IMM
DEPARTMENT OF MATHEMATICAL MODELLING

Technical University of Denmark
DK-2800 Lyngby – Denmark

J. No.
2000-03-04
jla

LAGRANGEAN DUALITY APPLIED ON VEHICLE ROUTING WITH TIME WINDOWS

EXPERIMENTAL RESULTS

Brian Kallehauge
Jesper Larsen
Oli B. G. Madsen

TECHNICAL REPORT

IMM-REP-2000-8

IMM

LAGRANGEAN DUALITY APPLIED ON VEHICLE ROUTING WITH TIME WINDOWS

EXPERIMENTAL RESULTS

Brian Kallehauge
Jesper Larsen
Oli B. G. Madsen

TECHNICAL REPORT

IMM-REP-2000-8

IMM

Abstract

This report presents the results of the application of a non-differentiable optimization methods 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. The optimal multipliers are found using the non-differentiable method denoted the proximal bundle method.

The bundle-method has been coupled with a Dantzig-Wolfe algorithm in a branch-and-bound scheme. The root node of the branch-and-bound tree is solved by the bundle-method and, if an integer solution is not obtained, shifting to a Dantzig-Wolfe algorithm in the tree nodes. The combined bundle- and Dantzig-Wolfe algorithm has been tested on the well-known Solomon VRPTW benchmark problems and a range of extended Solomon problems.

Since we have succeded in solving 14 previously unsolved problems and an extended Solomon problem with 1000 customers, which is the largest problem ever solved to optimality, and since the computational times were reduced significantly by the bundle method in the root node compared to the Dantzig-Wolfe method it seems very efficient to combine the use of a bundle-method with a Dantzig-Wolfe algorithm, thereby combining the strengths of an Lagrangian relaxation approach with the strengths of an Dantzig-Wolfe decomposition approach for the VRPTW.

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

1 Introduction

In the real world 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. in [BGAB83] 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 reports in [Hal92] 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 writes in [Fis97] 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 [The98].

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 visit 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 in section [Lar99].

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 is 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 [DDSS93, Bre95, CL98].

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 transportationscosts c_{ij} without changing the optimal solution to VRPTW.

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

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

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:

$$\min \sum_{k \in \mathcal{V}} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} c_{ij} x_{ijk} \quad s.t. \tag{1}$$

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

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

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

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

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

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

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

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

The 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 a 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:

$$\begin{aligned}
&\min \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \hat{c}_{ij} x_{ij}, && s.t. && (10)\\
&\sum_{i \in \mathcal{C}} d_i \sum_{j \in \mathcal{N}} x_{ij} &\leq\ & q && (11)\\
&\sum_{j \in \mathcal{N}} x_{0j} &=\ & 1 && (12)\\
&\sum_{i \in \mathcal{N}} x_{ih} - \sum_{j \in \mathcal{N}} x_{hj} &=\ & 0 \quad \forall h \in \mathcal{C} && (13)\\
&\sum_{i \in \mathcal{N}} x_{i,n+1} &=\ & 1 && (14)\\
&s_i + t_{ij} - K(1 - x_{ij}) &\leq\ & s_j \forall i, j \in \mathcal{N} && (15)\\
&a_i \leq s_i \leq b_i && & \forall i \in \mathcal{N} && (16)\\
&x_{ij} \in \{0, 1\} && & \forall i, j \in \mathcal{N} && (17)
\end{aligned}$$

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 $\hat{c}_{ij} = c_{ij} - \pi_i$. 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 (sometimes denoted the Elementary Shortest Path Problem with Time Windows and Capacity Constraints (ESPPTWCC)) there is no known efficient algorithm, making the problem unsolvable for practical purposes. Therefore some of these 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 not processed 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. $c(i, t_1, d_1) \leq c(i, t_2, d_2)$.

