadvantageous.

In this thesis, we implement a two-phase solution procedure, consisting of a routing and a scheduling phase. Mitrović-Minić and Laporte (2004) use a similar approach in the context of dynamic pickup and delivery with time windows, where they solve the routing and scheduling problem in a sequential manner. Similar to Jabali et al. (2015), a tabu-search procedure is implemented in the routing phase.

A noteworthy aspect is the stochastic environment that is dealt with in many extensions of the classical VRP. For example, Laporte et al. (1992) examined stochastic travel times where a penalty is imposed if the limit on route duration is exceeded. Jabali et al. (2009) used the idea that the travel time can be delayed by stochastic disruptions. Both of these ideas were used by Jabali et al. (2015), where tardiness and overtime penalties are imposed when the route duration is exceeded.

When time preferences of customers are considered, these are often treated as compulsory and exogenous. The compulsory nature of time preferences leads to the VRPTW. Efficient heuristics to solve this problem have been proposed, among others, by Cordeau et al. (2001) and Chiang and Russell (1997). Although the problem at hand is different from the above literature in the sense that time windows are set by the company, useful insights are provided in tabu search algorithms. For example, Chiang and Russell (1997) present a dynamically sized tabu list, based on the randomly varying list sizes reported by Taillard (1991). The idea behind this is that it is not necessary to decide on a constant size for the tabu list. Cordeau et al. (2001) discuss diversification mechanisms to explore a broad portion of the solution space. Such a mechanism has also been implemented by Potvin et al. (1996). They describe a procedure in which the search alternates between two neighborhood generation methods, thereby exploring new regions of the solution space.

3 Problem Description

The problem discussed in this thesis is to allocate customers to routes and to decide on service time windows for these customers. In this section, we discuss the most important features of the VRP-SITW.

In this problem, a set of N customers is considered as well as a homogeneous fleet of K vehicles. Each customer i has a demand of q_i and has to be assigned to a route, constrained by vehicle capacity Q. The solution of this problem, also referred to as Z, consists of a set of routes where each route R_r consists of n_r customers, both of which are determined during the routing phase. Each route starts and ends at the hub, which is defined as nodes 0 and n_{r+1} respectively. Therefore, the total number of nodes visited on the route is equal to $n_r + 2$. For each node $i \in R_r$, a departure time s_i is specified which is stored in a vector $\mathbf{s}$. The shift length is the time interval $[s_s, s_e]$ and each customer has a time window of length W_i within which the arrival of the vehicle is desired. Each node i has a service time of u_i, and the service times u_0 and u_{n_r+1} at the depot are set to zero.

The objective of the VRP-SITW, as described by Jabali et al. (2015), is twofold. The first goal, which is similar to the goal of the VRP, is to minimize the total travel time, which is captured by the first term of the objective function. The second part of the objective function consists of tardiness and overtime penalties. The idea is to assign a penalty if the time windows are not respected. Arrival before the scheduled window is not penalized, because the driver cost is assumed to be fixed, but arrival after the time window leads to a penalty proportional to the tardiness. Although early arrivals are not penalized, service can only start at the beginning of the indicated time window. In addition to this, if the shift duration is exceeded, overtime penalties have to be paid to the drivers. The objective function is therefore of the following form:

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

with c the cost of traveling one unit of distance, Z the set of routes R_r for which $|Z| \leq K$ (the maximum number of vehicles), d_{ij} the distance between customer i and j (this is later described as d_i, when j is the customer following i on the route) and $\Theta(R_r)$, which are the overtime and tardiness penalties of route R_r. The penalty function $\Theta(R_r)$ is described later in this section.

Jabali et al. (2015) assume that disruptions might occur during the execution of a route which increase the travel time. The length of this delay is equal to L_i and occurs with probability p_i on edge d_i. The variable L_i is distributed with probability mass function $g_i(\cdot)$. That is, all $l_{ik} \in \Psi_{ik}$ are positive values associated with a disruption for d_i for which $\sum_{k \in \Psi_{ik}} g_i(l_{ik}) = 1$. It is important to note that the actual value of L_i only becomes known when the arc is traversed, therefore no alterations to the schedule can be made at this point. Furthermore, we assume that only one disruption can occur, which is widely assumed for the VRP with stochastic demand and is also assumed by Jabali et al. (2015).

The uncertainty in travel times because of disruptions, such as traffic jams or roadblocks, is an important aspect of this problem. This uncertainty might affect departure times, which in turn might hinder the departure within the imposed time-window. We can define the actual departure time as follows:

$$s_i^a = max\{s_i; s_{i-1}^a + d_{i-1} + L_{i-1} + u_i\} \text{ for } i = 1, ..., n_r + 1 \text{ with } s_0^a = s_0. \tag{2}$$

The intuition behind this is that the delay should be incorporated in the actual departure time. Arrival before the indicated time window is not penalized. Late arrivals at customer i, however, are penalized with a tardiness penalty t_i per-unit-time delay. Similarly, late arrival at the depot is penalized with t_{n_r+1}. The driver is assumed to receive a fixed payment for a shift, considering it ends at time s_e. Arrival after the end of the shift is penalized with an overtime penalty b per time unit. Considering these penalties, the penalty term can be specified as follows:

$$\Theta(R_r) = \sum_{i \in R_r \setminus \{0\}} t_i \, \mathbb{E}[max\{0; s_i^a(\mathbf{s}) - (s_i - u_i + W_i)\}] + b \, \mathbb{E}[max\{0; s_{n_r+1}^a - s_e\}]. \tag{3}$$

The choice to use the expected performance instead of the worst-case performance is motivated by Jabali et al. (2015). They emphasize that in the context of the VRP-SITW good average customer service is desired, which does not entail hard constraints per customer.

The problem can be split into two stages. In the first stage, the customers have to be assigned to routes such that the capacity of the trucks appointed to these routes is not exceeded. In the second stage, the time windows are established considering shift length and disruptions, to which we refer as the scheduling problem. The procedure to find a good solution for this problem is discussed in the next section.

4 Methodology

In this section, we outline the most important methodology. The first subsection contains a description of the scheduling problem. Thereafter, we provide a hybrid solution procedure by implementing tabu search. In Section 4.3, we extend the model by including time preferences.

4.1 Scheduling and Determining Buffers

For a given route R_r we can find an optimal schedule, considering both disruptions and penalties, by using the linear program below. The variable Δ_{ijk} is the tardiness at client j due to a disruption of d_i according to scenario k. Likewise, Λ_{ik} is the overtime resulting from a disruption at customer i by scenario k. Because of the modeled disruptions, two situations can be distinguished for each route, that is, either no leg is disturbed or a single leg is disturbed. The probability that the leg between customer i and its descendant is disturbed is defined as p_i. If no leg is disturbed, ζ is the overtime for the route. In this case, tardiness penalties are not relevant.

$$\Theta(R_r) = min \sum_{i=0}^{n_r} \sum_{j=i+1}^{n_r+1} \sum_{k=1}^{|\Psi_i|} p_i g_{ik} t_j \Delta_{ijk} + b \sum_{i=0}^{n_r} \sum_{k=1}^{|\Psi_i|} p_i g_{ik} \Lambda_{ik} + b(1 - \sum_{i=0}^{n_r} p_i)\zeta \tag{4}$$

subject to

$$s_{i-1} + d_{i-1} + u_i \leq s_i \qquad i \in R_r \setminus \{0\} \tag{5}$$

$$s_0 \geq s_s \tag{6}$$

$$s_i + d_i + l_{ik} + \sum_{m=i+1}^{j-1} (u_m + d_m) \leq s_j - u_j + W_j + \Delta_{ijk} \tag{7}$$

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

$$s_{n_r+1} + \Delta_{i,n_r+1,k} - s_e \leq \Lambda_{ik} \qquad i \in R_r \setminus \{n_r + 1\};\ k \in \Psi_i \tag{8}$$

$$\zeta \geq s_{n_r+1} - s_e \tag{9}$$

$$\Delta_{ijk} \geq 0;\ s_i \geq 0;\ \Lambda_{ik} \geq 0; \zeta \geq 0 \tag{10}$$

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

The objective function (4) minimizes the penalties for tardiness and overtime for a given route. The first term is based on the tardiness penalty at the customer and the cost for arriving late at the depot. The second term provides the overtime penalty in case a leg is disturbed. The last term in this function gives the overtime penalty for later arrival at the depot in case no leg is disturbed. Restrictions (5) can be viewed as the precedence constraints. The scheduled departure from customer i should be at least the departure time of the preceding customer, plus the service time of customer i and the distance between the two customers. The buffer between customers $i-1$ and i is therefore defined as $s_i - (s_{i-1} + d_{i-1} + u_i)$. Constraint (6) ensures that the scheduled departure time from the depot cannot be earlier than the starting time of the shift. Constraints (7), (8) and (9) determine each of the delay terms described above.

