IMM
INFORMATICS AND MATHEMATICAL MODELLING

Technical University of Denmark
DK-2800 Kgs. Lyngby - Denmark

2001-08-23
jla

LAGRANGEAN DUALITY APPLIED ON VEHICLE ROUTING WITH TIME WINDOWS

EXPERIMENTAL RESULTS

Brian Kallehauge
Jesper Larsen
Oli B.G. Madsen

TECHNICAL REPORT

IMM-TR-2001-9

IMM

LAGRANGEAN DUALITY APPLIED ON VEHICLE ROUTING WITH TIME WINDOWS

EXPERIMENTAL RESULTS

Brian Kallehauge
Jesper Larsen
Oli B.G. Madsen

TECHNICAL REPORT

IMM-TR-2001-9

IMM

Abstract

This paper presents the results of the application of a non-differentiable optimization method in connection with the Vehicle Routing Problem with Time Windows (VRPTW). The VRPTW is an extension of the Vehicle Routing Problem. In the VRPTW the service at each customer must start within an associated time window.

The Shortest Path decomposition of the VRPTW by Lagrangian relaxation require the finding of the optimal Lagrangian multipliers. This problem is a convex non-differentiable optimization problem. We propose a cutting-plane algorithm with a trust-region stabilizing device for finding the optimal multipliers.

The cutting-plane algorithm with trust-region has been combined with a Dantzig-Wolfe algorithm for finding integer solutions in a branch-and-bound scheme. The root node of the branch-and-bound tree is solved by the cutting-plane algorithm and, if an integer solution is not obtained, shifting to a Dantzig-Wolfe algorithm in the tree nodes occurs. The combined cutting-plane and Dantzig-Wolfe algorithm has been tested on the well-known Solomon VRPTW benchmark problems and a range of extended Solomon problems created by Homberger.

We have succeeded in solving 14 previously unsolved Solomon problems and a Homberger problem with 1000 customers, which is the largest problem ever solved to optimality. The computational times were reduced significantly by the cutting-plane algorithm in the root node compared to the Dantzig-Wolfe method due to easier subproblems. It therefore seems very efficient to stabilize the dual variables.

KEYWORDS: Lagrangian relaxation, duality, non-differentiable optimization, cutting plane method, trust region method, Vehicle Routing Problem with Time Windows.

1 Introduction

Many companies are faced with problems regarding the transportation of people, goods or information – commonly denoted routing problems. This is not restricted to the transport sector itself but also other companies e.g. factories may have transport of parts to and from different sites of the factory, and big companies may have internal mail deliveries. These companies have to optimize transportation. As the world economy turns more and more global, transportation will become even more important in the future.

Back in 1983 Bodin et al. reported that in 1980 approximately $400 billion were used in distribution cost in the United States and in the United Kingdom the corresponding figure was £15 billion. Halse (1992) reports from an article from the Danish newspaper Berlingske Tidende that in 1989 76.5% of all the transportation of goods was done by vehicles, which underlines the importance of routing and scheduling problems.

Fisher (1995) writes that a study from the National Council of Physical Distribution estimates that transportation accounts for 15% of the U.S. gross national product (1978). In Denmark the figures are 13% for 1981 and 15% for 1994 according to The Danish Ministry of Transport (1998).

In a *pure* routing problem there is only a *geographic* component, more realistic routing problems also include a scheduling part, that is, a time component.

The simplest routing problem is the Traveling Salesman Problem (or TSP). A number of cities have to be visited by a salesman who has to return to the city where he started.

The route has to be constructed in order to minimize the distance to be traveled. In the m-TSP problem, m salesmen have to cover the cities given. Each city must be visited by exactly one salesman. Every salesman starts off from the same city (called the depot) and must at the end of his journey return to this city again. We now want to minimize the sum of the distances of the routes. Both the TSP and m-TSP problems are pure routing problems in the sense defined above.

The Vehicle Routing Problem (or VRP) is the m-TSP where a demand is associated with each city, and each vehicle have a certain capacity (not necessarily identical). Be aware that during the later years a number of authors have "renamed" this problem the Capacitated Vehicle Routing Problem (or CVRP). The sum of demands on a route can not exceed the capacity of the vehicle assigned to this route. As in the m-TSP we want to minimize the sum of distances of the routes. Note that the VRP is not purely geographic since the demand may be constraining. The VRP is the basic model for a large number of vehicle routing problems.

If we add a *time window* to each customer we get the Vehicle Routing Problem with Time Windows (VRPTW). In addition to the capacity constraint, a vehicle now has to service a customer within a certain time frame. The vehicle may arrive before the time window "opens" but the customer can not be serviced until the time windows "opens". It is not allowed to arrive after the time window has "closed".

These problems are all "hard" to solve (ie. the problems are $\mathcal{NP}$*-hard*). For the VRPTW exact solutions can be found within reasonable time for some instances up to about 100 customers. A review of exact methods for the VRPTW is given by Larsen (1999).

If the term "vehicle" is considered more loosely, numerous scheduling problems can also be regarded as VRPTW. An example is that for a single machine, we want to schedule a number of jobs where we know the flow time and the time to go from running one job to the next one. This scheduling problem can be regarded as a VRPTW with a single depot, single vehicle and the customers represents the jobs. The cost of changing from one job to another is equal to the distance between the two customers. The time it takes to perform the action is the service time of the job.

For a general and in-depth description of the field of routing and scheduling see Desrosiers, Dumas, Solomon, and Soumis (1995), Breedam (1995) and Crainic and Laporte (1998).

2 The Vehicle Routing Problem with Time Windows

The VRPTW is given by a fleet of homogeneous vehicles (denoted $\mathcal{V}$), a set of customers $\mathcal{C}$ and a directed graph $\mathcal{G}$. The graph consists of $|\mathcal{C}| + 2$ vertices, where the customers are denoted $1, 2, \ldots, n$ and the depot is represented by the vertex 0 ("the driving-out depot") and $n+1$ ("the returning depot"). The set of vertices, that is, $0, 1, \ldots, n+1$ is denoted $\mathcal{N}$. The set of arcs (denoted $\mathcal{A}$) represents connections between the depot and the customers and among the customers. No arc terminates in vertex 0, and no arc originates from vertex $n + 1$. With each arc (i, j), where $i \neq j$, we associate a *cost* c_{ij} and a *time* t_{ij}, which may include service time at customer i.

Each vehicle has a capacity q and each customer i a demand d_i. Each customer i has a

time window $[a_i, b_i]$. A vehicle must arrive at the customer before b_i. It can arrive before a_i but the customer will not be serviced before. The depot also has a time window $[a_0, b_0]$ (the time windows for both depots are assumed to be identical). $[a_0, b_0]$ is called the *scheduling horizon*. Vehicles may not leave the depot before a_0 and must be back before or at time b_{n+1}.

It is assumed that q, a_i, b_i, d_i, c_{ij} are non-negative integers, while the t_{ij}'s are assumed to be positive integers. It is also assumed that the triangular inequality is satisfied for both the c_{ij}'s and the t_{ij}'s. It is possible to add a scalar to all transportations costs c_{ij} without changing the optimal solution to VRPTW as the triangular inequality still holds.

The model contains two sets of decision variables x and s. For each arc (i, j), where $i \neq j, i \neq n+1, j \neq 0$, and each vehicle k we define x_{ijk} as

$$x_{ijk} = \begin{cases} 0, & \text{if vehicle } k \text{ does not drive from vertex } i \text{ to vertex } j \\ 1, & \text{if vehicle } k \text{ drives directly from vertex } i \text{ to vertex } j. \end{cases}$$

The decision variable s_{ik} is defined for each vertex i and each vehicle k and denotes the time vehicle k starts to service customer i. In case the given vehicle k does not service customer i s_{ik} does not mean anything. We assume $a_0 = 0$ and therefore $s_{0k} = 0$, for all k.

We want to design a set of minimal cost routes, one for each vehicle, such that

- each customer is serviced exactly once,
- every route originates at vertex 0 and ends at vertex $n+1$, and
- the time windows and capacity constraints are observed.

We can state the VRPTW mathematically as:

$$z_{VRPTW} = \text{minimize} \sum_{k \in \mathcal{V}} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} c_{ij} x_{ijk} \tag{1}$$

subject to

$$\sum_{k \in \mathcal{V}} \sum_{j \in \mathcal{N}} x_{ijk} = 1 \qquad \forall i \in \mathcal{C} \tag{2}$$

$$\sum_{i \in \mathcal{C}} d_i \sum_{j \in \mathcal{N}} x_{ijk} \leq q \qquad \forall k \in \mathcal{V} \tag{3}$$

$$\sum_{j \in \mathcal{N}} x_{0jk} = 1 \qquad \forall k \in \mathcal{V} \tag{4}$$

$$\sum_{i \in \mathcal{N}} x_{ihk} - \sum_{j \in \mathcal{N}} x_{hjk} = 0 \qquad \forall h \in \mathcal{C}, \forall k \in \mathcal{V} \tag{5}$$

$$\sum_{i \in \mathcal{N}} x_{i,n+1,k} = 1 \qquad \forall k \in \mathcal{V} \tag{6}$$

$$s_{ik} + t_{ij} - K(1 - x_{ijk}) \leq s_{jk} \qquad \forall i, j \in \mathcal{N}, \forall k \in \mathcal{V} \tag{7}$$

$$a_i \leq s_{ik} \leq b_i \qquad \forall i \in \mathcal{N}, \forall k \in \mathcal{V} \tag{8}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i, j \in \mathcal{N}, \forall k \in \mathcal{V}. \tag{9}$$

Constraints (2) states that each customer is visited exactly once, and (3) means that no vehicle is loaded with more than it's capacity allows it to. The next three equations (4), (5) and (6) ensures that each vehicle leaves the depot 0, after arriving at a customer the vehicle leaves again, and finally arrives at the depot $n+1$. The inequalities (7) states that a vehicle k can not arrive at j before $s_{ik} + t_{ij}$ if it is traveling from i to j. Here K is a large scalar. Finally constraints (8) ensures that time windows are observed, and (9) are the integrality constraints. Note that an unused vehicle is modelled by driving the "empty" route $(0, n+1)$.

If we remove the assignment constraints (2) the problem becomes an Elementary Shortest Path Problem with Time Windows and Capacity Constraints (ESPPTWCC) for every vehicle, that is, find the shortest path from the depot and back to the depot that does not violate the time and capacity constraints and visits the customers on the route at most one time. As all vehicles are identical all ESPPTWCC's also become identical.