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

**repeat**
    $(i,t,d) = \text{BestLabel}(NPS)$

    **for** $j := 1$ **to** $n+1$ **do**
        **if** $(i \neq j \wedge t + t_{ij} \leq b_j \wedge d + d_j \leq q)$ **then**
            ⟨ Label feasible ⟩
            **if** $c(j, \max\{t+t_{ij}, a_j\}, d+d_j) > c(i,t,d) + \hat{c}_{ij}$ **then**
                ⟨ New label better ⟩
                $\text{InsertLabel}(NPS,\ (j, \max\{t+t_{ij}, a_j\}, d+d_j))$
                $c(j, \max\{t+t_{ij}, a_j\}, d+d_j) = c(i,t,d) + \hat{c}_{ij}$

**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.

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.

3 A Hybrid Solution Method

OLI SKRIVER HER.

4 Solutions

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 $(\frac{IP_{opt} - LB_{opt}}{IP_{opt}})$ for remaining 46 solved problems is 2.8%, which shows the quality of the lower bound given by our shortest path 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 instancesis 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 the next section we give a overview of the solutions to the solved Solomon instances. Fro every problem there is given a lower bound for the VRPTW found by the proximal bundle-method, the optimal primal IP value as found by the Dantzig-Wolfe algorithm, 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 proximal bundle-method and the Dantzig-Wolfe algorithm and finaly the total running time.

4.1 The R1 instances

Problem	LB$_{opt}$	IP$_{opt}$	veh	no	VI	iter	Beregningstid
R101.25	617,100	617,100	8	1	0	33	1,0
R101.50	1043,367	1044,000	12	1	2	77	1,1
R101.100A	1631,150	1637,700	20	15	9	235	17,1
R102.25	546,333	547,100	7	1	3	57	1,0
R102.50	909,000	909,000	11	1	0	89	1,1
R102.100	1466,600	1466,600	18	1	0	208	18,1
R103.25	454,600	454,600	5	1	0	55	1,0
R103.50	765,950	772,900	9	39	4	249	13,2
R103.100	1206,313	1208,700	14	49	2	504	161,4
R104.25	416,900	416,900	4	1	0	76	1,0
R104.50	616,500	625,400	6	173	4	701	288,5
R104.100							
R105.25	530,500	530,500	6	1	0	40	1,0
R105.50	892,120	899,300	9	15	13	155	2,3
R105.100	1346,142	1355,300	15	87	24	458	62,7
R106.25	457,300	465,400	5	1	12	55	1,2
R106.50	791,367	793,000	8	1	7	128	2,4
R106.100	1226,440	1234,600	13	1399	10	2991	3288,0
R107.25	422,925	424,300	4	3	3	87	1,2
R107.50	704,438	711,100	7	55	1	310	21,6
R107.100	1051,842	1064,600	11	3317	9	7278	38384,8
R108.25	396,139	397,300	4	3	2	111	2,9
R108.50C	588,926	617,700	6	7213	17	16840	67344,2
R108.100							
R109.25	441,300	441,300	5	1	0	34	0,0
R109.50	775,096	786,800	8	157	7	471	20,8
R109.100A	1130,587	1146,900	13	4005	29	8504	53009,9
R110.25	437,300	444,100	5	27	0	140	1,4
R110.50	692,577	697,000	7	5	3	140	4,3
R110.100CD	1048,482	1068,000	12	108024	8	1210219	664037,5
R111.25	423,788	428,800	4	5	3	72	1,4
R111.50	691,812	707,200	7	405	10	1002	184,6
R111.100CD	1032,028	1048,700	12	5945	5	66740	27097,7
R112.25	384,200	393,000	4	9	18	102	4,6
R112.50	607,219	630,200	6	9383	5	16102	39561,8
R112.100							

Table 1: Solution overview for the R1 instances.

Figure 2 shows the solution to R110.100. Arcs to and from the depot are not drawn. The box in the center represents the depot. Figure 3 is also the solution of R110.100 but depicted with respect to time. As can be seen R110.100 has relatively wide time windoes 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 restictive time windows).