4.2 Solution procedure

The VRP-SITW is NP-hard. It is therefore wise to use a heuristic when solving this problem. For solving the VRP and the VRPTW, the tabu search heuristic has been widely used, for example by Potvin et al. (1996). Jabali et al. (2015) also adopt a tabu search algorithm to solve the VRP-SITW. The algorithm consists of two stages. In the first stage, neighbors of the current route are generated. The best neighbor is selected according to the criteria that are discussed in Section 4.2.3. Thereafter, in the second stage, the scheduling of the customers is done according to the LP problem discussed in Section 4.1. Because of time limitations, Jabali et al. (2015) proposed three different criteria for selecting the best move in the neighborhood, to avoid solving the LP problem for every neighbor in every iteration.

The tabu search algorithm is described in Algorithm 1. In Section 4.2.1 we propose four different methods to construct the initial solution in line 1. The neighborhood generation in line 4 is discussed in Section 4.2.2. The three aforementioned move selection criteria are discussed in Section 4.2.3 and are based on distance (C1), combined with marginal penalties (C2) and buffer size (C3).

Algorithm 1: Tabu Search for VRP-SITW

Input: Instance containing coordinates, demand and service time of each customer and the depot
Output: The solution to the VRP-SITW with the lowest objective among all generated solutions

1 Construct initial solution Z_0 and compute $F(Z_0)$
2 **for** ξ = *1 to 3* **do**
3 Set $Z \leftarrow Z_0$ and $F(Z) \leftarrow F(Z_0)$
4 Generate the neighborhood of Z
5 Evaluate all neighbors on criterion C_ξ and retain the best non-tabu solution as new solution Z
6 Evaluate $F(Z)$ and update the tabu list to include Z
7 **if** *Z is feasible and is better than the current best solution* **then**
8 Update the best feasible solution for C_ξ to Z
9 Update excess demand penalty
10 **if** *No improvement in* η_{max} *iterations* **then**
11 Store best solution for C_ξ
12 **else**
13 Go to Step 4
14 **return** *The best solution from* ξ *= 1, 2 and 3*

4.2.1 Initial Solution

Jabali et al. (2015) use the nearest neighbor heuristic to obtain their initial solution. In this heuristic, the nearest neighbor of the current customer is iteratively added to a route. When adding the nearest neighbor exceeds the capacity limit on a route, a new route is created. The capacity limit is imposed by the capacity of the vehicles. This process is repeated until all customers are covered. The advantage of using this method is that it is fast and simple to implement. However, the drawback of this method is that the quality can be rather low. Besides this, it does not consider the maximum shift duration, which might lead to high overtime penalties in the initial solution.

Another classical method is the savings heuristic by Clarke and Wright (1964). The idea behind the savings heuristic is that all customers start on separate routes. Then the savings s_{ij}, obtained by merging the two routes with customers i and j, are computed. The biggest saving is iteratively selected and the two routes are merged by positioning all customers of one route after the customers of the other, if this leads to a feasible solution.

A third classical method is the sweep heuristic, which is often attributed to Gillett and Miller (1974). According to Laporte et al. (2000), both the savings heuristic and the sweep heuristic lead to good solutions quickly. In contrast to the other methods, a customer is represented by its polar coordinates (θ_i, ρ_i) where θ_i is the angle and ρ_i is the ray length. An arbitrary customer i^* is chosen for which θ_{i^*} is set to 0. All other angles are calculated centered at the depot, from the initial ray $(0, i^*)$. The customers are then sorted in non-decreasing order of their θ_i. The first step of the sweep heuristic is to form clusters. The first customer on this list is added to a cluster iteratively as long as the capacity is not exceeded, otherwise a new cluster is created. The task of creating a route within a cluster is done according to the Traveling Salesman Problem (TSP). We find a solution to the TSP heuristically with the nearest neighbor heuristic, because the TSP is NP-hard and is therefore time-consuming to solve exactly. The division of clusters can be highly influenced by the choice of i^*. For this reason, we perform the sweep heuristic for every i^* and select the set of routes that minimizes the total distance traveled.

Lastly, we adjust the nearest neighbor heuristic to include the maximum shift duration. This means that we include an extra restriction that imposes that the total expected time of a route (service + travel time) cannot exceed the shift length, which reduces the imposed overtime penalty on the initial solution. An important note is that for some instances it is impossible to add a certain customer without exceeding the shift length. For example, when this customer lives far away. In these cases, the customer should be added to a single route for which the time restriction can be ignored, because otherwise this procedure does not lead to a solution. This procedure generally leads to more routes than the original nearest neighbor heuristic. However, since Jabali et al. (2015) did not specify a maximum value for K, we assume that this is allowed.

The nearest neighbor heuristic is implemented to obtain the results in Section 5. The other initial solutions discussed in this section are used to obtain a comparison of the initial solutions in Section 5.5.

4.2.2 Neighborhood Generation

The algorithm implements two types of neighborhood generation methods. Both methods create a neighborhood for every customer i with the η closest customers, as is described by Potvin et al. (1996). The first method is the 2-opt* neighborhood by Potvin and Rousseau (1995). This exchange method is especially effective for problems with time-windows because it preserves the direction of the routes. Hence, according to Potvin and Rousseau (1995), this method is likely to generate a feasible solution. The second algorithm is the Or-opt neighborhood by Or (1976). The Or-opt algorithm moves small sequences of adjacent customers (one, two or three) that

are generally close together and inserts these customers at a new position while preserving the orientation. According to Potvin and Rousseau (1995), the Or-opt heuristic can generate solutions that are close to the quality of 3-opt for problems with time-windows but with much less computation time.

In the 2-opt* neighborhood generation, two links are replaced by two different links. More specifically, the links $(i, i+1)$ and $(j, j+1)$ are replaced by the links $(i, j+1)$ and $(j, i+1)$, where $j+1$ is one of the η nearest neighbors of i. One drawback of this method is that it can only be applied when i and j are not on the same route. Because we select the η nodes closest to i in distance, it is very likely that those are on the same route, therefore eliminating many possible neighbors. In the Or-opt neighborhood generation, customer i (possibly with customers $i+1$ and $i+2$) is moved after customer j if customer i is one of the η nearest neighbors of customer j.

Potvin et al. (1996) describe a refinement to the tabu search heuristic known as diversification. This means that the search alternates between Or-opt and 2-opt* neighborhoods. When $\lceil 0.5\eta_{max} \rceil$ iterations have been performed without an improvement to the best known solution with one of the methods, the other method is used. The general iteration limit of η_{max} still holds in this case. The advantage of this method is that it forces the heuristic to explore new regions of the search space.

4.2.3 Selection Criteria

Jabali et al. (2015) use the three aforementioned criteria to select the best move in a neighborhood. Each criterion tackles a different aspect of the problem. All of the criteria use the travel costs associated with solution Z, which are defined as,

$$\Omega(Z) = c \sum_{R_r \in Z} \sum_{(i,j) \in R_r} d_{ij}. \tag{11}$$

However, this generally allows solutions where the total demand exceeds the capacity, because there is no hard capacity constraint in the selection criteria. Therefore, we include a penalty if the capacity is violated, which is done in the following way:

$$\Omega_2(Z) = \Omega(Z) + \omega \sum_{R_r \in Z} \left[(\sum_{i \in R_r} q_i) - Q \right]^+, \tag{12}$$

where q_i is the demand of each node on the route and Q is the capacity of the vehicle. The excess penalty ω is decreased by multiplication with a factor υ after ϕ successive feasible iterations. Likewise, ω is increased (multiplied by factor υ^{-1}) after ϕ infeasible moves.