Using the column generation approach as introduced with the set partitioning problem as the master problem, the subproblem becomes the following mathematical model:

$$\text{minimize} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \hat{c}_{ij} x_{ij} \tag{10}$$

subject to

$$\sum_{i \in \mathcal{C}} d_i \sum_{j \in \mathcal{N}} x_{ij} \leq q \tag{11}$$

$$\sum_{j \in \mathcal{N}} x_{0j} = 1 \tag{12}$$

$$\sum_{i \in \mathcal{N}} x_{ih} - \sum_{j \in \mathcal{N}} x_{hj} = 0 \quad \forall h \in \mathcal{C} \tag{13}$$

$$\sum_{i \in \mathcal{N}} x_{i,n+1} = 1 \tag{14}$$

$$s_i + t_{ij} - K(1 - x_{ij}) \leq s_j \; \forall i, j \in \mathcal{N} \tag{15}$$

$$a_i \leq s_i \leq b_i \quad \forall i \in \mathcal{N} \tag{16}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in \mathcal{N}. \tag{17}$$

Constraints (11) are the capacity constraint, constrains (15) and (16) are time constraints, while constraints (17) ensures integrality. The constraints (12), (13) and (14) are flow constraints resulting in a path from the depot 0 and back to the depot $n+1$. The $\hat{c}_{ij}$ is the *modified cost* of using arc (i, j), where $\hat{c}_{ij} = c_{ij} - \pi_i$, where π_i is the dual cost of customer i. Note that while c_{ij} is a non-negative integer, $\hat{c}_{ij}$ can be any real number. As we are now working with one route the index k for the vehicle has been removed.

As can be seen from the mathematical model above the subproblem is a shortest path problem with time windows and capacity constraints, where each vertex can participate at most once in the route. For this problem there is no known efficient algorithm, making the problem unsolvable for practical purposes. Therefore some of the constraints are relaxed. Cycles are allowed thereby changing the problem to the Shortest Path Problem with Time Windows and Capacity Constraints (SPPTWCC). Even though there is a possibility for negative cycles in the graph the time windows and the capacity constraints prohibits infinite cycling. Note that capacity is accumulated every time the customer is serviced.

In order to build the SPPTWCC algorithm we have to make two assumptions:

1. Time is always increasing along the arcs, i.e. $t_{ij} > 0$.
2. Time and capacity are discretized.

The algorithm maintains a set of "shortest subpaths" defined by a list of labels. A label is a state that contains a customer number, the current time t of arrival (at the given customer) and the accumulated demand d:

(i, t, d).

The cost of the label is then defined as $c(i, t, d)$. The algorithm is based on the following simple extension of the dynamic programming behind the Dijkstra algorithm:

$$\begin{aligned} c(0,0,0) &= 0 \\ c(j,t,d) &= \min_i \{\hat{c}_{ij} + c(i,t',d') \mid t' + t_{ij} = t \wedge d' + d_i = d\} \end{aligned}$$

States are treated in order of increasing time (t). Note that for each label i there may now exist more than one state. An upper bound on the number of states is given by

$$\Gamma = \sum_{i \in \mathcal{N}} (b_i - a_i)(q - 1)$$

As this is the upper limit, many of these states might not be possible, and others will not be considered as they are dominated by other states (see later).

In a straightforward implementation we maintain a set *NPS* of unprocessed states. Initially this set only has one member: the label $(0, 0, 0)$. As long as there exist unprocessed labels in the set the one with the lowest time is chosen and the algorithm tries to extend this to the successors of the vertex. States at vertex $n + 1$ are not processed and are therefore kept in a special set of "solutions", from which the best one is returned as the algorithm terminates. When a label has been processed it is removed from the set of unprocessed labels. The algorithm is described in pseudo-code in figure 1.

In order to make the algorithm considerably more efficient we will (like in Dijkstra's algorithm) introduce a *dominance criterion*.

Assume that for a given vertex i we have two states (i, t_1, d_1) and (i, t_2, d_2) where $c(i, t_1, d_1) \leq c(i, t_2, d_2)$, $t_1 \leq t_2$ and $d_1 \leq d_2$. Clearly as long as the extensions based on (i, t_2, d_2) are valid the extensions based on (i, t_1, d_1) are also valid, and these will always be lower in cost (or at least not higher). Therefore the label (i, t_2, d_2) can be discarded. Formally we say that (i, t_1, d_1) *dominates* (i, t_2, d_2) (or $(i, t_1, d_1) \prec (j, t_2, d_2)$) if and only if all of the following three conditions hold:

1. $c(i, t_1, d_1) \leq c(i, t_2, d_2)$.
2. $t_1 \leq t_2$.
3. $d_1 \leq d_2$.

Each time a new label is generated we have to check with the other labels at the same vertex to see if the new label is dominated by some label or the new label dominates another label.

```
⟨ Initialization ⟩
NPS= {(0, 0, 0)}
c(0, 0, 0) = 0

repeat
    (i, t, d) = BestLabel(NPS)

    for j := 1 to n + 1 do
        if (i ≠ j ∧ t + t_ij ≤ b_j ∧ d + d_j ≤ q) then
            ⟨ Label feasible ⟩
            if c(j, max{t + t_ij, a_j}, d + d_j) > c(i, t, d) + ĉ_ij then
                ⟨ New label better ⟩
                InsertLabel(NPS, (j, max{t + t_ij, a_j}, d + d_j))
                c(j, max{t + t_ij, a_j}, d + d_j) = c(i, t, d) + ĉ_ij

until (i = n + 1)
return
```

Figure 1: The algorithm for finding the shortest path with time windows and capacity constraints. *BestLabel* returns a label with vertex different from $n+1$ and minimal accumulated time if one exists. Otherwise a label with vertex $n+1$ is returned. *InsertLabel* inserts the newly generated label in *NPS* possibly overwriting an old label if it already exists.

3 Lagrangian Relaxation

We consider the Lagrangian relaxation of VRPTW with respect to the constraints (2), by introducing a vector of Lagrange multipliers $\lambda = (\lambda_1, \ldots, \lambda_n)$, where λ_i is associated with the ith constraint in (2):

$$z_D(\lambda) = \underset{\substack{\text{subject to} \\ \text{(3)-(9)}}}{\text{minimize}} \sum_{k\in\mathcal{V}}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} c_{ij}x_{ijk} - \sum_{i\in\mathcal{C}} \lambda_i \left(\sum_{k\in\mathcal{V}}\sum_{j\in\mathcal{N}} x_{ijk} - 1 \right). \tag{18}$$

We will call (18) the Lagrange problem. The minimal value in the Lagrange problem (18) is called the dual function and is denoted z_D. The set of feasible solutions to the Lagrange problem (18) defined by (3)-(9) is denoted P. P splits into $|V|$ disjoint subsets, i.e. $P = P_1 \times P_2 \times \cdots \times P_{|V|}$, where each P_k is defined by (11)-(17) for a given k. The Lagrange problem (18) therefore splits into $|V|$ "simpler" problems, one for each vehicle:

$$z_D(\lambda) = \sum_{k\in\mathcal{V}} z_k(\lambda) + \sum_{i\in C} \lambda_i, \tag{19}$$

where

$$z_k(\lambda) = \underset{\substack{\text{subject to} \\ (3)\text{-}(9)}}{\text{minimize}} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} (c_{ij} - \lambda_i) x_{ijk}, \text{ for } k = 1, \ldots, |V|. \tag{20}$$

Each z_k is determined by solving an ESPPTWCC on the graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, but the cost c_{ij} we previously assigned each arc (i, j) is modified, i.e. the ESPPTWCC is solved with respect to the modified arc costs $\hat{c}_{ij} = c_{ij} - \lambda_i$ for all $(i, j) \in \mathcal{A}$.

Since the set of feasible solutions to an ESPPTWCC is finite, we can consider each $z_k(\lambda)$ to be determined by minimization over the set P_k of constrained shortest paths. For a given vehicle $k \in V$ we describe each such path p with the integer variables x_{ijkp}, $(i, j) \in \mathcal{A}$. Let c_{kp} be the cost of path p for vehicle k and let a_{ikp} be the number of times customer i is served by vehicle k on path p:

$$\begin{aligned}
c_{kp} &= \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijkp}, \text{ for } p = 1, \ldots, |P_k| \text{ and } k = 1, \ldots, |V|, \\
a_{ikp} &= \sum_{j \in N} x_{ijkp}, \text{ for } i = 1, \ldots, |N|, p = 1, \ldots, |P_k| \text{ and } k = 1, \ldots, |V|,
\end{aligned}$$

then (20) is expressed as:

$$z_k(\lambda) = \underset{p \in P^k}{\text{minimize}}\, c_{kp} - \sum_{i \in C} a_{ikp} \lambda_i, \text{ for } k = 1, \ldots, |V|. \tag{21}$$

The fact that $z_D(\lambda) \leqslant z_{VRPTW}$, for $\lambda \in \mathbb{R}^n$, provides us with lower bounds in a branch-and-bound algorithm for the VRPTW. Clearly, we wish to find the best lower bound by solving the Lagrangian dual problem:

$$\begin{aligned}
z_{LD} &= \underset{\lambda \in \mathbb{R}^n}{\text{maximize}}\, z_D(\lambda) \\
&= \underset{\lambda \in \mathbb{R}^n}{\text{maximize}} \sum_{k \in \mathcal{V}} z_k(\lambda) + \sum_{i \in C} \lambda_i
\end{aligned} \tag{22}$$

where $z_k(\lambda)$ is given by (20) or (21).

Since each set P_k of feasible solutions to (21) is finite it allows us to express (22) as the following linear program with many constraints or rows:

$$\begin{aligned}
&z_{LD} = \underset{\substack{\lambda \in \mathbb{R}^n \\ \theta \in \mathbb{R}}}{\text{maximize}} \sum_{k \in V} \theta_k + \sum_{i \in C} \lambda_i \\
&\text{subject to} \\
&\theta_k \leqslant c_{kp} - \sum_{i \in C} a_{ikp} \lambda_i \quad \text{for all } p \in P_k \text{ and for all } k \in V.
\end{aligned} \tag{23}$$

The LP dual of (23) is a linear program with many variables or columns:

$$
\begin{aligned}
& z_{LD} = \text{minimize} \sum_{k \in V} \sum_{p \in P_k} c_{kp} y_{kp} && (24) \\
& \text{subject to} \\
& \sum_{k \in V} \sum_{p \in P_k} a_{ikp} y_{kp} = 1 \quad \text{for all } i \in C \\
& \sum_{p \in P_k} y_{kp} = 1 \quad \text{for all } k \in V \\
& y_{kp} \geqslant 0 \quad \text{for all } p \in P_k \text{ and for all } k \in V.
\end{aligned}
$$

Problem (24) with y_{kp} required to be integral is equivalent with the original VRPTW formulation (1)-(9). Problem (24) is the LP relaxation of the Dantzig-Wolfe decomposition obtained when any solution to the VRPTW is expressed as a non-negative convex combination of constrained paths. The method of Desrochers, Desrosiers, and Solomon (1992) can be characterized as column generation on the problem (24) and their formulation of the VRPTW can therefore be viewed as a Dantzig-Wolfe decomposition of the formulation (1)-(9).