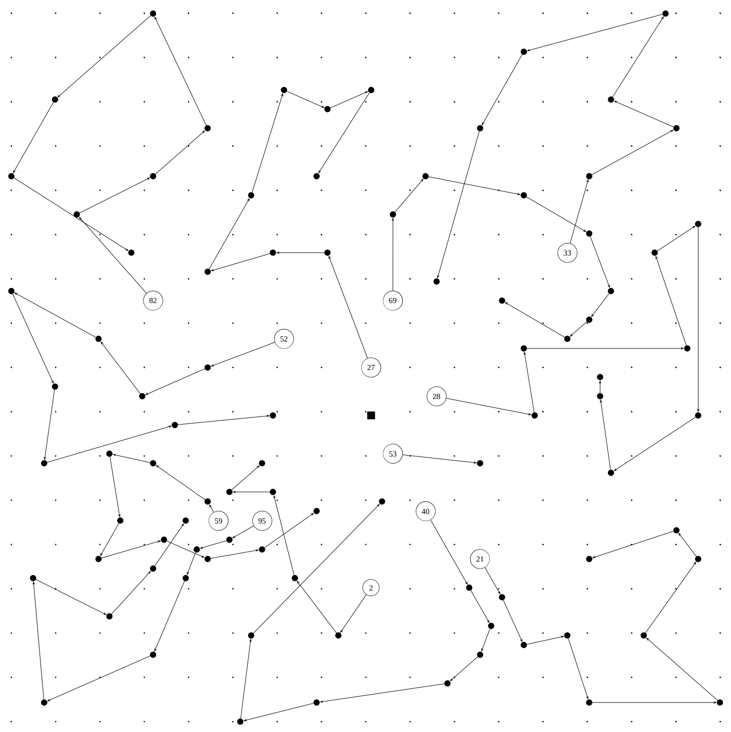

Figure 2: Geographic view of the solution to R110 with 100 customers.