The first criterion is purely based on minimizing the distance and does not take the time windows into account. Let Z' be a neighbor of current solution Z, where we define $\Delta_1(Z') = \Omega_2(Z) - \Omega_2(Z')$. According to criterion C_1, the chosen move is not tabu and maximizes $\Delta_1(\cdot)$. Alternatively, it might be convenient to incorporate a penalty component for the assessment of the time window. By choosing the move that is not tabu and maximizes $\Delta_2(\cdot)$, we use the

observation that penalties increase with the number of customers on the route. On the contrary, decreasing the number of customers on a route with a large penalty is likely to decrease the total objective value of the route, because the probability of overcapacity decreases. In this equation, moves considering routes R_1 and R_2 lead to the new solution Z'. In addition to this, n_r and n'_r are the number of nodes on route r in the current and new solution respectively.

$$\Delta_2(Z') = \Omega_2(Z) - \Omega_2(Z') + \rho[\Theta(R_1) + \Theta(R_2) - \frac{\Theta(R_1)}{n_1+1}(n'_1+1) - \frac{\Theta(R_2)}{n_2+1}(n'_2+1)] \tag{13}$$

Criterion C_3 favors moves with a small buffer size. Buffer size $bu(i)$ between customers i and $i+1$ is defined as $s_{i+1} - (s_i + d_i + u_{i+1})$. In this equation, Z' involves a move between customer i and j. The selected move is the one that is not tabu and maximizes $\Delta_3(\cdot)$. The logic behind this is that improvements in travel times are more likely to decrease the penalties when the buffers are small.

$$\Delta_3(Z') = \Omega_2(Z) - \Omega_2(Z') - \gamma[bu(i) + bu(j)] \tag{14}$$

The evaluation of $\Theta(\cdot)$ and $bu(\cdot)$ is based on the current solution rather than the neighbor to avoid solving the LP problem for every neighbor, which is time-consuming.

4.2.4 Tabu list

To avoid cycling, Jabali et al. (2015) use a tabu list. Their approach is to add solutions (Z) to the tabu list. In addition to this implementation, we add a random element to this list size, as described by Taillard (1991). Rather than choosing a fixed list size of κ, we choose κ between κ_{min} and κ_{max}. The value for κ is randomly drawn from a uniform distribution on the interval $[\kappa_{min}, \kappa_{max}]$ and rounded upwards to the nearest integer every $2\kappa_{max}$ iterations. In this way, the problem related to finding the optimal tabu list size is avoided.

The basis for our algorithm is to use the same implementation as Jabali et al. (2015), therefore implementing a tabu list with a fixed size. However, in Section 5.7 we focus on the computational differences when a tabu list has a random size.

4.3 VRP-SITW with time preferences

Customers can often indicate during what time window they wish to be served. A first possibility is treating these time windows as compulsory, making the problem a VRPTW. Another option is taking the time preferences as a suggestion, which is referred to as the Vehicle Routing Problem with Self-Imposed Time Windows and Time Preferences (VRP-SITW-TP). In this way, a penalty is imposed when the time window is not satisfied. The intuition behind this is that customers can indicate during which part of the day they wish to be served, for example in the morning or during the evening.

4.3.1 Description of the VRP-SITW-TP

The mathematical model for the VRP-SITW-TP builds forth on the VRP-SITW discussed in Section 4.1. An extra penalty term is imposed when the indicated service window lies outside the preferred time window. The time gap is two-sided, that is, both early and late service should be penalized. The time gap between the indicated service window and the preferred time window for customer i is referred to as π_i and can be defined as:

$$\pi_i = max\{0,\ s_i - e_i,\ a_i - (s_i - u_i)\}, \tag{15}$$

where a_i and e_i are the start and end time of the preferred time window respectively. The intuition behind this equation is provided in Figure 1.

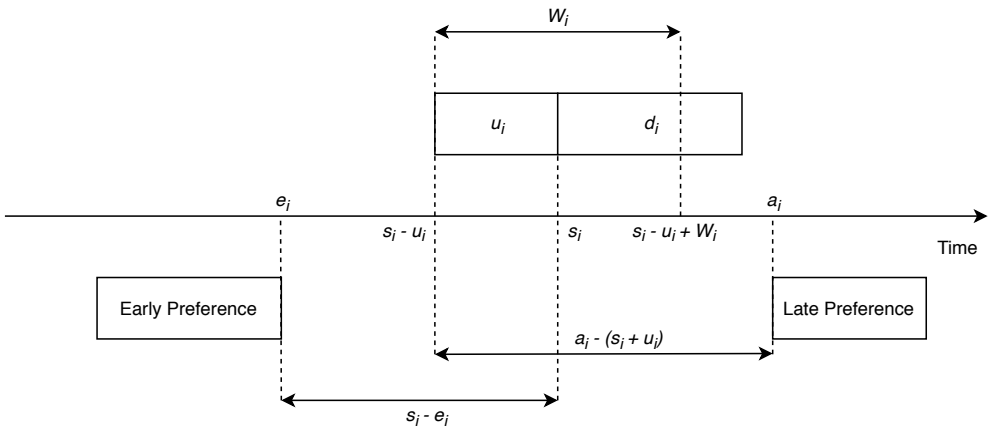

Figure 1: Illustration of a time window and time preference at customer i

For this equation to hold, three assumptions have to be made. First, we assume that the time gap is based on the indicated time window rather than the real time window. The intuition behind this is that customers know their indicated departure time beforehand, therefore having the opportunity to reschedule other appointments. In addition to this, it will foster computational simplicity, because the actual departure times are unknown during the scheduling phase. Second, the time gap only applies to customers, since no time frame has to be communicated to the depot. Third, the preferred time window should be at least as long as the service time. The reason for this is that there always needs to be a possibility to satisfy the preferences.

We can now model the restrictions for the time gap in the same way as the delay terms in Section 4.1. The new model contains all the restrictions from the previous model, with an additional penalty term in the objective function:

$$\Theta(R_r) = min \sum_{i=0}^{R_r} \sum_{j=i+1}^{R_r+1} \sum_{k=1}^{|\Psi_i|} p_i g_{ik} t_j \Delta_{ijk} + b \sum_{i=0}^{n_r} \sum_{k=1}^{|\Psi_i|} p_i g_{ik} \Lambda_{ik} + b(1 - \sum_{i=0}^{n_r} p_i)\zeta + v \sum_{i=1}^{n_r} \pi_i \tag{16}$$

where v is the penalty for each unit of time the assigned window differs from the preferred window. In addition to this, new restrictions have to be introduced based on Equation (15),

which implicitly determine the time gap:

$$\pi_i \geq s_i - e_i \qquad \forall i \in R_r \setminus \{0, n_r + 1\}, \tag{17}$$

$$\pi_i \geq a_i - (s_i - u_i) \qquad \forall i \in R_r \setminus \{0, n_r + 1\}, \tag{18}$$

$$\pi_i \geq 0 \qquad \forall i \in R_r \setminus \{0, n_r + 1\}. \tag{19}$$

The tabu search heuristic can also be applied to this model. Since the problem differs slightly from the original problem, some alterations to the heuristic are advantageous. Because of the introduced time preferences, it might be beneficial to use more vehicles than the amount used before. Therefore, we introduce an additional neighborhood generation method that generates extra routes, which is discussed in Section 4.3.2. In addition to this, we consider the time preferences in our neighborhood moves. For this reason, we introduce a new selection criterion in Section 4.3.3.

4.3.2 Extra Routes

Our procedure to create extra routes is similar to the Or-opt moves discussed in Section 4.2.2. For every customer i, we generate Or-opt neighborhoods where customer i (and possibly $i + 1$ and $i + 2$) is moved from its current route to a newly created one. The customers are moved directly after the depot, suggesting that $j = 0$. These new neighbors are added to the existing neighborhood of Or-opt and 2-opt* moves. The selection of the best neighbor is performed with the new selection criterion, which is discussed in the next section.

4.3.3 Selection Criterion

Jabali et al. (2015) introduced various criteria to select the best move in a neighborhood. However, none of these criteria incorporates the effect of time preferences. For this reason, we introduce a new criterion C_{TP} that favors moves of customers with a large time gap. That is, criterion C_{TP} favors to interchange customer i with a large time gap, with customer j whose position would be favorable for customer i. We define $\pi_{i,j}$ as the time gap of customer i if it is placed after customer j. The value for $\pi_{i,j}$ is calculated as follows:

$$s_j^* = s_j + d_{ij} + u_j, \tag{20}$$

$$\pi_{i,j} = max\{0,\ s_j^* - e_i,\ a_i - (s_j^* - u_i)\}. \tag{21}$$

In addition to this, $\pi_{i,j}$ is equal to zero if i is moved to a new route. The intuition behind this is that moving a customer to a new route provides more freedom for selecting a departure time. The selected move is not tabu and maximizes

$$\Delta_{TP}(Z') = \Omega_2(Z) - \Omega_2(Z') + \delta(\pi_i - \pi_{i,j}), \tag{22}$$

where Z' involves a move between customers i and j. The logic behind this is that moving a customer with an unsatisfied preference to a position where the preference has a higher satisfaction is likely to decrease the total objective value.