If we consider the case where all vehicles are identical and therefore $z_1 = z_k$, for $k = 2, \ldots, |V|$, we get $\sum_{k \in \mathcal{V}} z_k(\lambda) = |V| z_1(\lambda)$, which means the Lagrange problem (18) can be expressed as:

$$
z_D(\lambda) = |V| \left(\underset{\substack{\text{subject to} \\ (11)\text{-}(17)}}{\text{minimize}} \sum_{i \in N} \sum_{j \in N} (c_{ij} - \lambda_i) x_{ij} \right) + \sum_{i \in C} \lambda_i = |V| \left(\underset{p \in P}{\text{minimize}}\, c_p - \sum_{i \in C} a_{ip} \lambda_i \right) + \sum_{i \in C} \lambda_i
$$

The Lagrangian dual problem is then:

$$
z_{LD} = \underset{\lambda \in \mathbb{R}^n}{\text{maximize}}\, |V| \left(\underset{p \in P}{\text{minimize}}\, c_p - \sum_{i \in C} a_{ip} \lambda_i \right) + \sum_{i \in C} \lambda_i,
$$

and the corresponding linear program is:

$$
\begin{aligned}
& z_{LD} = \underset{\substack{\lambda \in \mathbb{R}^n \\ \theta \in \mathbb{R}}}{\text{maximize}}\, |V| \theta + \sum_{i \in C} \lambda_i && (25) \\
& \text{subject to} \\
& \theta \leqslant c_p - \sum_{i \in C} a_{ip} \lambda_i \quad \text{for all } p \in P,
\end{aligned}
$$

for which we find the following LP dual:

$$
\begin{aligned}
& z_{LD} = \text{minimize} \sum_{p \in P} c_p y_p && (26) \\
& \text{subject to} \\
& \sum_{p \in P} a_{ip} y_p = 1 \quad \text{for all } i \in C \\
& \sum_{p \in P} y_p = |V| \\
& y_p \geqslant 0 \quad \text{for all } p \in P.
\end{aligned}
$$

Desrochers, Desrosiers, and Solomon (1992) assumes a homogeneous fleet of vehicles and therefore considers the set partitioning problem (26), but with out the constraint requiring a fixed number of vehicles. This is equivalent with setting the cost of the "empty tour" $c_{0,n+1} = 0$, and choosing an upper bound for the number of vehicles $|V|$ in (26). The "empty tour" is simply the tour directly from the source depot 0 to the sink depot $n + 1$, which is included in the feasibility set (11)-(17) of the ESPPTWCC formulation. Considering a free number of vehicles (25) can be expressed as:

$$z_{LD} = \underset{\lambda \in \mathbb{R}^n}{\text{maximize}} \sum_{i \in C} \lambda_i \tag{27}$$

subject to

$$\sum_{i \in C} a_{ip} \lambda_i \leqslant c_p \quad \text{for all } p \in P,$$

Let $x = \{x_{ijk} | i = 0, \ldots, n+1,\ j = 0, \ldots, n+1 \text{ and } k = 0, \ldots, |V| - 1\}$ and let x_p, $p \in P$ denote a solution to the Lagrange problem. Clearly, x_p is obtainable via the simpler decomposed Lagrange problem (19).

Now consider the Lagrange function or *Lagrangian*:

$$L(\lambda, x) = \sum_{k \in \mathcal{V}} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} c_{ij} x_{ijk} + \sum_{i \in \mathcal{C}} \lambda_i \left(1 - \sum_{k \in \mathcal{V}} \sum_{j \in \mathcal{N}} x_{ijk} \right). \tag{28}$$

which for a fixed solution x_p is an affine function in λ:

$$L(\lambda, x_p) = \langle s_p, \lambda \rangle + c_p \tag{29}$$

where $\langle \cdot, \cdot \rangle$ denotes the ordinary dot-product and $s_p = (s_{0,p}, \ldots, s_{n-1,p}) \in \mathbb{R}^n$, where

$$s_{ip} = 1 - \sum_{k \in V} \sum_{j \in N} x_{ijkp} = 1 - \sum_{k \in V} a_{ikp}, \tag{30}$$

and

$$c_p = \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijkp} = \sum_{k \in V} c_{kp}.$$

The dual function can then be expressed as:

$$z_D(\lambda) = \underset{1 \leqslant p \leqslant |P|}{\text{minimize}}\ L(\lambda, x_p) \tag{31}$$

which shows that the dual function z_D (18) is the minimum of a finite number of affine functions and is therefore piecewise affine and concave (Nemhauser and Wolsey 1988).

However, the function is nondifferentiable or nonsmooth at any $\lambda \in \mathbb{R}^n$ where the solution of the Lagrange problem (18) is not unique. This corresponds to any subproblem (20) of the decomposed Lagrange problem (19) having several shortest path solutions.

Several nonsmooth methods have already been applied to the dual problem (22), e.g. subgradient methods and bundle methods (Kohl 1995, Kohl and Madsen 1997). Previous to Kallehauge (2000a) these methods did not seem to be competitive compared to the Dantzig-Wolfe column generation method. In traditional Dantzig-Wolfe column generation the multipliers are not directly controlled but stabilizing techniques have been proposed (e.g. du Merle, Villeneuve, Desrosiers, and Hansen 1999), but no applications on VRPTW have been reported.

Remark 3.1 Maximizing (minimizing) a function given by the minimimum (maximum) of a finite number of affine functions is called the *maximin* (*minimax*) *problem*. However, the number of affine functions $|P|$ of (31) is very large, but is bounded by $2^{|V||N|^2}$, since the number of x-variables is $|V||N|^2$ and they are restricted to the values 0 or 1. □

The subdifferential of the dual function at λ is defined by the convex hull of the gradients of the Lagrange functions that give the minimal value:

$$\begin{aligned} \partial z_D(\lambda) &= \text{conv}\{\nabla L(\lambda, x_p) : L(\lambda, x_p) = z_D(\lambda)\} \\ &= \text{conv}\{s_p : \langle s_p, \lambda \rangle + c_p = z_D(\lambda)\} \end{aligned} \tag{32}$$

Neame (1999) and Neame, Boland, and Ralph (2000) presents an outer approximation of the subdifferential in connection with Lagrangian duality. First define an index set

$$P_E(\lambda) = \{p : \langle s_p, \lambda \rangle + c_p \leqslant z_D(\lambda) + E,\ E \geqslant 0\}, \tag{33}$$

which is the set of solutions to the Lagrange problem where the solution values are less or equal to the dual function value plus a positive constant. Then the outer approximation to the subdifferential is

$$\partial_E z_D(\lambda) = \text{conv}\{s_p : p \in P_E(\lambda)\}. \tag{34}$$

In fact we use at least one "optimal subgradient" corresponding to a shortest path solution to the subproblem and then any "suboptimal subgradient" corresponding to a nondominated path with negative reduced cost at the sink node. This approach is equivalent with "multiple pricing" in colum generation.

Now, since the Lagrange function is an affine function in λ for a fixed solution x_p, we have

$$L(\lambda, x_p) = z_D(\lambda_p) + \langle s_p, \lambda - \lambda_p \rangle \tag{35}$$

The algorith we now present for solving the dual problem is an extension of Kelley's and Cheney and Goldstein's cutting-plane algorithm for convex minimization. The algorithm is related to the method of Madsen (1975) in the sense that it solves the dual restricted

master subject to bounds on the dual variables and that these bounds are being adjusted automatically, depending on the ratio between the "nominal" decrease of the dual restricted master. This type of method is called a trust region method, introduced by Levenberg (1944) and Marquardt (1963) in connection with least squares calculations. To stabilize the cutting plane algorithm we force the next iterate to be a priori in a box centered at the current point which we refer to as the stability center. Instability refers to if the current iterate is closer (with respect to some norm) to the solution than the next iterate. In this way we stay within the framework of linear programming. We refer to this box as the trust-region since we assume the cutting plane approximation of the dual function is good within this region. For a recent presentation on ways to stabilize the cutting-plane algorithm see Hiriart-Urruty and Lemaréchal (1993) chapter [XV]. The algorithm presented in this paper is an extension of Algorithm 2.1.1 in Hiriart-Urruty and Lemaréchal (1993) chapter [XV] in the sense that we update the trust-region size so that $\Delta \to 0$ when $u \to |P|$. We also use several subgradients in each iteration to speed up convergence.

The algorithm is given the initial point λ_1 (we have choosen $\underline{0}$) and the trust-region size Δ (we have choosen 1). The first query point μ_1 is equal to the trial point we have choosen. Then we solve the corresponding SPPTWCC for the initial trial point and calculate subgradients s_p for $p \in P_E(\mu_1)$ (if $\mu_1 = \underline{0}$, then only the empty tour p is returned by the subproblem and $z_D^p(\mu_1) = 0$ and $s_p = \underline{1}$)). The cutting plane model (36) is updated with the initial information and in step 1 we compute the next query point μ_{u+1} and the cutting plane approximation of the dual function value $\hat{z}_D^u(\mu_{u+1})$. In step 2 we check the stopping criterion. In step 3 we compute the actual dual function value $z_D(\mu_{u+1})$ and calculate the corresponding subgradients at the new query point. In step 4 we compare the decrease in the dual function value with the decrease in the cutting plane approximation; the decrease predicted by our cutting plane model. The approximation ratio is used as a measure of how good the cutting plane approximation is, and we increase or decrease the trust-region size according to this ratio.

Algorithm 1 (Cutting-Plane Algorithm with Trust-Region) Choose an initial point λ_1, a stopping tolerance $\delta \geqslant 0$ and a trust-region size $\Delta > 0$. Initialize the iteration-counter $u = 1$ and $\mu_1 = \lambda_1$; compute $z_D^p(\mu_1)$ and s_p, for all $p \in P_E(\mu_1)$.

STEP 1 (Master problem). Solve the following relaxation of the dual problem

$$\begin{aligned} &\hat{z}_D^u(\mu) = \text{maximize}\, \theta \\ &\text{subject to} \\ &\theta \leqslant z_D^p(\mu_q) + \langle s_p, \mu - \mu_q \rangle \quad \text{for } q = 1, \ldots, u \quad p = 1, \ldots, |P_E(\mu_q)| \\ &\mu \leqslant \lambda_u + \Delta \\ &\mu \geqslant \lambda_u - \Delta \end{aligned} \tag{36}$$

to get a solution μ_{u+1} and compute

$$\delta_u = \hat{z}_D^u(\mu_{u+1}) - z_D^1(\lambda_u)$$

STEP 2 (Stopping criterion). If $\delta_u \leqslant \delta$ then stop.

STEP 3 (Local problem). Find a solution to the Lagrange problem to get $z_D^p(\mu_{u+1})$ and s_p, for all $p \in P_E(\mu_{u+1})$.

STEP 4 (Update). Compute the gain ratio $\rho = (z_D^1(\mu_{u+1}) - z_D^1(\lambda_u))/\delta_u$.
If $\rho = 1$ then $\Delta = \Delta * 2$.
If $\rho < 0$ then $\Delta = \Delta/3$.
If $\rho > 0.01$ then set $\lambda_{u+1} = \mu_{u+1}$ (ascent-step). Otherwise set $\lambda_{u+1} = \lambda_u$ (null-step).
Set $u = u + 1$ and go to step 1. □

Let B denote the basis of the constraint matrix of (26) and c_{BS} the cost coefficients of the basis variables y_{BS}.

The traditional Dantzig-Wolfe column generation is as follows.

Algorithm 2 (Dantzig-Wolfe's algorithm) The Dantzig-Wolfe master problem (26) is initialized with a feasible basis. Initialize the column counter $p = n$ and compute the initial simplex multipliers $\pi = c_{BS}B^{-1}$.