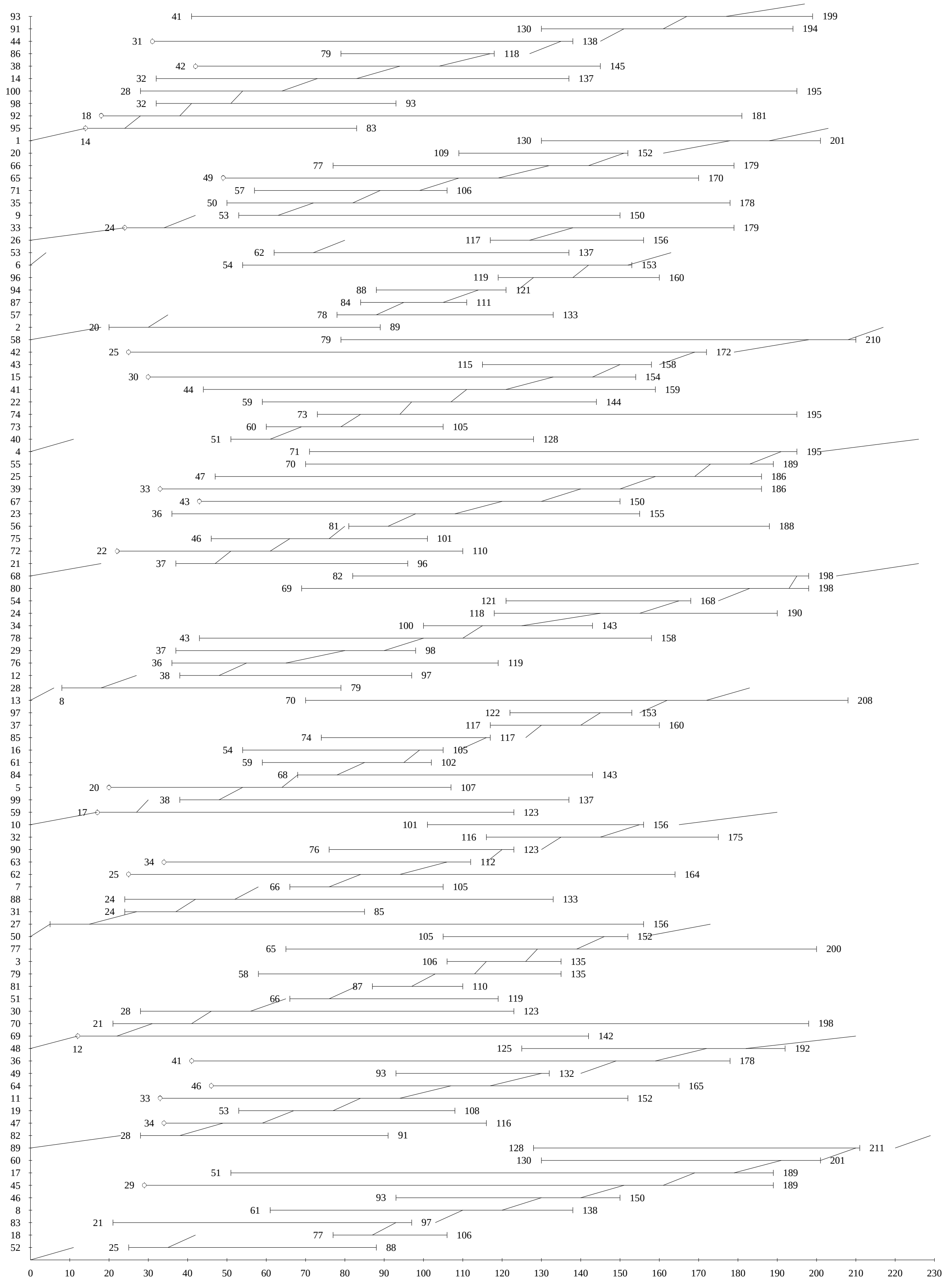

Figure 3: Tidsmæssig view of the solution to R110 with 100 customers.

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 [Kal] pp.185ff. for a graphical view of the solutions.

Problem	LB_{opt}	IP_{opt}	veh	no	VI	iter	Beregningstid
C101.25	191,300	191,300	3	1	0	22	1,0
C101.50	362,400	362,400	5	1	0	40	0,3
C101.100	827,300	827,300	10	1	0	77	4,4
C102.25	190,300	190,300	3	1	0	50	0,2
C102.50	361,400	361,400	5	1	0	47	0,5
C102.100	827,300	827,300	10	1	0	106	19,4
C103.25	190,300	190,300	3	1	0	58	2,1
C103.50	361,400	361,400	5	1	0	81	8,6
C103.100	826,300	826,300	10	1	0	145	110,7
C104.25	186,900	186,900	3	1	0	62	2,2
C104.50	357,250	358,000	5	1	2	174	42,9
C104.100	822,900	822,900	10	1	0	204	220,4
C105.25	191,300	191,300	3	1	0	28	0,1
C105.50	362,400	362,400	5	1	0	49	0,7
C105.100	827,300	827,300	10	1	0	81	9,9
C106.25	191,300	191,300	3	1	0	23	1,0
C106.50	362,400	362,400	5	1	0	40	0,3
C106.100	827,300	827,300	10	1	0	89	11,9
C107.25	191,300	191,300	3	1	0	27	0,1
C107.50	362,400	362,400	5	1	0	39	0,4
C107.100	827,300	827,300	10	1	0	89	16,0
C108.25	191,300	191,300	3	1	0	32	0,1
C108.50	362,400	362,400	5	1	0	43	1,6
C108.100	827,300	827,300	10	1	0	92	15,2
C109.25	191,300	191,300	3	1	0	61	0,1
C109.50	362,400	362,400	5	1	0	71	2,3
C109.100	825,640	827,300	10	1	3	152	18,5

Table 2: Solution overview for the C1 instances.

4.3 The RC1 instances

Problem	LB$_{opt}$	IP$_{opt}$	veh	no	VI	iter	Beregningstid
RC101.25	406,625	461,100	4	11	3	73	1,2
RC101.50	850,021	944,000	8	3	34	118	1,8
RC101.100	1584,094	1619,800	15	11	64	238	18,8
RC102.25	351,800	351,800	3	1	0	34	1,1
RC102.50	719,902	822,500	7	1685	6	3270	1027,7
RC102.100CD	1403,646	1457,500	14	15356	38	209820	27750,3
RC103.25	332,050	332,800	3	3	0	90	1,2
RC103.50	643,133	710,900	6	5	3	134	8,2
RC103.100CD	1218,495	1258,200	11	16455	39	251053	80419,2
RC104.25	305,825	306,600	3	7	0	117	1,9
RC104.50	543,750	545,800	5	27	0	245	87,4
RC104.100							
RC105.25	410,950	411,300	4	3	0	74	1,1
RC105.50	754,443	855,300	8	157	5	587	33,3
RC105.100	1471,160	1513,700	15	43	33	392	64,7
RC106.25	342,829	345,500	3	15	1	115	61,0
RC106.50	664,433	723,200	6	21	10	205	8,6
RC106.100							
RC107.25	298,300	298,300	3	1	0	49	1,1
RC107.50	591,476	642,700	6	79	2	430	64,1
RC107.100							
RC108.25	293,791	294,500	3	1	4	65	34,0
RC108.50	538,957	598,100	6	9	2	148	61,1
RC108.100							