5 Computational Results

In this section, we discuss the computational results we obtained from various experiments with the implemented heuristic. All experiments are performed on an Intel Core I7 with 2.8 GHz and 12 GB DDR4 memory. The implementation is coded in Java in single thread. Java CPLEX is used for solving the LP instances. We have adopted two datasets from the literature, which are also used by Jabali et al. (2015). The first dataset contains VRP instances from Augerat et al. (1998), in which the number of customers ranges from 31 to 79. All customers are assumed to have a service time of 10 time units and a time window of 60 time units. The start and end time of a shift are 0 and 200 respectively. In this dataset, the vehicle capacity is 100 units. The second dataset contains VRPTW instances from Solomon (1987), which can be found in Solomon (2010). All sets contain 100 customers, with service time and time windows that are provided by the instances. The start and end times of the shifts can be found in the opening and closing time of the depot. In this dataset, the vehicle capacity is 200 units.

The algorithm and formulation discussed in Section 4 require a large number of parameters. To verify the results, we use the same parameters as used by Jabali et al. (2015). One exception is the value for ρ, associated with criterion C_2, which we chose equal to 0.1 instead of 1. The results obtained by using $\rho = 0.1$ are more similar to those reported by Jabali et al. (2015), in the sense that when using $\rho = 1$ almost no improvements are found by the algorithm. This suggests that the value for ρ has been misspecified.

The value for the parameter η_{max} has not been specified by Jabali et al. (2015). In general, instances with more customers require a higher value for η_{max}. For simplicity, we use the same value for η_{max} for all instances. Two main aspects that need consideration in deciding on a suitable value are that low values might terminate the algorithm too soon, and, moreover, high values for η_{max} increase the computation time without improving the solution. This makes the computation time more difficult to interpret, because a part of the time is unnecessarily spent on waiting for the iteration limit to be reached. This can be deducted from the search pattern entering a cycle, suggesting that no update will be found in the remaining iterations. In addition to this, high computation times are undesirable in practice. After careful consideration, we specify the value of η_{max} to be 200. Experiments with different values show that the algorithm would have performed more updates after termination for values below 200. On the contrary, values higher than 200 do not improve the objective.

In Section 5.1, we compare the different move selection criteria based on their objective value as well as computation time. Thereafter, we compare the results for different tardiness penalties in Section 5.2, followed by a comparison of the VRP-SITW with the VRP and VRPTW in Sections 5.3 and 5.4 respectively. Then we discuss the effects of different implementations of the initial solution (5.5), neighborhood generation (5.6) and tabu list (5.7). Finally, we analyze the influence of incorporating time preferences in our model in Section 5.8.

5.1 Move Selection

In this section, we analyze the performance of the three different evaluation criteria discussed in Section 4.2.3. We base the performance on both the objective value and the computation time of the algorithm with each of the criteria implemented. Table 9 and 10 in the Appendix contain the results for the Augerat and Solomon instances respectively and these results have been summarized in Table 1.

For both sets of instances, the C_2 criterion produces the best objective values the most often. The computation time with the C_2 criterion is also the highest for the Solomon instances. For the Augerat instances, however, the computation time with the C_3 criterion is the highest. The difference in computation times between the two sets of instances can be explained by the fact that the Solomon instances consist of more customers, thus creating larger neighborhoods. In addition to this, the routes consist of more customers, therefore the LP problem consumes more time. Some of the Solomon instances produce similar results as these are identical to each other, aside from the time windows that are relaxed. It should be noted that for many instances criterion C_1 and C_3 produce similar results. The reason for this is that the time buffers are often equal to zero. Therefore, it is likely that the two criteria select similar moves.

Table 1: Comparison of the three neighbor selection criteria

Instance	Frequency lowest objective			Average CPU time (milliseconds)		
	C_1	C_2	C_3	C_1	C_2	C_3
Augerat	13	17	13	3,754.8	5,739.1	6,566.5
Solomon	17	29	16	17,702.1	31,632.3	18,771.6

An important note is the differences between the obtained results and the results provided by Jabali et al. (2015). Considering the Solomon instances, Jabali et al. (2015) have higher objective values for the C1 instances, but lower for the R1 and RC1 instances. An even more stunning observation is the difference between the computation times. The computation times reported by Jabali et al. (2015) are up to one hundred times larger than the computation times we found. There are various explanations for these differences. A first explanation for the differences in computation times is the different computer that has been used, as well as different software. Another explanation is the unspecified value for η_{max}. It is possible that Jabali et al. (2015) used a higher value for η_{max} which consequently led to higher computation times. A last explanation is that the implementation of the neighborhood generation is different. Jabali et al. (2015) describe their neighborhood generation method as follows: *"For each customer $i \in V$, we construct 2-opt* (Potvin and Rousseau, 1995) and Or-opt (Or, 1976) neighborhoods for the η nodes closest to i"*. This leaves much room for interpretation. Therefore, it is likely that the neighborhood generation method used by Jabali et al. (2015) differs from the one we used, which is explained in Section 4.2.2. The different implementation produces different results regarding objective values and computation time. We elaborate on this reason in Section 5.6.

An interesting observation is that the running time with the C_3 criterion is the highest for the Augerat instances, but not for the Solomon instances. The C_3 criterion requires additional computation time for the computation of the time buffers in every iteration. For the Solomon instances, this is accompanied with an increase in efficiency of the search pattern, therefore reducing overall computation time. For the Augerat instances, the buffers are relatively small and do not influence the search pattern significantly, therefore increasing the overall computation time.

Other than Jabali et al. (2015), we recommend running all three criteria for all instances. This difference is mainly constituted by the reduced running time. The running time of the larger Solomon instances is beneath two minutes which is substantially lower than the running times reported by Jabali et al. (2015).

5.2 Tardiness Penalty

Jabali et al. (2015) also evaluated the effect of varying delay penalty costs t_i. In this section, we analyze their results. Jabali et al. (2015) proposed four different cost settings. The first two cost settings, referred to as P5 and P10, are constant with respectively $t_i = 5$ and $t_i = 10$. The third setting, referred to as Prop, deals with costs that are proportional to the quantity ordered ($t_i = q_i$). The final setting, 1.3dist, increases all distances by 30% and has delay penalty costs $t_i = 5$. According to Jabali et al. (2015), this reduces the available slack time, leading to less buffer time, thus resulting in tighter instances.

The results of this experiment are in Table 11 in the Appendix and have been summarized in Table 2. The table contains the average increase in objective value for each of the settings compared to the P5 setting, as well as the average penalty ratio for all settings. This ratio is defined as the proportion of the total objective that corresponds to penalties,

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

The results indicate that the objective values of the P10 and Prop settings are on average only 0.1% higher than those of the P5 setting. This illustrates that even when the penalties are doubled, this has almost negligible effect on the objective value. On the other hand, the 1.3dist setting has a large influence. The distances are increased by 30%, which suggests that the objective would also increase with 30%. However, because the instances are tighter, there is less buffer time which in turn leads to higher overtime and delay penalties.

Table 2: Comparison of four different tardiness penalty settings

	P5	P10	Prop	1.3dist
Average penalty ratio (%)	33.4	33.4	33.4	43.9
Increase in objective compared to P5 (%)	*	0.1	0.1	54.1

Considering the penalty ratio, the P5 setting corresponds to the lowest ratio. However, the P10 and Prop settings do not increase the penalty ratio significantly. The 1.3dist setting, on the other hand, does increase the penalty ratio with a substantial percentage. From this, we conclude, similar to Jabali et al. (2015), that an increase in distance has a considerable impact on the delay and overtime penalties.

Comparing the average penalty ratios in Table 2 to those reported by Jabali et al. (2015), a significant difference can be witnessed. The same reasoning can be applied as in Section 5.1, concerning the neighborhood generation method, on which we elaborate in Section 5.6.

5.3 Comparison of VRP-SITW to VRP

In this section, we analyze to what extent the addition of the self-imposed time windows to the VRP affects the distance traveled. We compare the traveled distance in the Augerat instances using the aforementioned delay penalty costs, with the optimal VRP solutions taken from Ralphs (2010). For the 1.3dist setting, the distances of the VRP solution have been scaled by a factor of 1.3. The details are provided in Table 12 in the Appendix and have been summarized in Table 3. The average distance increase is only 4% for all settings. Therefore, we conclude that the time windows do not have a substantial influence on the distance traveled. Companies that do not incorporate self-imposed time windows can provide a more customer-oriented service that is associated with a relatively small increase in the distance traveled.

Table 3: Average distance of VRP-SITW compared to VRP using the Augerat instances

P5 (%)	P10 (%)	Prop (%)	1.3dist (%)
104.2	104.2	104.2	104.1

5.4 Comparison of VRP-SITW to VRPTW

In this section, we compare our results of the VRP-SITW to the best known solution of the VRPTW. The aim is to evaluate the difference between the flexible, self-imposed time windows and the exogenous predetermined time windows. In our evaluation, we use the same instances as used by Jabali et al. (2015), which are the VRPTW instances from Solomon (1987). The best-known solutions are retrieved from Solomon (2010). The results are reported in Table 13 in the Appendix and are summarized in Table 4. Abbreviations have been used for the distance and the number of vehicles. T_F and T_S denote the distance for the VRPTW and the VRP-SITW respectively. Similarly, K_F and K_S denote the number of vehicles used.