STEP 1 (Subproblem). Compute a solution of SPPTWCC with respect to the modified costs $\hat{c}_{ij} = c_{ij} - \pi_i$ to obtain a candidate column y_{p+1} with reduced cost $\bar{z}_{p+1}$.

STEP 2 (Stopping criterion). If $\bar{z}_{p+1} \geqslant 0$ then stop (all variables price out correctly). Otherwise adjoin the column with negative reduced cost to the restricted master problem.

STEP 3 (Master problem). Compute a solution of (26), i.e. determine a new basis and calculate the new simplex multipliers $\pi = c_{BS}B^{-1}$ and go to step 1. □

Assume we have initialized (26) with the paths $(0, i, n+1), \forall i \in C$, and suppose $c_p = 10000$ for $p = 0, \ldots, n-1$, then $\pi_i = 10000$ for $i \in C$. Figure 2 illustrates the effect of the size of the multipliers on the computational difficulty of the SPPTWCC subproblems. In the Dantizig-Wolfe column genereration algorithm the multipliers are large, compared to the optimlal level, in the beginning of the solution process, while the multipliers in the cutting-plane algorithm are small. As would be expected the computational times of solving the subproblems in the two algorithms are almost the same when the optimal level is reached.

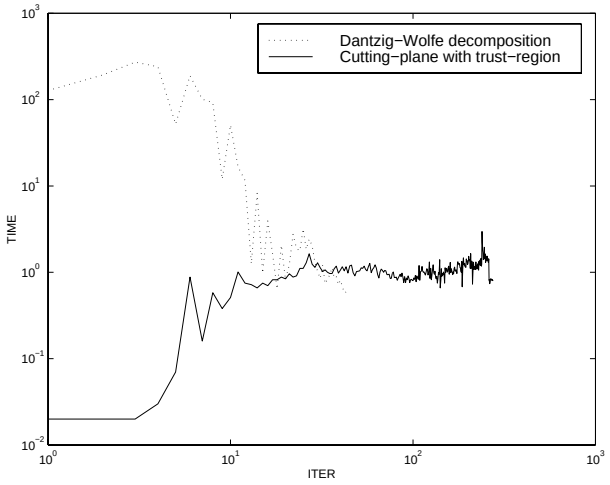

Figure 2: Development of the computational time in the SPPTWCC subproblem against the iteration number when solving the dual problem (22) for a 100 customer VRPTW instance (R104.100).

Figure 3 illustrates the instability of the column generation algorithm compared to the stabilized cutting-plane algorithm.

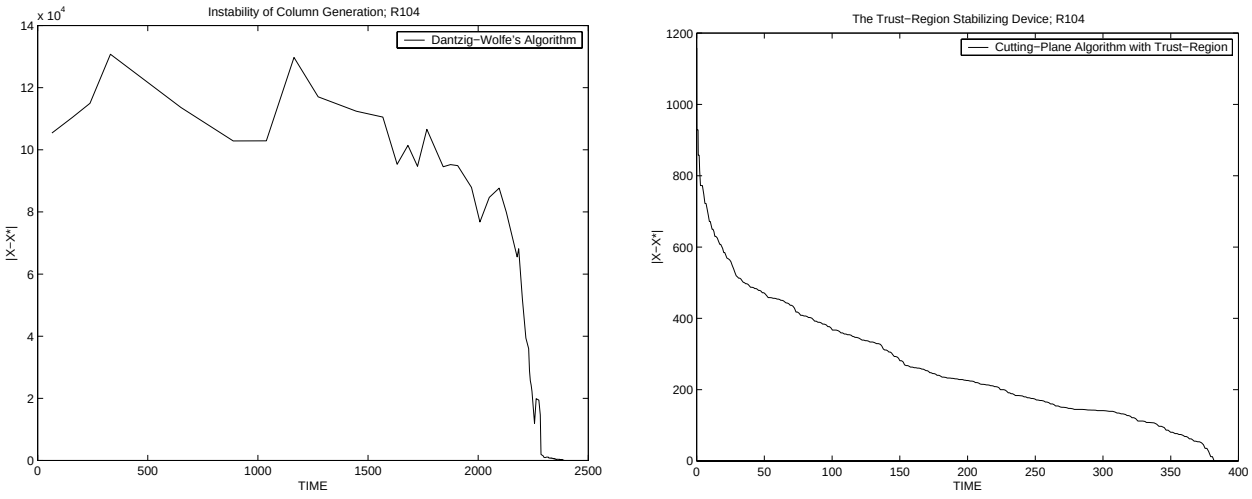

Figure 3: The Euclidian distance between the current dual variables and the optimum dual variables.

We have observed the same behaviour as illustrated in figure 2 and 3 when solving several other instances. In fact, the total computational time for solving the dual problems for the R1 instances with 100 customers was decreased by a factor 6.

		Cutting-plane		Dantzig-Wolfe	
Problem	LB_{opt}	Iterations	Seconds	Iterations	Seconds
R101	1631.15	131	17.43	22	1.08
R102	1466.60	135	83.03	41	38.31
R103	1206.31	301	227.69	50	557.97
R104	949.50	276	288.77	44	1408.69
R105	1346.14	150	40.61	23	2.94
R106	1226.44	214	106.81	41	465.41
R107	1051.84	381	295.69	44	3930.36
R108	907.16	307	434.43	48	3419.23
R109	1130.59	228	88.99	36	44.39
R110	1048.48	223	131.40	28	245.07
R111	1032.03	276	200.93	39	466.79
R112	919.19	213	261.73	34	2843.32
Total		2835	2177.51	661	13423.56

Table 1: Comparing the trust-region algorithm to Dantzig-Wolfe's algorithm on R1 instances with 100 customers.

In order to find primal solutions we have combined the trust-region algorithm with a Dantzig-Wolfe algorithm embedded in a branch and bound scheme. The Dantzig-Wolfe algorithm is developed in Larsen (1999), where it is noted that the solution of the root node is a bottle-neck in the branch-and-bound procedure and the current work can be viewed

as an effort to attack this problem. The overall solution procedure can be described as follows. First the dual problem is solved by the trust-region. If the dual optimum corresponds to a feasible primal solution then we have found the solution of the VRPTW instance. Else we add all the corresponding paths of the dual restricted master to the Dantzig-Wolfe restricted master together with the initial basis. The initial constraint matrix of the (primal) restricted master then consists of a feasible basis and all the paths found by the dual algorithm. The Dantzig-Wolfe algorithm then proceeds with step 1-3 as described above.

4 Computational results

The method was tested on the Solomon (1987) instances. They are widely regarded as **the** test set for the VRPTW.

Of the 80 Solomon type 1 problems solved to optimality 34 of the problems were solved in the root node. The average relative dual gap, $(IP_{opt} - LB_{opt})/IP_{opt}$ (where IP_{opt} denotes the optimal integer solution and LB_{opt} denotes the optimal LP-relaxed solution), for remaining 46 solved problems is 2.8%, which shows the quality of the lower bound given by our decomposition. The relative dual gap for the groups R1, C1 and RC1 are 1.2%, 0.2% respectively 5.3%, which is an indication of why there are relatively more unsolved problems in the RC1-set than in R1, and why the C1-set was the first set of instances that was solved to optimality.

Of the 46 Solomon type 2 problems solved to optimality integer solutions are found in the root node in 9 cases (among them all the C2-problems solved). The average relative gap of the remaining solved instances is 5.8% which is more than a factor 2 higher than the relative dual gap for the type 1 instances. The relative gap for the sets R2, C2 and RC2 are 2.6%, 2.9% respectively 14%. The reason why the average gap of the C2 instances is higher than for the R2 instances is that we were able to solve more "difficult" instances from the C2 set (all problems except C204.100 are solved to optimality) than in the R2 set (where we only have solved one instance with 100 customers (R201.100)). Among the type 2 instances the RC set is again the most difficult e.g. the problem RC203.25 has a dual gap of 40%.

In each section we give an overview of the solutions to the solved Solomon instances. For every problem there is given a lower LP-bound for the VRPTW found by the trust-region algorithm, the optimal primal IP value as found by the Dantzig-Wolfe algorithm (plus Branch-and-Bound), number of vehicles, number of branch-and-bound nodes and the number of valid inequalities used in the Dantzig-Wolfe algorithm. Then we give the total number of call to the SPPTWCC routine made by the trust-region algorithm and the Dantzig-Wolfe algorithm and finally the total running time in seconds on a HP J7000 computer. For the linear programs CPLEX version 6.5.1 was used. Exceptions are indicated by A, C and D. A problem marked with an 'A' is solved on an HP J2240 and with CPLEX 3.0, 'C' that the maximum number of columns returned are 20 instead of 200 and finally for the problems marked 'D' we use the method of column reduction from Larsen (1999) after every third Branch-and-Bound node.

4.1 The R1 instances

Problem	LB_{opt}	IP_{opt}	veh	no	VI	iter	Running time
R101.25	617.100	617.100	8	1	0	44	0.1
R101.50	1043.367	1044.000	12	1	2	73	0.7
R101.100A	1631.150	1637.700	20	15	9	229	35.6
R102.25	546.333	547.100	7	1	3	61	0.3
R102.50	909.000	909.000	11	1	0	135	3.5
R102.100	1466.600	1466.600	18	1	0	233	87.0
R103.25	454.600	454.600	5	1	0	58	0.4
R103.50	765.950	772.900	9	39	4	267	16.2
R103.100	1206.313	1208.700	14	49	2	662	418.9
R104.25	416.900	416.900	4	1	0	70	0.9
R104.50	616.500	625.400	6	173	4	690	302.4
R104.100							
R105.25	530.500	530.500	6	1	0	44	0.1
R105.50	892.120	899.300	9	15	13	148	2.5
R105.100	1346.142	1355.300	15	87	24	519	115.2
R106.25	457.300	465.400	5	1	12	78	0.5
R106.50	791.367	793.000	8	1	7	128	4.5
R106.100	1226.440	1234.600	13	1399	10	3043	3452.1
R107.25	422.925	424.300	4	3	3	73	0.7
R107.50	704.438	711.100	7	55	1	354	28.9
R107.100	1051.842	1064.600	11	3317	9	7467	38772.2
R108.25	396.139	397.300	4	3	2	111	3.1
R108.50C	588.926	617.700	6	7213	17	16860	67361.9
R108.100							
R109.25	441.300	441.300	5	1	0	26	0.1
R109.50	775.096	786.800	8	157	7	483	21.6
R109.100A	1130.587	1146.900	13	4005	29	8527	53100.3
R110.25	437.300	444.100	5	27	0	125	1.3
R110.50	692.577	697.000	7	5	3	117	5.7
R110.100CD	1048.482	1068.000	12	108024	8	1210273	664238.4
R111.25	423.788	428.800	4	5	3	82	0.7
R111.50	691.812	707.200	7	405	10	1032	189.4
R111.100CD	1032.028	1048.700	12	5945	5	66786	27328.5
R112.25	384.200	393.000	4	9	18	120	4.4
R112.50	607.219	630.200	6	9383	5	16092	39568.5
R112.100							

Table 2: Solution overview for the R1 instances.