Table 3: Solution overview for the RC1 instances.

4.4 The R2 instances

We have solved all 25 customer instances to optimality except R204.25. Furthermore we have solve 3 50 customer instances and 1 instace with 100 customers, namely R201.100, which is shown geographically in figure 4 and time-wise in figure 5. It should be noted that the routes in figure 5 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	Beregningstid
R201.25A	460,100	463,300	4	1	0	453	1,9
R201.50	788,425	791,900	6	3	0	605	12,3
R201.100A	1136,222	1143,200	8	265	1	2124	7570,1
R202.25	406,350	410,500	4	23	0	894	18,8
R202.50A	692,737	698,500	5	7	1	881	1696,1
R202.100							
R203.25A	379,882	391,400	3	29	2	1892	355,4
R203.50							
R203.100							
R204.25							
R204.50							
R204.100							
R205.25A	381,283	393,000	3	11	6	911	21,4
R205.50	666,604	690,900	5	5282	6	87849	55507,0
R205.100							
R206.25A	363,132	374,400	3	77	6	1577	988,7
R206.50							
R206.100							
R207.25	347,592	361,600	3	273	9	2687	9296,8
R207.50							
R207.100							
R208.25D	318,105	330,900	1	20	8	1853	23370,7
R208.50							
R208.100							
R209.25A	353,875	370,700	2	61	6	1821	1991,4
R209.50							
R209.100							
R210.25A	395,844	404,600	3	59	4	545	2417,7
R210.50							
R210.100							
R211.25D	330,140	350,900	2	424	7	4558	27998,8
R211.50							
R211.100							

Table 4: Solution overview for the R2 instances.

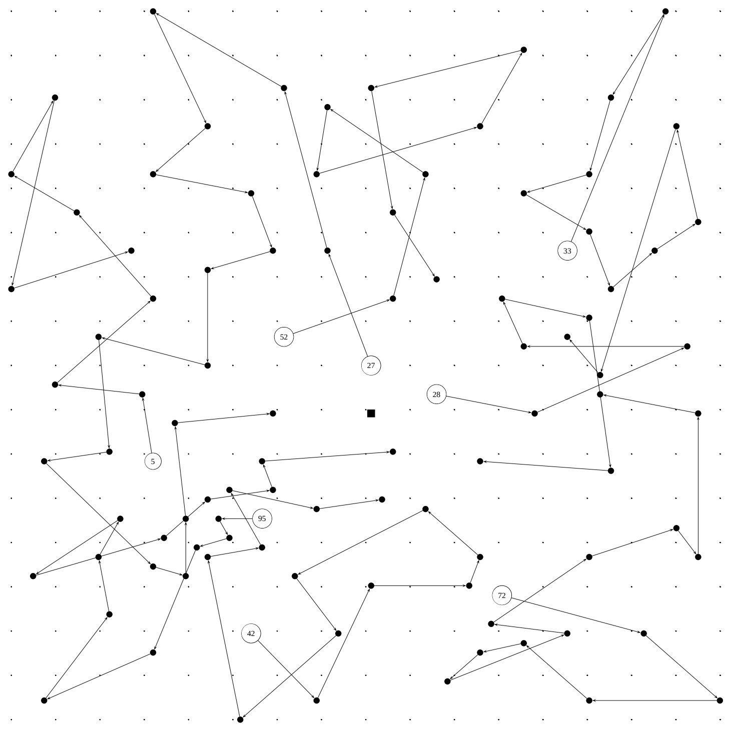

Figure 4: Geographic view of the solution to R201 with 100 customers.