The second column of the table indicates that the distance traveled increases on average when the time windows are exogenous. A difference can be observed between the R1 and RC1 instances on the one hand and C1 instances on the other. The limited increase in distance of C1 instances suggests that the time windows for these instances are quite unrestrictive. Investigation of the instances shows that the shift length of the C1 instances is significantly larger compared to the R1 and RC1 instance (1236 compared to 230 and 240 respectively), while the

time windows are still evenly distributed across the duration of the shift. This implies that the limited increase in distance is logical. Similar conclusions follow from the low penalty ratio as well as the absent vehicle reduction between VRPTW and VRP-SITW for the C1 instances.

Table 4: Comparison of VRP-SITW with the best-known VRPTW solutions for the Solomon instances

Instance	T_S/T_F (%)	$K_F - K_S$	Penalty ratio (%)
R1	74.9	5.0	24.6
C1	99.9	0.0	0.0
RC1	76.7	3.8	11.2
Average	83.1	3.1	13.3

Reflecting on all instances, we can conclude that restrictive instances (instances with a small shift length), allow for substantial reductions of travel times and number of used vehicles. Therefore, we advise companies that deal with small shift lengths to use self-imposed time windows rather than exogenous time windows. In addition to this, companies that have long-lasting shifts can use exogenous time windows without increasing the total distance traveled and the number of vehicles used drastically and thereby providing a more customer-oriented service.

5.5 Initial Solution

In this section, we compare the different initial solutions we proposed in Section 4.2.1. We consider both the objective value as well as the computation time. The results of this experiment are provided in Table 14 in the Appendix and have been summarized in Table 5. The results are generated by using the C_1 criterion for the Solomon instances. Similar results can be obtained for the other two criteria and for the Augerat instances.

Table 5: Comparison of the initial solutions for the Solomon instances using the C_1 criterion

Heuristic	Average objective value	Average CPU time (milliseconds)
Nearest neighbor	1,130.1	17,702
Nearest neighbor + TC [1]	1,017.8	10,556
Sweep	1,058.5	12,473
Saving	1,152.4	19,843

[1] TC = Time Constraints

Jabali et al. (2015) implemented the nearest neighbor heuristic. The results indicate that this procedure leads to a relatively long average computation time. Considering the objective value, this method produces a relatively high objective value compared to the other initial solutions. The nearest neighbor heuristic with time constraints performs the best of the four initial solutions considering both the achieved objective value and the computation time. However, it should be noted that this method also uses more vehicles for some instances compared to the other methods. Based on these findings, our advice is to use the nearest neighbor heuristic with

time constraints when there is no restriction on the number of vehicles. Otherwise, our advice would be to use the sweep heuristic. This heuristic produces slightly worse results, however, the number of vehicles used is lower.

5.6 Neighborhood Generation

The previous sections indicated large differences between our obtained results and the reported results by Jabali et al. (2015). Until now, these differences have been attributed to a different implementation of the neighborhood generation method. In this section, we implement a different method for neighborhood generation to verify whether this effect is indeed substantial. We implement the neighborhood generation method with diversification, as was discussed in Section 4.2.2. The effect of the different implementation on the results using the C_1 criterion in the Augerat instances is shown in Table 15 in the Appendix.

The objective values achieved by the two different implementations are different. The average difference is almost 3% and these differences can reach up to 11% in some instances when Or-opt is used in the first iterations. When 2-opt* is used in the first iterations, the average difference is more than 7% and these differences reach up to 20% in some instances. This indicates that, even though the difference in implementation is minimal, the effect on the objective value is substantial. For this reason, we believe that the difference in objective value between our research and that of Jabali et al. (2015) can possibly be justified by a different implementation of the neighborhood search heuristic.

In Section 4.2.2, we stated that many possible 2-opt* neighbors are eliminated because they are on the same route. By selecting the η nearest customers that are not on the same route for the 2-opt* method, we can evaluate the effect on the objective value. Experiments indicated that no or insignificant improvements were found by this alteration, thus suggesting that the elimination of neighbors because they are on the same route is acceptable.

Although these results might seem contradictory, a clear conclusion can be drawn. Our results suggest that the order in which neighbors are visited is of significant importance for the obtained objective value. However, the number of nearest neighbors evaluated in each iteration does not have a significant influence.

5.7 Tabu List

In this section we investigate the influence of using a random-sized tabu list on the results of the first criterion considering the Augerat instances. Jabali et al. (2015) used a tabu list of size κ, which was equal to 20. For this reason, we choose the interval $[\kappa_{min}, \kappa_{max}]$ in such a way that 20 is contained in the interval. We compare the results of four cases. In the first case, κ is fixed at 20. For the other three cases, the interval ranges from 10 to 30, 40 and 50 respectively. It should be noted that the value for κ is generated from a uniform distribution, rounded to the nearest integer. Each run therefore contains a random element, suggesting that the results can slightly differ between runs. Experiments with multiple runs have shown that these differences are not significant. The results of this experiment are provided in Table 16 in the Appendix and are summarized in Table 6. From these results we can conclude that the implementation of a random-sized tabu list does not have a significant influence on the objective value, nor on the computation time. This suggests that, when no specific estimation of the optimal tabu list size can be made, random size within a well-specified interval produces satisfying results.

Table 6: Average objective value and computation time for the different values of κ

κ	20	[10,30]	[10,40]	[10,50]
Objective value	1,688.5	1,692.2	1,689.7	1,692.5
CPU time (milliseconds)	3,755	3,730	3,679	3,854

5.8 Analysis of the VRP-SITW-TP

In this section we analyze the influence of time preferences, using the Solomon instances. We chose to use three equal sized time preference possibilities. Intuitively, this corresponds to the morning, afternoon and evening. We also allow selecting multiple time frames, thereby enlarging the preferred window. We generated the time preferences by using the exogenous time windows provided by the Solomon instances. The preferred time window covers the start and end time provided by the instance, such that multiple time windows are selected if those lie in different windows.

For the evaluation of the time preference satisfaction, we consider two evaluation criteria. The first is the percentage of the customers whose preference has been satisfied. That is, where the indicated service window falls within the preference of the customer. In addition to this, we consider the average deviation from the preferred time window, as a percentage of the shift length.

We consider four different penalty settings, of which the values for v and δ are shown in Table 7. The TP0 case aims to find the current situation. By setting δ to 0, the C_1 criterion is used and therefore time preferences are not considered during the route selection procedure. In addition to this, by choosing v extremely small, this is unlikely to affect the outcome of the scheduling problem and therefore provides insights into the situation where time preferences are not considered. The TP1 setting allows us to evaluate the solution when time preferences are

considered during the scheduling problem, but not during the routing procedure. The other penalty settings allow us to analyze the effect of higher penalties on the solution, when time preferences are considered in both stages of the problem. The same parameter values are used as before, except for the value of η_{max} which we increase to 300. The reason for this is that a larger neighborhood is generated, creating more possible moves. Various experiments have shown that choosing η_{max} to be 300 leads to good solutions regarding both the objective value and computation time.

Table 7: Penalty settings for the time preference experiment

	TP0	TP1	TP2	TP3
v	0.0000001	0.1	0.1	0.1
δ	0.0	0.0	0.1	1.0

The computation time of this problem is higher compared to the original VRP-SITW. The average computation time using the TP2 setting is 33 seconds, which is higher than the computation time of each of the three criteria, which are provided in Section 5.1. Multiple reasons for this can be found. For example, the scheduling problem is extended with an extra variable and two extra restrictions, which increases the time needed to solve the scheduling problem. In addition to this, an extra neighborhood is generated compared to the original solution procedure, which takes time to generate and evaluate.

The results of the experiment are displayed in Table 8. From the first row we can deduce that slightly more vehicles are needed when time preferences are considered. Additional experiments without the use of extra vehicles indicate that, on average, this has a negative effect on both distance and preference satisfaction. This can be explained by the need for extra vehicles to be able to satisfy the preferences of customers. This result is also visible in Section 5.4 where more vehicles are needed when time windows are exogenous.