Figure 4 shows the solution to R110.100. Arcs to and from the depot are not drawn. The box in the center represents the depot. Figure 5 is also the solution of R110.100 but depicted with respect to time. As can be seen R110.100 has relatively wide time windows and partly a common strip through the time windows, which is an important part of the explanation for the very long running time. One of the three remaining unsolved problems in the R1 set (R112.100) has to an even higher degree this band-structure in the time

windows. The two other unsolved instances R104.100 and R108.100 also have very wide time windows (only 25% of the customers have restrictive time windows).

In figure 5 we have depicted the solution to R110 with 100 customers with respect to time. Time in minutes is shown along the x-axis. On the y-axis the customer numbers are given. The horizontal line segments indicate the time windows, and an open circle on the start of a line segment indicates that the time window has been reduced. The sloping links indicates driving.

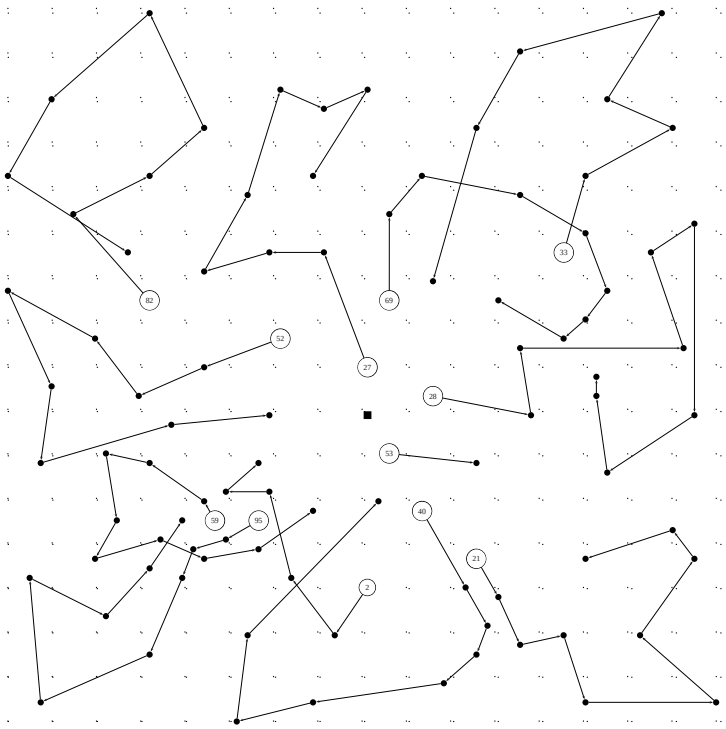

Figure 4: Geographic view of the solution to R110 with 100 customers. The cirkled numbers indicate the first customer on each route.

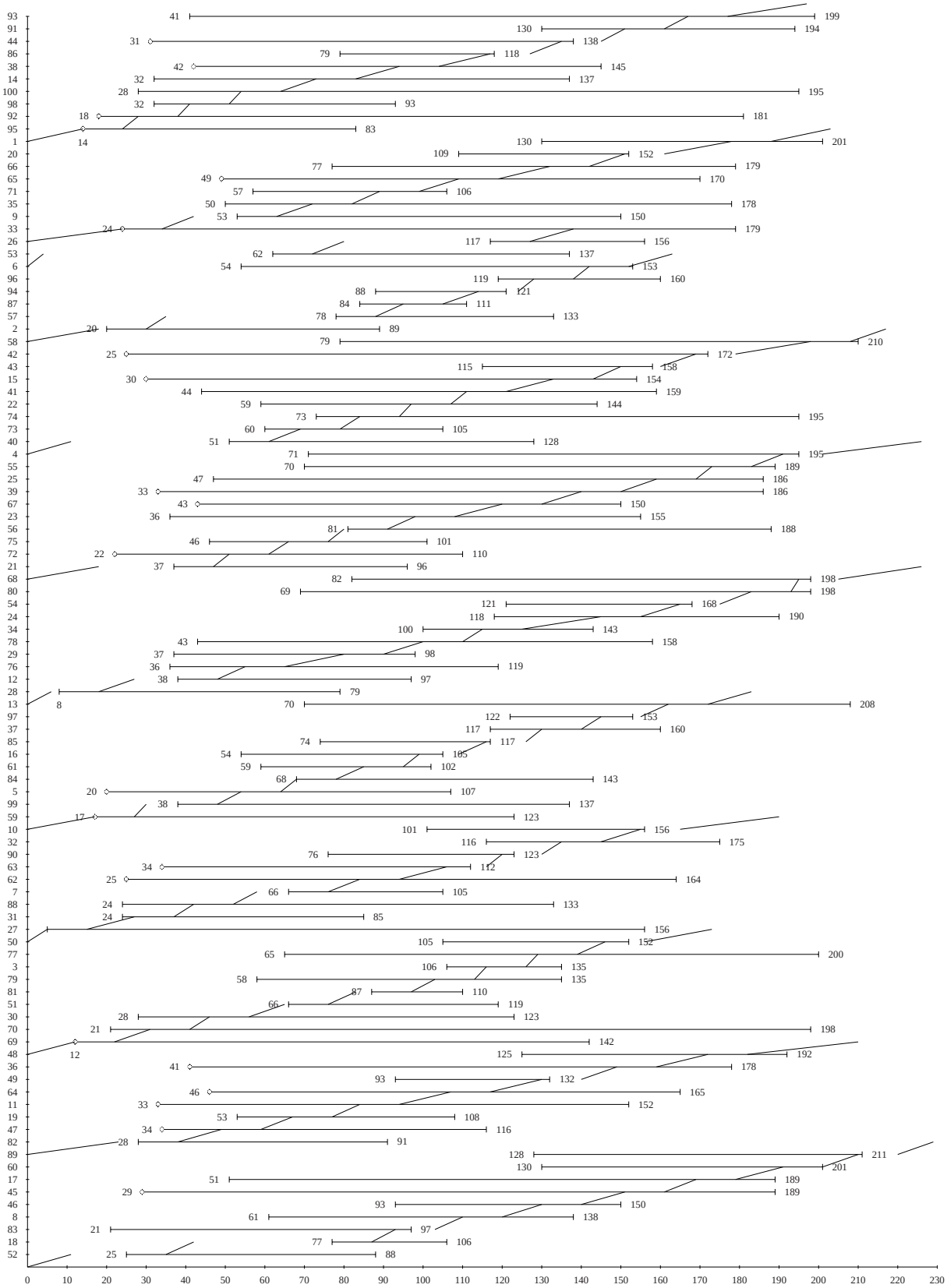

Figure 5: The solution to R110 with 100 customers depicted with respect to time.

4.2 C1 instances

These problems are known to be the easiest to solve, but we have been able to reduce the running times for instances with wide time windows (C103 and C104). Due to the clustering of customers and the construction of the time windows most of the solutions are obvious and could quickly be constructed by hand; one only has to use one vehicle for each cluster.

The solution you would find manually would be identical or very close to the optimal solution for C101.100, C102.100, C105.100, C106.100, C107.100 and C108.100, see Kallehauge (2000b) pp.185ff. for a graphical view of the solutions.

Problem	LB_{opt}	IP_{opt}	veh	no	VI	iter	Running time
C101.25	191.300	191.300	3	1	0	30	0.1
C101.50	362.400	362.400	5	1	0	52	0.5
C101.100	827.300	827.300	10	1	0	113	6.6
C102.25	190.300	190.300	3	1	0	59	0.6
C102.50	361.400	361.400	5	1	0	54	1.5
C102.100	827.300	827.300	10	1	0	134	27.8
C103.25	190.300	190.300	3	1	0	61	1.7
C103.50	361.400	361.400	5	1	0	101	12.4
C103.100	826.300	826.300	10	1	0	202	135.8
C104.25	186.900	186.900	3	1	0	72	3.2
C104.50	357.250	358.000	5	1	2	162	63.8
C104.100	822.900	822.900	10	1	0	230	362.9
C105.25	191.300	191.300	3	1	0	34	0.1
C105.50	362.400	362.400	5	1	0	56	0.9
C105.100	827.300	827.300	10	1	0	91	10.6
C106.25	191.300	191.300	3	1	0	30	0.1
C106.50	362.400	362.400	5	1	0	50	0.5
C106.100	827.300	827.300	10	1	0	100	13.2
C107.25	191.300	191.300	3	1	0	41	0.2
C107.50	362.400	362.400	5	1	0	53	0.7
C107.100	827.300	827.300	10	1	0	101	16.7
C108.25	191.300	191.300	3	1	0	45	0.2
C108.50	362.400	362.400	5	1	0	56	1.1
C108.100	827.300	827.300	10	1	0	160	23.5
C109.25	191.300	191.300	3	1	0	59	0.6
C109.50	362.400	362.400	5	1	0	73	2.5
C109.100	825.640	827.300	10	1	3	161	30.5

Table 3: Solution overview for the C1 instances.

4.3 The RC1 instances

Problem	LB_{opt}	IP_{opt}	veh	no	VI	iter	Running time
RC101.25	406.625	461.100	4	11	3	82	0.3
RC101.50	850.021	944.000	8	3	34	119	1.7
RC101.100	1584.094	1619.800	15	11	64	281	43.7
RC102.25	351.800	351.800	3	1	0	56	0.6
RC102.50	719.902	822.500	7	1685	6	3281	1029.8
RC102.100CD	1403.646	1457.500	14	15356	38	209876	27821.0
RC103.25	332.050	332.800	3	3	0	87	1.7
RC103.50	643.133	710.900	6	5	3	158	17.1
RC103.100CD	1218.495	1258.200	11	16455	39	251098	80650.6
RC104.25	305.825	306.600	3	7	0	110	2.6
RC104.50	543.750	545.800	5	27	0	254	103.0
RC104.100							
RC105.25	410.950	411.300	4	3	0	94	0.9
RC105.50	754.443	855.300	8	157	5	595	35.0
RC105.100	1471.160	1513.700	15	43	33	444	122.3
RC106.25	342.829	345.500	3	15	1	119	61.5
RC106.50	664.433	723.200	6	21	10	195	8.6
RC106.100							
RC107.25	298.300	298.300	3	1	0	53	1.5
RC107.50	591.477	642.700	6	79	2	454	77.6
RC107.100							
RC108.25	293.791	294.500	3	1	4	60	35.5
RC108.50	538.957	598.100	6	9	2	166	80.6
RC108.100							

Table 4: Solution overview for the RC1 instances.

Only four 100 customer instances remain unsolved within the RC1 instances. These instances also have very wide time windows like the unsolved instances of the R1 instances.

4.4 The R2 instances

We have solved all 25 customer instances to optimality except R204.25. Furthermore we have solved 3 50 customer instances and 1 instance with 100 customers, namely R201.100, which is shown geographically in figure 6 and time-wise in figure 7. It should be noted that the routes in figure 7 tend to break up in several parts, e.g. the route where customer 27 is the first to be serviced. The total service time for R201.100 is 1000, driving time (accumulated sum of t_{ij}'s) is 1143.2 and the waiting time is 4153.4. It could therefore be argued how realistic this solution is. This is a very good illustration of one of the problem using a purely geographic objective function for a "mixed" problem.