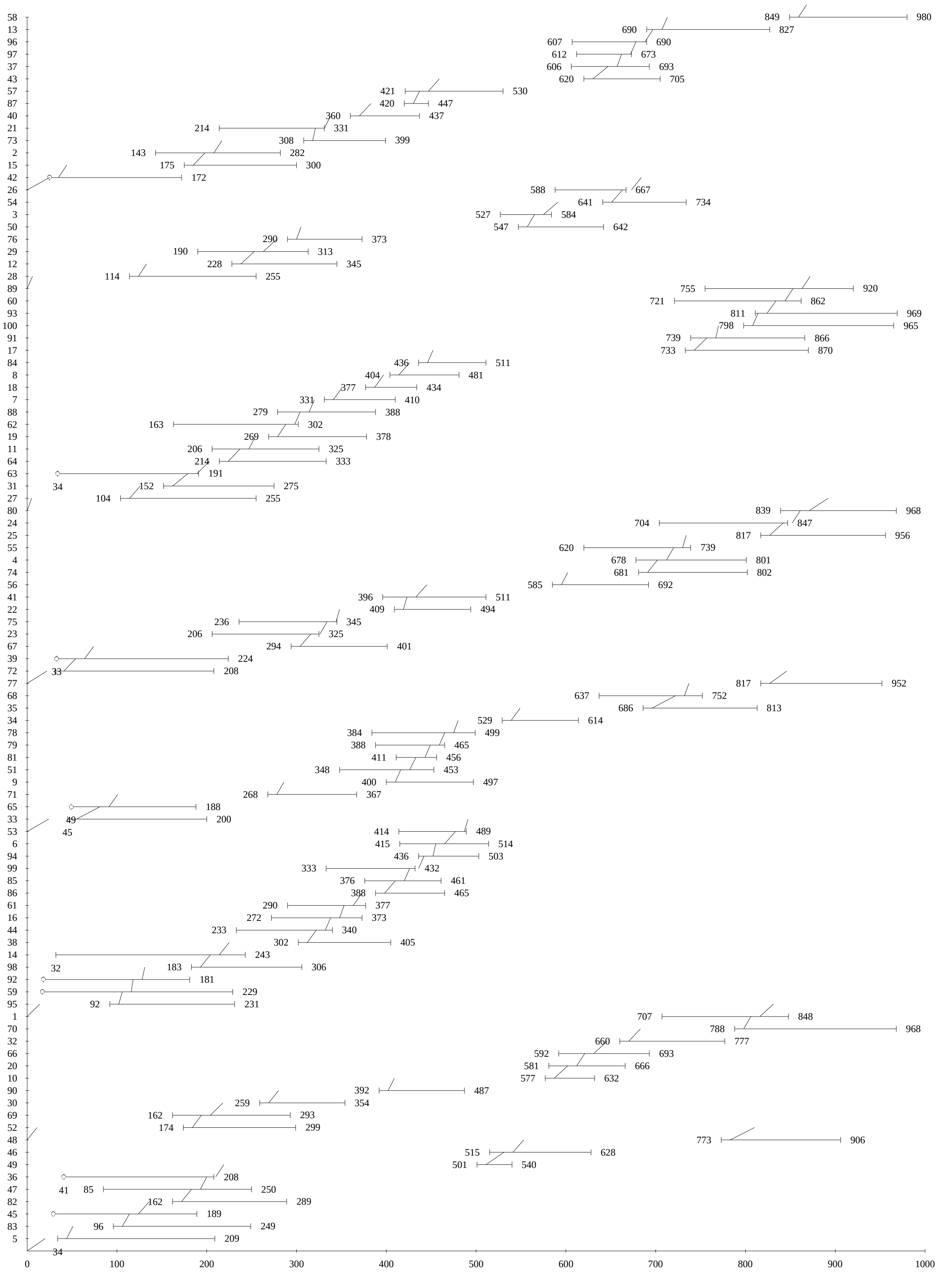

Figure 5: Tidsmæssig view of the solution to R201 with 100 customers.