Table 8: Comparison of the different penalty settings

	TP0	TP1	TP2	TP3
Average number of vehicles	8.9	8.9	9.1	9.2
Average distance	909.2	933.1	924.0	928.4
Average fraction of satisfied customers (%)	63.2	68.3	81.6	82.3
Average deviation from preference (%)	8.2	6.4	2.2	2.0

Considering the first two settings, we observe that incorporating time preferences during the scheduling phase increases the distance, but has a positive effect on the satisfaction of time preferences regarding both evaluation criteria. Comparing this to the results of TP2 indicates that when the time preference is also considered during the route generation phase, this has a substantial positive effect on the satisfaction of the preferences of the customers. The average distance obtained when using the TP2 setting decreases compared to the TP1 setting and increases with only 1.6% on average compared to the TP0 setting. This suggests that when time preferences are considered, it is important to incorporate those during the route generation

phase as well. The effect of increasing the penalty costs can be deduced from the comparison of the last two settings. This indicates that increasing the penalty costs does not have a significant effect on the satisfaction of preferences, nor on the distance.

Comparing the TP0 setting with the TP2 setting provides us with insight on the effect of time preferences. On average, the distance increases with only 1.6%. Considering the preference satisfaction on the other hand, the average deviation from the preference declines with 73.2% and the percentage of customers whose preference is satisfied increases with 29.1%. For this reason, we can conclude that taking into account the time preferences of customers in both phases, increases the percentage of customers of which the time preferences are satisfied. We should be careful to conclude that the customer satisfaction is also improved. When customers are provided with the option to indicate their preferences, they might expect that their preferences are likely to be satisfied. Due to higher expectations we are unable to conclude what the actual effect on customer satisfaction is. However, the differences in our evaluation criteria are substantial enough to suspect that customer satisfaction increases.

6 Conclusion

In this thesis we analyzed the vehicle routing problem with self-imposed time windows and the proposed solution procedure by Jabali et al. (2015). We implemented their hybrid algorithm, consisting of a routing and a scheduling phase. The routing phase is implemented via a tabu search heuristic, with 2-opt* and Or-opt neighborhood generation methods. Scheduling is performed by solving an LP problem that implicitly inserts time buffers into a scheduled route.

Our first contribution was to analyze the experiments discussed by Jabali et al. (2015). Our analysis indicated that there were major differences between their results and the results of our experiments. We attributed these differences to the interpretation of the neighborhood generation methods. As we stated, small differences in implementation have a large effect of the solution on the heuristic.

In addition to this, we investigated the effect of different initial solutions. Our experiments argue that both the objective value and the average computation time can be reduced by implementing a different initial solution method compared to the nearest neighbor heuristic that was implemented by Jabali et al. (2015). The best result among the initial solutions we implemented was achieved by the nearest neighbor heuristic with additional time constraints.

Furthermore, we implemented a random-sized tabu list to evaluate the importance of the optimal size of the tabu list. Our experiments indicated that, when no specific estimation of the optimal tabu list size can be made, random size within a well specified interval produces satisfying results.

Lastly, we included time preferences into our model. In this way, we can deal with exogenous time windows as a suggestion rather than a condemnation, which is the case in the VRPTW. We assumed that service before and after the preferred time window are both penalized. Due to the incorporation of time preferences, we extended the LP problem and proposed some alterations to the solution procedure. Our results indicate that incorporating time preferences in both stages of the problem increases the likeliness of satisfying a customers preference and decreases the average deviation from the preference of the customer. However, because of different expectations in a situation where customers can indicate their preferences, we cannot conclude with absolute certainty that overall customer satisfaction increases by incorporating time preferences.

For further research on this topic it would be interesting to analyze past customer behavior, in order to forecast their time preferences. In this case, the model including time preferences will indeed increases customer satisfaction, because the expectations of the customer will not change. Other topics of future research are to analyze the effect of time dependent travel times, as well as the inclusion of backhauls. Both of these topics are well studied in the context of vehicle routing problems, and would be interesting to investigate in the context of the VRP-SITW.

7 Appendix

Table 9: Comparison of the three move selection criteria for the Augerat instances

Instance	Objective value			CPU time (milliseconds)			
	C_1	C_2	C_3	C_1	C_2	C_3	Total
32 k5	1,466.5	1,372.5	1,466.5	2,529	3,878	4,409	10,816
33 k5	910.1	910.1	910.1	1,668	2,766	3,117	7,551
33 k6	987.5	992.3	987.5	1,907	3,518	3,819	9,244
34 k5	1,151.2	1,145.1	1,151.2	2,263	2,894	4,150	9,307
36 k5	1,389.6	1,429.4	1,389.6	1,890	3,924	3,474	9,288
37 k5	1,183.9	1,183.9	1,183.9	1,815	3,544	3,439	8,798
37 k6	1,471.0	1,402.9	1,471.0	3,678	3,887	7,046	14,611
38 k5	1,190.9	1,190.7	1,190.9	1,828	3,494	3,528	8,850
39 k5	1,533.2	1,535.0	1,533.2	2,409	4,753	4,411	11,573
39 k6	1,450.8	1,450.8	1,450.8	2,330	4,216	4,114	10,660
44 k6	1,533.6	1,545.9	1,533.6	2,192	3,938	4,078	10,208
45 k6	1,358.6	1,356.9	1,358.6	2,354	4,285	4,105	10,744
45 k7	1,875.1	1,859.5	1,875.1	2,675	4,650	4,563	11,888
46 k7	1,341.1	1,326.8	1,341.1	2,646	5,081	4,461	12,188
48 k7	1,795.6	1,786.6	1,795.6	2,697	4,743	4,489	11,929
53 k7	1,622.0	1,795.6	1,622.0	8,829	5,298	14,956	29,083
54 k7	2,178.1	2,121.6	2,178.1	5,423	5,970	9,360	20,753
55 k9	1,471.2	1,466.0	1,471.2	2,938	5,719	5,251	13,908
60 k9	2,144.1	2,233.5	2,144.1	6,215	6,526	10,618	23,359
61 k9	1,270.6	1,312.8	1,270.6	3,556	5,857	6,064	15,477
62 k8	2,278.7	2,211.4	2,278.7	4,849	13,974	8,469	27,292
63 k10	1,997.1	1,866.7	1,997.1	5,544	7,170	9,435	22,149
63 k9	2,880.3	2,960.6	2,880.3	4,206	9,736	7,615	21,557
64 k9	2,391.9	2,371.5	2,391.9	5,339	8,086	9,570	22,995
65 k9	1,583.4	1,580.6	1,583.4	4,293	7,610	7,343	19,246
69 k9	1,646.5	1,674.1	1,646.5	4,506	8,934	7,557	20,997
80 k10	3,486.9	3,516.5	3,486.9	10,800	10,504	17,855	39,159

Table 10: Comparison of the three move selection criteria for the Solomon instances

Instance	Objective value			CPU time (milliseconds)			
	C_1	C_2	C_3	C_1	C_2	C_3	Total
R101	1,344.4	1,143.2	1,346.6	22,123	57,852	25,693	105,668
R102	1,344.0	1,142.6	1,346.1	20,047	63,451	25,493	108,991
R103	1,343.1	1,141.7	1,345.1	18,613	59,319	22,606	100,538
R104	1,342.0	1,140.8	1,343.6	17,059	55,839	16,987	89,885
R105	1,341.4	1,140.3	1,341.4	15,505	49,777	19,062	84,344
R106	1,341.4	1,140.3	1,341.4	15,268	48,002	18,488	81,758
R107	1,341.4	1,140.3	1,341.4	15,088	41,912	18,432	75,432
R108	1,341.4	1,140.3	1,341.4	15,170	25,557	18,585	59,312
R109	1,341.4	1,140.3	1,341.4	15,288	25,366	18,650	59,304
R110	1,341.4	1,140.3	1,341.4	15,290	25,472	19,446	60,208
R111	1,341.4	1,140.3	1,341.4	15,190	25,362	18,456	59,008
R112	1,341.4	1,140.3	1,341.4	15,169	25,327	18,369	58,865
C101	826.0	826.0	826.0	14,753	22,556	17,409	54,718
C102	826.0	826.0	826.0	14,706	22,560	17,311	54,577
C103	826.0	826.0	826.0	14,621	22,189	17,023	53,833
C104	826.0	826.0	826.0	14,525	22,237	16,986	53,748
C105	826.0	826.0	826.0	14,668	22,580	17,250	54,498
C106	826.0	826.0	826.0	15,058	22,705	17,447	55,210
C107	826.0	826.0	826.0	14,710	27,709	18,016	60,435
C108	826.0	826.0	826.0	14,741	37,932	17,636	70,309
C109	826.0	826.0	826.0	14,728	22,195	17,100	54,023
RC101	1,154.3	1,154.3	1,154.3	16,513	23,922	19,660	60,095
RC102	1,154.3	1,154.3	1,154.3	16,360	23,996	19,542	59,898
RC103	1,154.3	1,154.3	1,154.3	16,308	23,938	17,091	57,337
RC104	1,154.3	1,154.3	1,154.3	16,855	23,630	19,285	59,770
RC105	1,155.2	1,155.2	1,157.2	31,867	25,519	14,652	72,038
RC106	1,154.3	1,154.3	1,154.3	25,725	23,562	19,364	68,651
RC107	1,154.3	1,154.3	1,154.3	26,960	23,437	19,231	69,628
RC108	1,154.3	1,154.3	1,154.3	30,425	23,433	19,107	72,992