Problem	LB_{opt}	IP_{opt}	veh	no	VI	iter	Running time
R201.25A	460.100	463.300	4	1	0	53	1.1
R201.50	788.425	791.900	6	3	0	180	9.2
R201.100A	1136.222	1143.200	8	265	1	1589	7969.6
R202.25	406.350	410.500	4	23	0	234	14.9
R202.50A	692.738	698.500	5	7	1	400	1707.7
R202.100							
R203.25A	379.882	391.400	3	29	2	287	258.4
R203.50							
R203.100							
R204.25							
R204.50							
R204.100							
R205.25A	381.283	393.000	3	11	6	172	18.2
R205.50	666.604	690.900	5	5282	6	86840	55507.5
R205.100							
R206.25A	363.132	374.400	3	77	6	663	988.0
R206.50							
R206.100							
R207.25	347.592	361.600	3	273	9	1612	9269.2
R207.50							
R207.100							
R208.25D	318.105	330.900	1	20	8	902	23277.7
R208.50							
R208.100							
R209.25A	353.875	370.700	2	61	6	418	1976.1
R209.50							
R209.100							
R210.25A	395.844	404.600	3	59	4	520	2421.4
R210.50							
R210.100							
R211.25D	330.140	350.900	2	424	7	4481	28002.9
R211.50							
R211.100							

Table 5: Solution overview for the R2 instances.

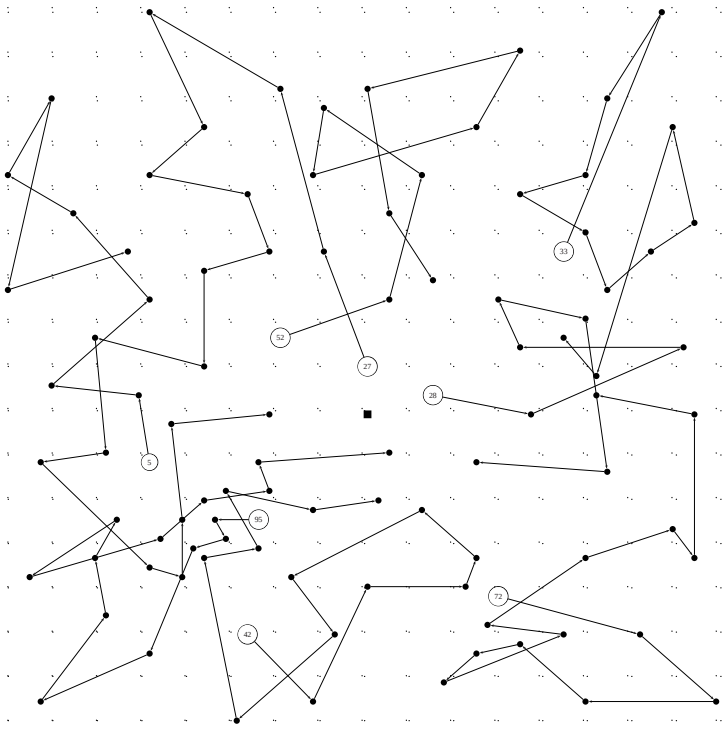

Figure 6: Geographic view of the solution to R201 with 100 customers.

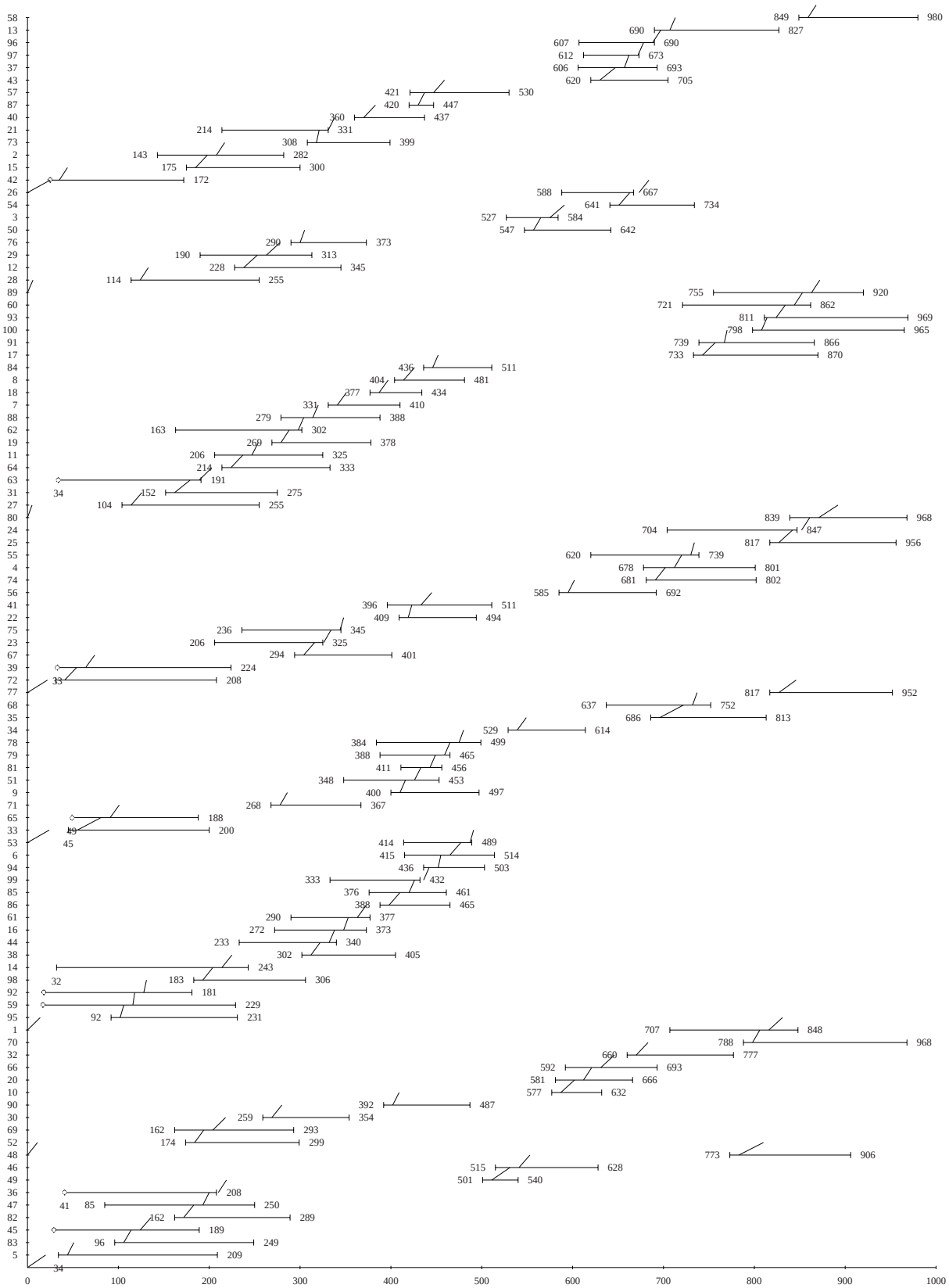

Figure 7: The solution to R201 with 100 customers depicted with respect to time.

4.5 The C2 instances

Among the C2 instances we solved all but one instance. C204.100 is the most difficult problem (due to its large time windows) and was not solved. We tried several times to solve C204.100. But we stopped either due to an upper bound of 100000 columns generated or that we ran out of memory. The problems are structually the same as the C1 instances, but longer routes are allowed, which can be seen in the solution of C208.100 in figure 8. On figure 9 we can see that the time windows are almost constructed to fit the geographical postition of the customers.

Problem	LB_{opt}	IP_{opt}	veh	no	VI	iter	Running time
C201.25A	214.700	214.700	2	1	0	36	0.2
C201.50A	360.200	360.200	3	1	0	55	1.2
C201.100A	589.100	589.100	3	1	0	60	61.8
C202.25A	214.700	214.700	2	1	0	164	7.3
C202.50A	360.200	360.200	3	1	0	177	18.8
C202.100A	589.100	589.100	3	1	0	61	37.6
C203.25A	214.700	214.700	2	1	0	191	24.7
C203.50	359.800	359.800	3	1	0	272	167.1
C203.100	588.700	588.700	3	1	0	188	859.4
C204.25	211.004	213.100	1	4	0	268	424.3
C204.50	350.100	350.100	2	1	0	368	851.3
C204.100							
C205.25A	212.050	214.700	2	2	0	116	3.4
C205.50	357.350	363.500	3	2	0	160	390.3
C205.100	586.400	586.400	3	9	0	94	31.4
C206.25A	197.700	214.700	2	1	5	198	11.5
C206.50A	344.200	359.800	3	3	4	568	1363.9
C206.100A	585.400	586.000	3	1	2	419	1976.1
C207.25	207.981	214.500	2	73	0	1066	825.8
C207.50A	356.269	359.600	3	1	10	500	925.4
C207.100A	581.969	585.800	3	1	6	664	5538.8
C208.25A	193.280	214.500	2	9	6	336	85.6
C208.50A	340.425	350.500	2	1	3	529	420.6
C208.100A	581.767	585.800	3	1	4	730	8347.3

Table 6: Solution overview for the C2 instances.

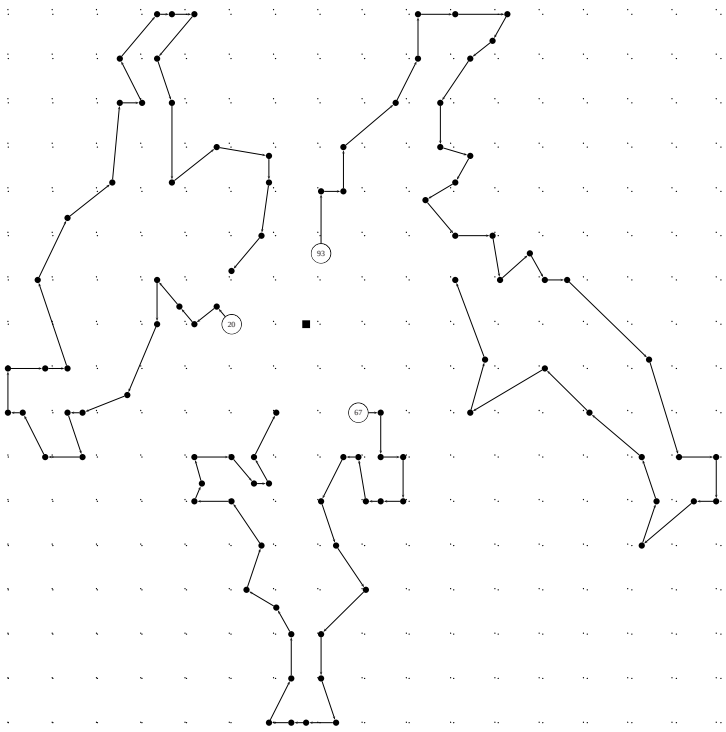

Figure 8: Geographic view of the solution to C208 with 100 customers.