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 try several times to solve C204.100, but either stopped due to an upper bound of 100000 columns generated or in the a later attemp ran out of memory. The problems are structually the same as the C1 instances, men longer routes are allowed, which can be seen in the solution of C208.100 in figure 6. On figure 7 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	Beregningstid
C201.25AB	214,700	214,700	2	1	0	173	1,0
C201.50AB	360,200	360,200	3	1	0	177	1,1
C201.100AB	589,100	589,100	3	1	0	105	58,2
C202.25AB	214,700	214,700	2	1	0	916	7,1
C202.50AB	360,200	360,200	3	1	0	923	17,2
C202.100AB	589,100	589,100	3	1	0	134	21,0
C203.25AB	214,700	214,700	2	1	0	968	26,1
C203.50B	359,800	359,800	3	1	0	930	80,2
C203.100B	588,700	588,700	3	1	0	287	609,6
C204.25B	211,004	214,500	2	6	0	778	386,4
C204.50B	350,100	350,100	2	1	0	172	64,3
C204.100							
C205.25AB	212,050	214,700	2	2	0	825	4,4
C205.50B	357,350	363,500	3	2	0	646	387,5
C205.100B	586,400	586,400	3	9	0	151	12,0
C206.25A	197,700	214,700	2	1	5	573	12,1
C206.50A	344,200	359,800	3	3	4	733	1353,5
C206.100A	585,400	586,000	3	1	2	565	1938,0
C207.25B	207,981	214,500	2	73	0	1194	819,7
C207.50A	356,269	359,600	3	1	10	1208	910,8
C207.100A	581,969	585,800	3	1	6	851	5475,8
C208.25A	193,280	214,500	2	9	6	2415	96,5
C208.50A	340,425	350,500	2	1	3	1220	394,9
C208.100A	581,767	585,800	3	1	4	1173	8326,6

Table 5: Solution overview for the C2 instances.

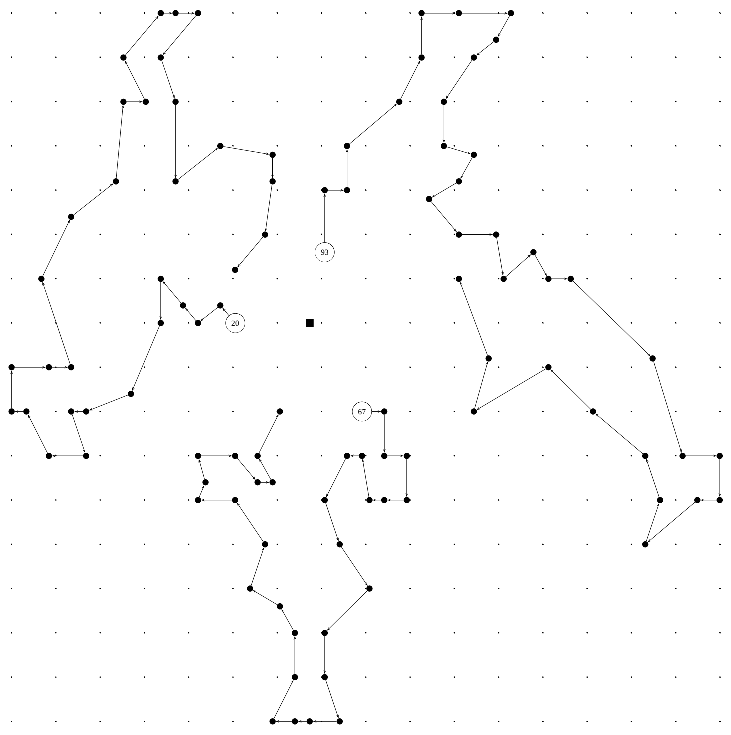

Figure 6: Geographic view of the solution to C208 with 100 customers.