Table 11: Comparison of the four penalty settings for the Augerat instances

Instance	Objective value				Penalty Ratio (%)			
	P5	P10	Prop	1.3dist	P5	P10	Prop	1.3dist
32 k5	1,372.5	1,375.2	1,375.9	2,082.6	39.4	39.5	39.5	48.1
33 k5	910.1	910.1	910.1	1,444.1	23.1	23.1	23.1	38.1
33 k6	987.5	987.5	987.5	1,507.6	24.5	24.5	24.5	35.7
34 k5	1,145.1	1,145.2	1,145.2	1,847.6	30.9	30.9	30.9	44.3
36 k5	1,389.6	1,389.6	1,389.7	2,057.9	41.6	41.6	41.6	48.8
37 k5	1,183.9	1,183.9	1,183.9	1,750.2	40.4	40.4	40.4	47.3
37 k6	1,402.9	1,403.8	1,404.1	2,354.5	29.2	29.3	29.3	44.7
38 k5	1,190.7	1,190.7	1,190.7	1,830.8	33.4	33.4	33.4	45.8
39 k5	1,533.2	1,533.2	1,533.2	2,222.3	45.8	45.8	45.8	50.9
39 k6	1,450.8	1,450.8	1,450.8	2,236.8	38.5	38.5	38.5	48.1
44 k6	1,533.6	1,534.3	1,534.4	2,349.5	32.0	32.1	32.1	42.3
45 k6	1,356.9	1,358.8	1,359.6	2,113.8	26.2	26.3	26.4	39.4
45 k7	1,859.5	1,862.1	1,864.8	2,825.0	37.0	37.1	37.2	46.5
46 k7	1,326.8	1,327.5	1,328.6	2,052.9	30.4	30.4	30.5	41.6
48 k7	1,786.6	1,788.2	1,788.7	2,751.7	35.1	35.2	35.2	46.3
53 k7	1,622.0	1,622.7	1,622.7	2,733.9	35.6	35.6	35.6	46.9
54 k7	2,121.6	2,123.1	2,124.9	3,166.2	42.1	42.1	42.1	49.9
55 k9	1,466.0	1,466.0	1,466.0	2,217.0	24.0	24.0	24.0	32.5
60 k9	2,144.1	2,147.6	2,151.7	3,184.8	34.3	34.4	34.5	43.9
61 k9	1,270.6	1,270.6	1,270.6	2,016.9	14.5	14.5	14.5	31.2
62 k8	2,211.4	2,214.6	2,216.3	3,425.5	39.6	39.7	39.8	49.0
63 k10	1,866.7	1,867.1	1,867.5	2,896.0	26.0	26.0	26.1	38.4
63 k9	2,880.3	2,886.8	2,888.9	4,285.0	41.9	42.0	42.0	50.0
64 k9	2,371.5	2,374.7	2,374.5	3,530.2	38.9	39.0	39.0	46.1
65 k9	1,580.6	1,581.1	1,581.3	2,585.8	23.8	23.8	23.9	37.5
69 k9	1,646.5	1,646.5	1,646.5	2,605.5	26.2	26.2	26.2	40.5
80 k10	3,486.9	3,491.8	3,491.9	5,055.7	46.9	47.0	47.0	52.6
Average					33.4	33.4	33.4	43.9

Table 12: Comparison of distance of VRP-SITW with the distance in the optimal VRP solutions for the Augerat instances

Instance	P5 (%)	P10 (%)	Prop (%)	1.3dist (%)
32 k5	106.2	106.2	106.2	106.2
33 k5	105.8	105.8	105.8	104.0
33 k6	100.5	100.5	100.5	100.5
34 k5	101.7	101.7	101.7	101.7
36 k5	101.5	101.5	101.5	101.4
37 k5	100.9	100.9	100.9	101.5
37 k6	104.6	104.6	104.6	105.6
38 k5	108.7	108.7	108.7	104.6
39 k5	101.1	101.1	101.1	102.1
39 k6	107.5	107.5	107.5	107.5
44 k6	111.3	111.3	111.3	111.3
45 k6	106.0	106.0	106.0	104.4
45 k7	102.2	102.2	102.2	101.4
46 k7	101.0	101.0	101.0	100.8
48 k7	108.0	108.0	108.0	105.9
53 k7	103.3	103.4	103.4	110.5
54 k7	105.4	105.4	105.4	104.5
55 k9	103.9	103.9	103.9	107.2
60 k9	104.1	104.1	104.1	101.6
61 k9	105.0	105.0	105.0	103.1
62 k8	103.5	103.5	103.5	104.1
63 k10	105.0	105.0	105.0	104.4
63 k9	102.5	102.5	102.5	100.8
64 k9	103.3	103.3	103.3	104.4
65 k9	102.3	102.3	102.3	105.6
69 k9	104.0	104.0	104.0	102.1
80 k10	104.9	104.9	104.9	104.5
Average	104.2	104.2	104.2	104.1

Table 13: Comparison of VRP-SITW with the best known VRPTW solutions for the Solomon instances

Instance	T_F	T_S/T_F (%)	K_F	$K_F - K_S$	Penalty ratio (%)
R101	1,637.7	52.5	20	12	24.8
R102	1,466.6	58.6	18	10	24.7
R103	1,208.7	71.2	14	6	24.7
R104	971.5	88.5	11	3	24.6
R105	1,355.3	63.5	15	7	24.6
R106	1,252.0	68.7	12	4	24.6
R107	1,064.6	80.8	11	3	24.6
R108	960.9	89.5	9	1	24.6
R109	1,146.9	75.0	13	5	24.6
R110	1,068.0	80.5	12	4	24.6
R111	1,048.7	82.0	12	4	24.6
R112	982.1	87.6	9	1	24.6
C101	827.3	99.8	10	0	0.0
C102	827.3	99.8	10	0	0.0
C103	826.3	100.0	10	0	0.0
C104	822.9	100.4	10	0	0.0
C105	827.3	99.8	10	0	0.0
C106	827.3	99.8	10	0	0.0
C107	827.3	99.8	10	0	0.0
C108	827.3	99.8	10	0	0.0
C109	827.3	99.8	10	0	0.0
RC101	1,619.8	63.3	15	6	11.1
RC102	1,457.4	70.4	14	5	11.1
RC103	1,258.0	81.5	13	4	11.1
RC104	1,261.7	81.3	11	2	11.1
RC105	1,513.7	67.8	15	6	11.2
RC106	1,424.7	72.0	11	2	11.1
RC107	1,207.8	84.9	12	3	11.1
RC108	1,114.2	92.1	11	2	11.1
Average		83.1			13.3

T_F = distance for VRPTW, T_S = distance for VRP-SITW,
K_F = number of vehicles used for VRPTW,
K_S = number of vehicles used for VRP-SITW

Table 14: Comparison of the different initial solutions for the Solomon instances using the C_1 criterion