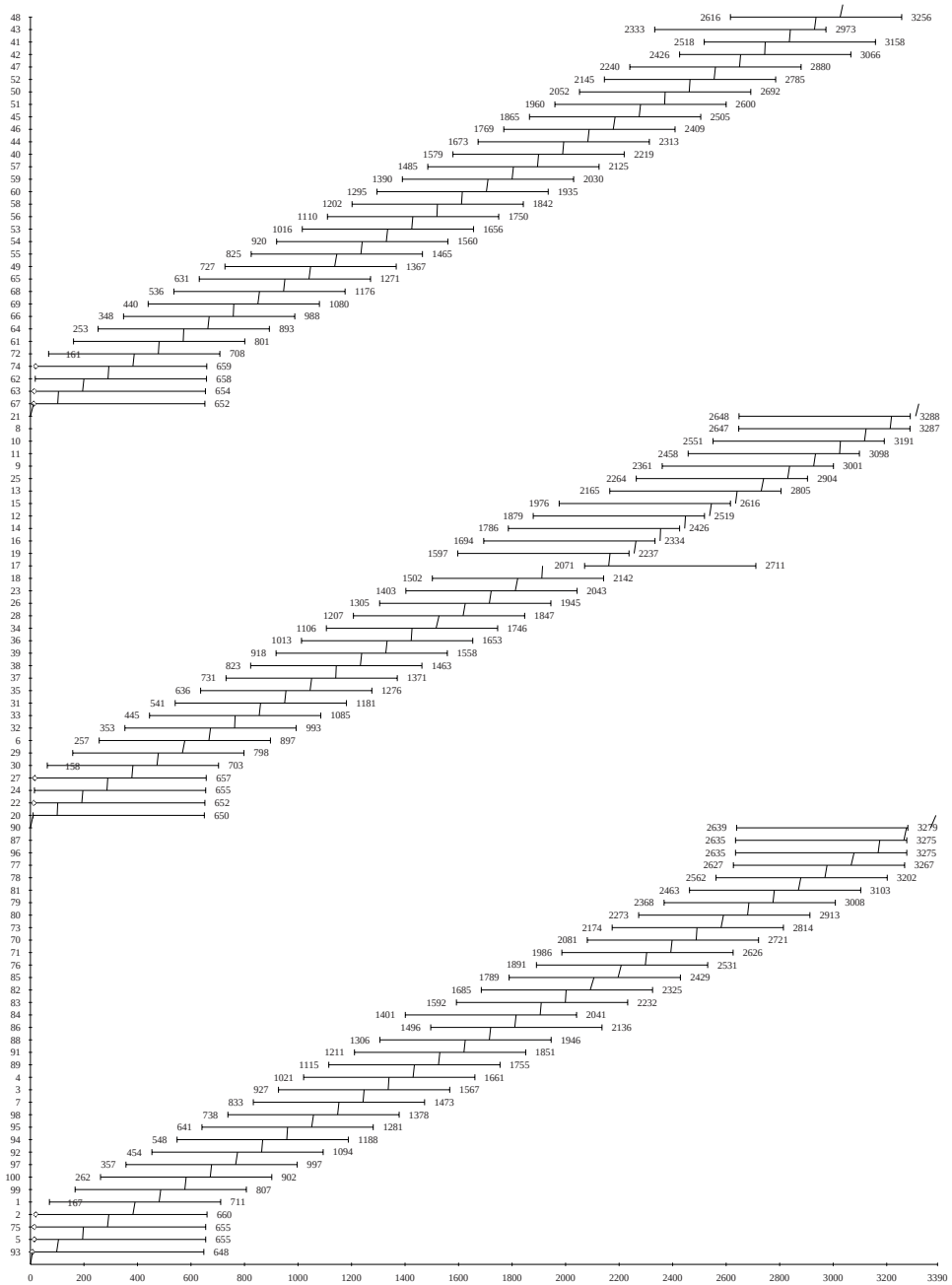

Figure 9: The solution to C208 with 100 customers depicted with respect to time.

4.6 The RC2 instances

The RC2 instance were the computationally most difficult problems to solve. Two 25 customer problems are not solved yet (RC204.25 and RC208.25) and for the remaining (solved) 25 customer instances the running time is very large compared to R2 and C2. We have succeeded in solving one 100 customer problem, namely RC201.100, which is shown in figure 10 and 11. For RC201.100 the total service time is 1000, the driving time 1261 and the waiting time 4243, that is, the same proportion as we saw in R201.100.

Problem	LB_{opt}	IP_{opt}	veh	no	VI	iter	Running time
RC201.25	356.650	360.200	3	3	0	79	0.6
RC201.50	670.150	684.800	5	31	0	401	61.8
RC201.100CD	1240.398	1261.800	9	524	10	33610	35068.7
RC202.25A	290.408	338.000	3	117	6	1207	6351.7
RC202.50							
RC202.100							
RC203.25CD	214.475	356.400	2	12399	5	383930	213306.2
RC203.50							
RC203.100							
RC204.25							
RC204.50							
RC204.100							
RC205.25	307.600	338.000	3	47	0	453	33.1
RC205.50							
RC205.100							
RC206.25	250.390	324.000	3	2465	8	27251	82384.9
RC206.50							
RC206.100							
RC207.25CD	217.965	298.300	3	13395	3	215695	221087.1
RC207.50							
RC207.100							
RC208.25							
RC208.50							
RC208.100							

Table 7: Solution overview for the RC2 instances.

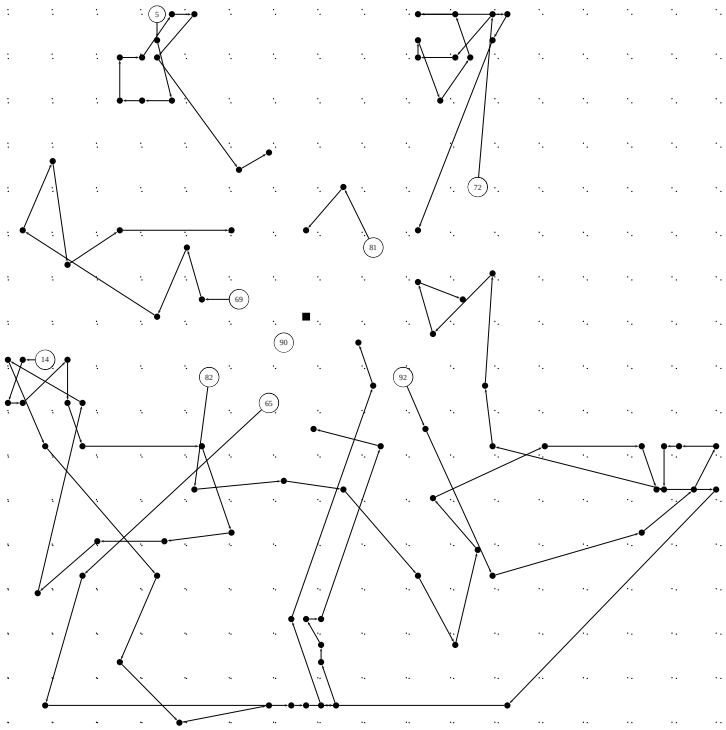

Figure 10: Geographic view of the solution to RC201 with 100 customers.

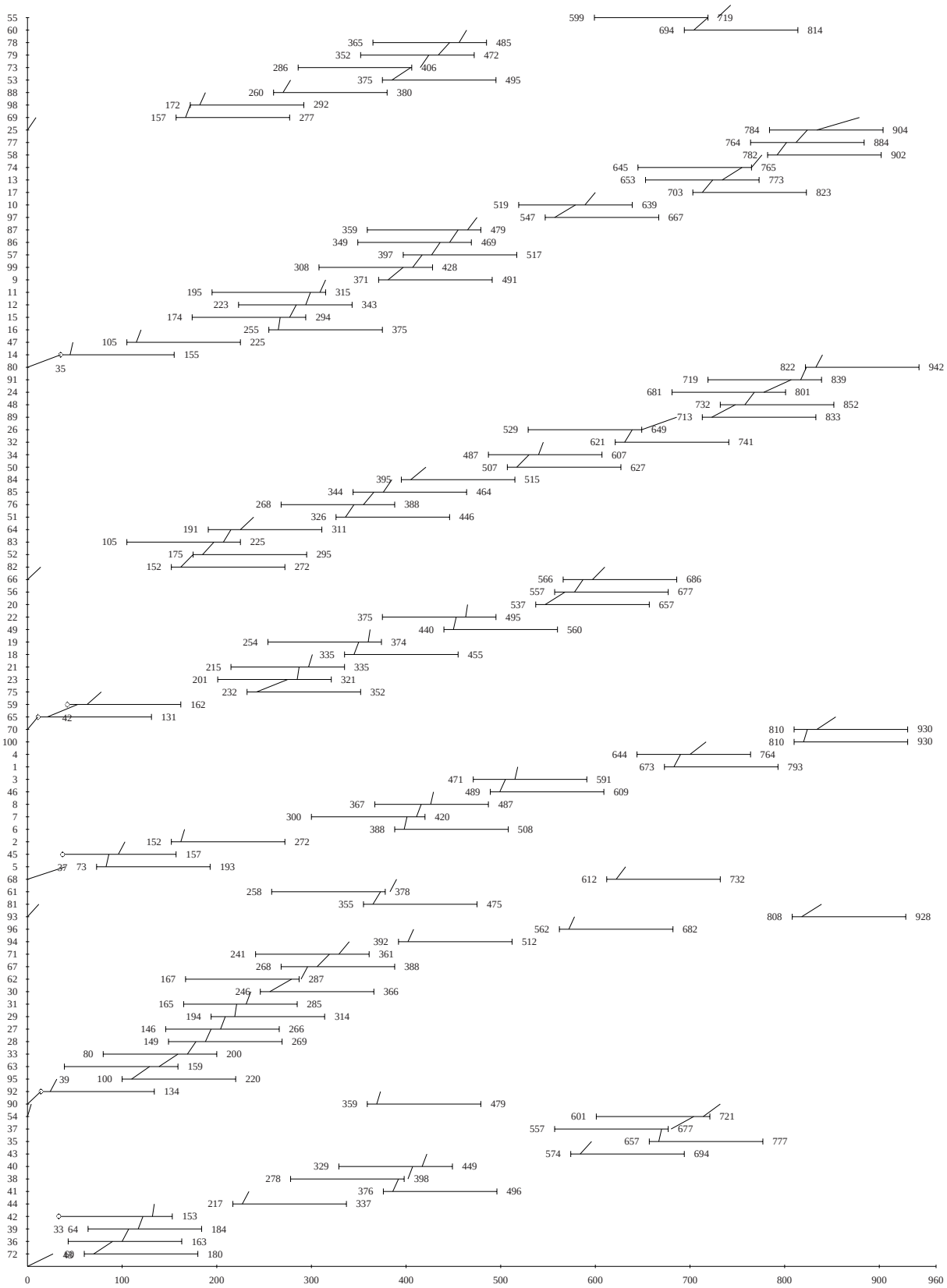

Figure 11: The solution to RC201 with 100 customers depicted with respect to time.

4.7 The Homberger instances

We have solved 9 problems from the Homberger testsets Homberger, among them problems with 400 and 1000 customers. In 8 of the problems the customers are grouped (C-instances), while we succeeded in solving a 200 customer problem where the customers are randomly located.