Instance	Objective value				CPU time (milliseconds)			
	NN	NN time	Sweep	Saving	NN	NN time	Sweep	Saving
R101	1,344.4	1,088.8	1,072.3	1,295.2	22,123	12,749	15,203	26,658
R102	1,344.0	1,088.5	1,072.0	1,294.2	20,047	11,242	15,322	42,796
R103	1,343.1	1,087.8	1,071.7	1,292.6	18,613	11,045	12,996	33,762
R104	1,342.0	1,086.6	1,071.3	1,290.9	17,059	12,466	13,840	19,056
R105	1,341.4	1,086.1	1,071.1	1,289.5	15,505	11,769	12,046	17,545
R106	1,341.4	1,086.1	1,071.1	1,289.5	15,268	10,899	12,089	17,759
R107	1,341.4	1,086.0	1,071.1	1,289.5	15,088	11,467	11,548	17,715
R108	1,341.4	1,085.9	1,071.1	1,289.5	15,170	11,535	10,940	17,353
R109	1,341.4	1,085.9	1,071.1	1,289.5	15,288	11,207	11,694	17,249
R110	1,341.4	1,085.9	1,071.1	1,289.5	15,290	10,771	11,527	17,175
R111	1,341.4	1,085.9	1,071.1	1,289.5	15,190	10,519	11,596	17,324
R112	1,341.4	1,085.9	1,071.1	1,289.5	15,169	11,051	10,790	17,027
C101	826.0	825.2	923.0	972.2	14,753	10,667	10,655	10,430
C102	826.0	825.2	923.0	972.2	14,706	10,674	10,442	10,420
C103	826.0	825.2	923.0	972.2	14,621	9,730	10,755	10,709
C104	826.0	825.2	923.0	972.1	14,525	9,751	10,440	17,925
C105	826.0	825.2	923.0	972.1	14,668	9,849	10,832	19,118
C106	826.0	825.2	923.0	972.1	15,058	10,671	10,852	20,117
C107	826.0	825.2	923.0	972.1	14,710	10,708	10,328	19,968
C108	826.0	825.2	923.0	972.1	14,741	10,738	10,739	18,542
C109	826.0	825.2	923.0	972.1	14,728	10,376	10,401	19,354
RC101	1,154.3	1,131.1	1,192.8	1,148.2	16,513	9,465	14,118	20,068
RC102	1,154.3	1,131.1	1,192.1	1,147.9	16,360	9,789	14,800	20,809
RC103	1,154.3	1,131.1	1,191.5	1,147.6	16,308	9,929	13,861	20,727
RC104	1,154.3	1,131.1	1,191.2	1,147.5	16,855	9,304	15,007	20,665
RC105	1,155.2	1,131.5	1,192.7	1,148.9	31,867	10,022	15,768	24,070
RC106	1,154.3	1,131.1	1,190.9	1,147.1	25,725	9,095	14,657	20,375
RC107	1,154.3	1,131.1	1,190.9	1,147.1	26,960	9,410	14,772	20,449
RC108	1,154.3	1,131.1	1,191.0	1,147.2	30,452	9,238	13,692	20,295
Average	1,130.1	1,017.8	1,058.5	1,152.4	17,702	10,556	12,473	19,843

NN = Nearest Neighbor Heuristic, NN time = Nearest Neighbor Heuristic with time constrains,
Sweep = Sweep Heuristic, Saving = Savings Heuristic

Table 15: Comparison of the two neighborhood generation methods using the C_1 criterion for the Augerat instances

Instance	Objective		CPU time (milliseconds)	
	Jabali et al. (2015)	Diversification	Jabali et al. (2015)	Diversification
32 k5	1,466.5	1,521.2	2,529	2,660
33 k5	910.1	933.7	1,668	1,997
33 k6	987.5	986.4	1,907	3,298
34 k5	1,151.2	1,154.4	2,263	6,746
36 k5	1,389.6	1,438.6	1,890	4,239
37 k5	1,183.9	1,179.8	1,815	7,546
37 k6	1,471.0	1,545.4	3,678	3,687
38 k5	1,190.9	1,259.1	1,828	3,791
39 k5	1,533.2	1,409.8	2,409	8,213
39 k6	1,450.8	1,426.1	2,330	4,831
44 k6	1,533.6	1,533.6	2,192	4,658
45 k6	1,358.6	1,358.6	2,354	4,533
45 k7	1,875.1	1,967.2	2,675	4,933
46 k7	1,341.1	1,342.3	2,646	5,289
48 k7	1,795.6	1,806.0	2,697	5,371
53 k7	1,622.0	1,809.9	8,829	6,731
54 k7	2,178.1	2,237.9	5,423	6,280
55 k9	1,471.2	1,480.7	2,938	6,191
60 k9	2,144.1	2,158.7	6,215	8,600
61 k9	1,270.6	1,274.2	3,556	7,438
62 k8	2,278.7	2,274.9	4,849	13,199
63 k10	1,997.1	2,102.2	5,544	8,595
63 k9	2,880.3	3,064.5	4,206	8,872
64 k9	2,391.9	2,444.3	5,339	26,609
65 k9	1,583.4	1,576.1	4,293	25,334
69 k9	1,646.5	1,626.1	4,506	24,272
80 k10	3,486.9	3,621.6	10,800	12,153

Table 16: Comparison of the different tabu list sizes κ for the C_1 criterion for the Augerat instances

Instance	Objective value				CPU time (milliseconds)			
κ	20	[10,30]	[10,40]	[10,50]	20	[10,30]	[10,40]	[10,50]
32 k5	1,466.5	1,466.5	1,466.5	1,466.5	2,529	2,564	2,576	2,702
33 k5	910.1	910.1	910.1	910.1	1,668	1,549	1,504	1,565
33 k6	987.5	987.5	987.5	987.5	1,907	2,078	2,146	2,114
34 k5	1,151.2	1,151.2	1,151.2	1,151.2	2,263	2,231	2,232	2,292
36 k5	1,389.6	1,389.6	1,389.6	1,389.6	1,890	1,911	1,985	1,900
37 k5	1,183.9	1,183.9	1,183.9	1,183.9	1,815	1,855	1,874	1,869
37 k6	1,471.0	1,499.1	1,499.1	1,499.1	3,678	4,159	4,994	4,149
38 k5	1,190.9	1,190.9	1,190.9	1,190.9	1,828	1,787	1,800	1,788
39 k5	1,533.2	1,424.8	1,533.2	1,533.2	2,409	4,266	2,333	2,380
39 k6	1,450.8	1,450.8	1,450.8	1,450.8	2,330	2,246	2,306	2,230
44 k6	1,533.6	1,533.6	1,533.6	1,533.6	2,192	2,167	2,156	2,132
45 k6	1,358.6	1,358.6	1,358.6	1,358.6	2,354	2,283	2,287	2,278
45 k7	1,875.1	1,875.1	1,875.1	1,875.1	2,675	2,566	2,582	2,577
46 k7	1,341.1	1,341.1	1,341.1	1,341.1	2,646	2,513	2,540	2,506
48 k7	1,795.6	1,795.6	1,795.6	1,795.6	2,697	2,602	2,552	2,584
53 k7	1,622.0	1,785.4	1,675.7	1,675.7	8,829	2,950	5,894	5,613
54 k7	2,178.1	2,237.9	2,159.6	2,237.9	5,423	2,981	4,673	3,033
55 k9	1,471.2	1,471.2	1,471.2	1,471.2	2,938	2,980	2,995	2,988
60 k9	2,144.1	2,135.9	2,144.1	2,144.1	6,215	4,818	6,278	6,291
61 k9	1,270.6	1,270.6	1,270.6	1,270.6	3,556	3,593	3,634	3,696
62 k8	2,278.7	2,278.7	2,259.8	2,278.7	4,849	5,442	6,863	5,000
63 k10	1,997.1	1,976.9	1,976.9	1,976.9	5,544	4,996	5,081	5,054
63 k9	2,880.3	2,866.1	2,880.3	2,866.1	4,206	5,683	4,200	5,290
64 k9	2,391.9	2,391.9	2,391.9	2,391.4	5,339	6,273	5,488	8,786
65 k9	1,583.4	1,583.4	1,583.4	1,583.4	4,293	4,588	4,313	4,263
69 k9	1,646.5	1,646.5	1,646.5	1,646.5	4,506	4,891	4,565	4,675
80 k10	3,486.9	3,486.9	3,494.3	3,486.9	10,800	14,730	9,483	14,303
Average	1,688.5	1,692.2	1,689.7	1,692.5	3,755	3,730	3,679	3,854

References

Augerat, P., Belenguer, J.-M., Benavent, E., Corbéran, A., and Naddef, D. (1998). Separating capacity constraints in the cvrp using tabu search. *European Journal of Operational Research*, 106(2-3):546–557.

Chiang, W.-C. and Russell, R. A. (1997). A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS Journal on computing*, 9(4):417–430.

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

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, 52(8):928–936.

Gillett, B. E. and Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, 22(2):340–349.

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

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

Laporte, G., Gendreau, M., Potvin, J.-Y., and Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International transactions in operational research*, 7(4-5):285–300.

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

Mitrović-Minić, S. and Laporte, G. (2004). Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(7):635–655.

Or, I. (1976). Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking. 11(2):86–95.

Potvin, J.-Y., Kervahut, T., Garcia, B.-L., and Rousseau, J.-M. (1996). The vehicle routing problem with time windows part i: tabu search. *INFORMS Journal on Computing*, 8(2):158–164.

Potvin, J.-Y. and Rousseau, J.-M. (1995). An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, 46(12):1433–1446.

Ralphs, T. (2010). Branch cut and price resource web. http://www.branchandcut.org. Accessed: May 2018.

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

Solomon, M. M. (2010). Vrptw benchmark problems. http://w.cba.neu.edu/ msolomon/problems.htm. Accessed: May 2018.

Taillard, É. (1991). Robust taboo search for the quadratic assignment problem. *Parallel computing*, 17(4-5):443–455.