Problem	LB_{opt}	IP_{opt}	veh	no	VI	iter	Running time
R1_2_1.200	4654.900	4667.200	23	469	21	1549	5198.2
C1_2_1.200	2698.600	2698.600	20	1	0	251	114.6
C1_2_2.200	2682.187	2694.300	20	95	6	1121	4695.6
C1_2_5.200	2694.900	2694.900	20	1	0	235	118.3
C1_2_6.200	2694.900	2694.900	20	1	0	341	159.1
C1_2_7.200	2694.900	2694.900	20	1	0	304	179.2
C1_2_8.200	2667.870	2684.000	20	129	13	1032	2048.7
C1_4_1.400	7138.767	7138.800	40	1	1	17527	9233.2
C110_1.1000	42444.400	42444.800	100	3	2	1141	42359.1

Table 8: Overview of the solved Homberger instances.

4.8 An instance with 1000 customers

The solution to an instance with 1000 customers is shown in figure 12. The objective value is 42444.8 and the solution needed 100 vehicles. The trust-region algorithm solved the root node in 849 iterations in 17523 seconds. The Dantzig-Wolfe algorithm used 292 master iterations, returning at most 200 columns for every call of the SPPTWCC subproblem and used a total of 24836 seconds to find the optimal solution. 2 valid inequalities were introduced and we needed one branching operation (branching on arcs), so a total of 3 branch-and-bound-nodes were necessary.

```
---------- Statistics
This program ran on serv3 ().
Total execution time                      24836.17 seconds
         (Solving root 23245.11 seconds)
Time used in separation                     34.25 seconds
         Cuts generated           2
Accumulated time used in calls of SPPTWCC  870.12 seconds
Time used in largest single SPPTWCC call     9.41 seconds
Branching nodes examined 3 (Veh 0, Arc 1, TW 0)
        (hereof 0 where not feasible)
No of calls to SPPTW 292, Routes generated 53294
Max no of columns selected per SPPTW 200
No of multiple customers deleted explicitly 0
IP value 424448
RP value 424446.833
LP value 424444.000
-----------------------------------------------------
Total CPLEX optimize time 23872.30 Biggest 1000.05
Total branching time 23.49 Biggest 23.49
```

Table 9: Program output for solving C110_1.1000.

In table 9 one can see that the main part of the running time is used in the LP-solver

(23872 seconds of a total of 24836 seconds). It is characteristic for problems with more than 100 customers that the relative amount of time used in the LP-solver in many cases is larger than the time used in the shortest path routine (see Kallehauge 2000b p.82).

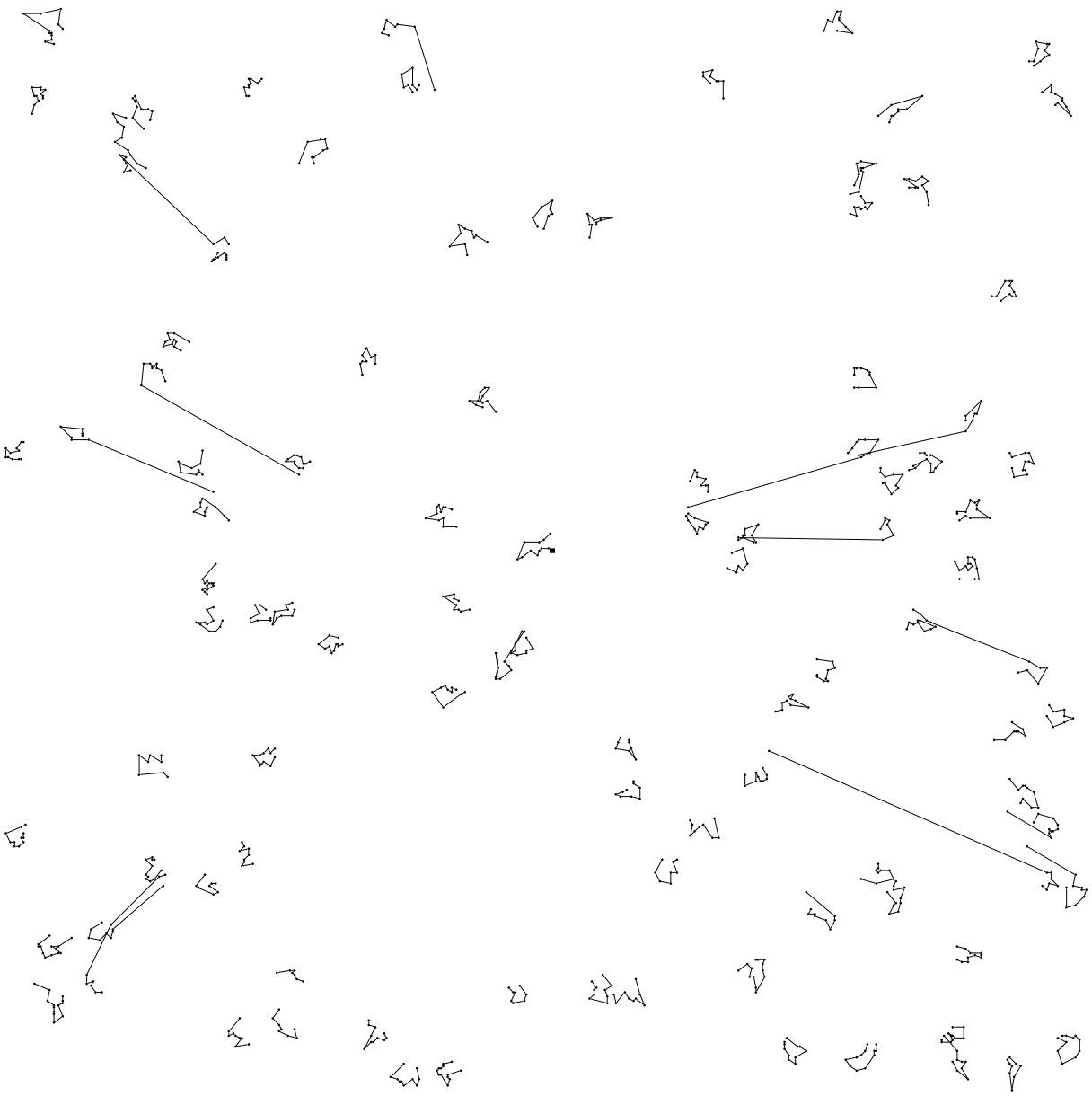

Figure 12: The solution to C110_1.1000.

5 Conclusion

The algorithm has been tested on the well-known Solomon VRPTW benchmark problems and a range of extended Solomon problems created by Homberger. We have succeeded in solving several previously unsolved Solomon problems and a Homberger problem with 1000 customers, which is the largest problem ever solved to optimality. The computational times were reduced significantly by the trust-region method in the root node of the branch-and-bound tree compared to the traditional Dantzig-Wolfe algorithm due to easier subproblems. It therefore seems very efficient to stabilize the dual variables even

with a simple and problem independent trust-region scheme.

Acknowledgement

Algorithm 1 is joint work with Kaj Madsen, Informatics and Mathematical Modelling, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark.

References

Bodin, L., B. Golden, A. Assad, and M. Ball (1983). Routing and Scheduling of Vehicles and Crews - The State of Art. *Computers & Operations Research 10*, 62 – 212.

Breedam, A. V. (1995). Vehicle Routing: Bridging the gap between theory and practice. *Belgian Journal of Operations Research, Statistics and Computer Science 35*, 63 – 80.

Cheney, E. W. and A. A. Goldstein (1959). Newton's Method for Convex Programming and Tchebycheff Approximation. *Numerische Mathematik 1*, 253–268.

Crainic, T. G. and G. Laporte (1998). *Fleet Management and Logistics*. Dordrecht, The Netherlands: Kluwer.

Desrochers, M., J. Desrosiers, and M. Solomon (1992). A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research 40*, 342 – 354.

Desrosiers, J., Y. Dumas, M. M. Solomon, and F. Soumis (1995). Time Constrained Routing and Scheduling. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser (Eds.), *Network Routing*, Volume 8 of *Handbooks in Operations Research and Management Science*, pp. 35 – 139. Amsterdam, The Netherlands: North-Holland.

du Merle, O., D. Villeneuve, J. Desrosiers, and P. Hansen (1999). Stabilized column generation. *Discrete Mathematics 194*, 229–237.

Fisher, M. (1995). Vehicle Routing. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser (Eds.), *Network Routing*, Volume 8 of *Handbooks in Operations Research and Management Science*, pp. 1 – 79. Amsterdam, The Netherlands: North-Holland.

Halse, K. (1992). *Modeling and Solving Complex Vehicle Routing Problems*. Ph. D. thesis, Department of Mathematical Modelling, Technical University of Denmark.

Hiriart-Urruty, J.-B. and C. Lemaréchal (1993). *Convex Analysis and Minimization Algorithms I-II*, Volume 305-306 of *Grundlehren der matematischen Wissenschaften*. Berlin Heidelberg: Springer-Verlag.

Homberger, J. (2000). Extended solomon's vrptw instances. Available on the web at `www.fernuni-hagen.de/WINF/touren/inhalte/probinst.htm`.

Kallehauge, B. (2000a). Lagrangean Duality and Non-differentiable Optimization - Applied on Routing with Time Windows. Master's thesis, Department of Mathematical Modelling, Technical University of Denmark. IMM-EKS-2000-13, [in Danish].

Kallehauge, B. (2000b). Solutions to the Solomon Instances for VRPTW. Supplementary report for Masters Thesis no. 13, [in Danish].

Kelley, J. E. (1960). The cutting-plane method for solving convex programs. *Journal of SIAM 8*, 703–712.

Kohl, N. (1995). *Exact methods for Time Constained Routing and Related Scheduling Problems*. Ph. D. thesis, Department of Mathematical Modelling, Technical University of Denmark. IMM-PHD-1995-16.

Kohl, N. and O. B. G. Madsen (1997). An Optimization Algorithm for the Vehicle Routing Problem with Time Windows based on Lagrangean Relaxation. *Operations Research 45*, 395 – 406.

Larsen, J. (1999). *Parallellization of the Vehicle Routing Problem with Time Windows*. Ph. D. thesis, Department of Mathematical Modelling, Technical University of Denmark. IMM-PHD-1999-62.

Levenberg, K. (1944). A method for the solution of certain problems in least squares. *Quart. Appl. Math. 2*, 164–168.

Madsen, K. (1975). An algorithm for Minimax Solution of Overdetermined Systems of Non-linear Equations. *Journal of the Institute of Mathematics and Its Applications 16*, 321–328.

Marquardt, D. (1963). An algorithm for Least Squares Estimation on Nonlinear Parameters. *SIAM J. Appl. Math. 11*, 431–441.

Neame, P., N. Boland, and D. Ralph (2000). An outer approximate subdifferential method for piecewise affine optimization. *Mathematical Programming 87*, 57 – 86.

Neame, P. J. (1999). *Nonsmooth Dual Methods in Integer Programming*. Ph. D. thesis, Department of Mathematics and Statistics, The University of Melbourne.

Nemhauser, G. L. and L. A. Wolsey (1988). *Integer and Combinatorial Optimization*. New York: Wiley.

Solomon, M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research 35*, 254 – 265.

The Danish Ministry of Transport (1998, January). Trafikredegørelse 1997. [in danish].