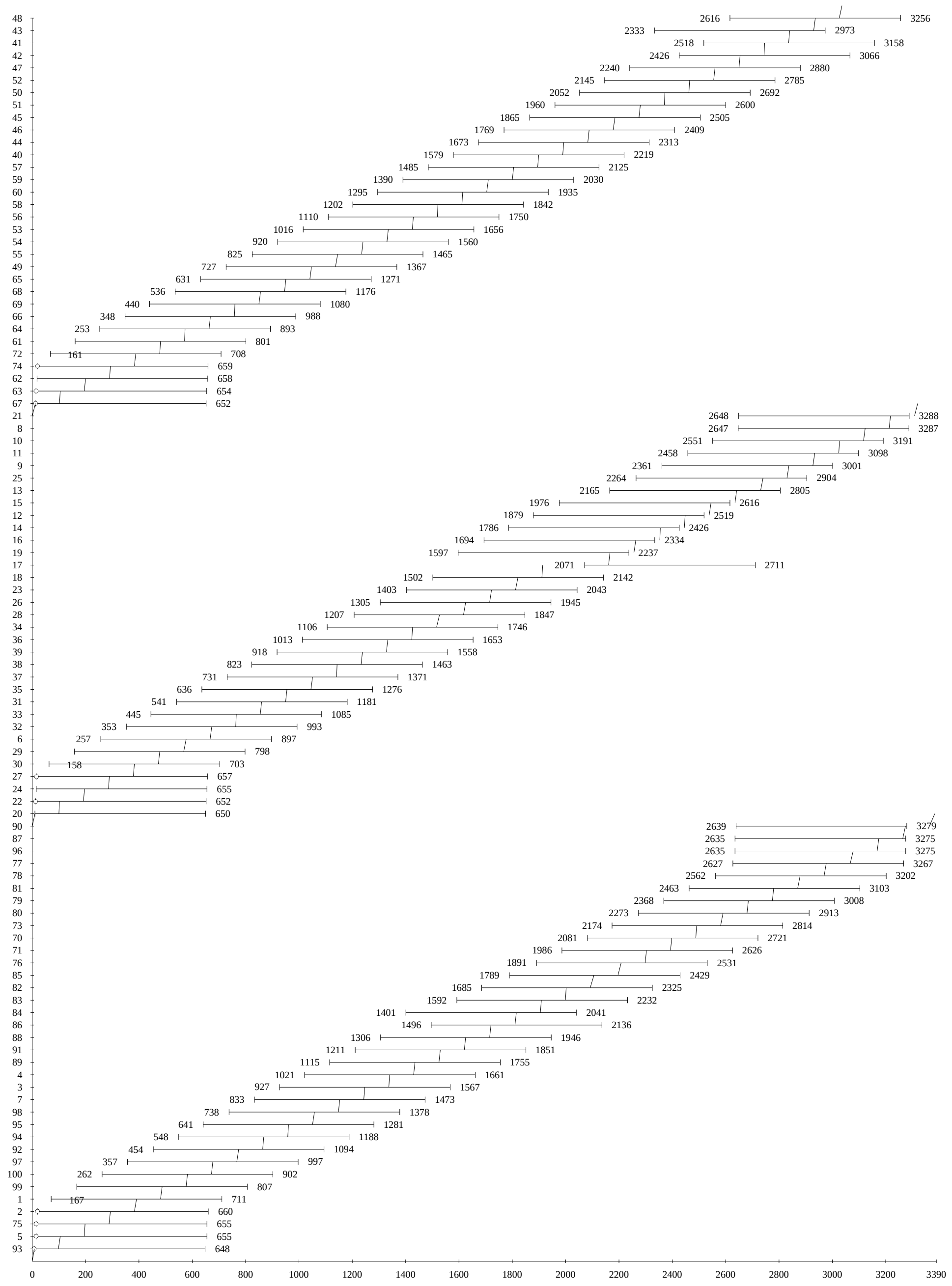

Figure 7: The solution to C208 with 100 customers from a time perspective.

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 show in figure 8 and 9. 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	Beregningstid
RC201.25	356,650	360,200	3	3	0	419	2,3
RC201.50	670,150	684,800	5	31	0	1361	94,2
RC201.100CD	1240,398	1261,800	9	524	10	33642	34869,0
RC202.25A	290,408	338,000	3	117	6	1791	6386,7
RC202.50							
RC202.100							
RC203.25CD	214,475	356,400	2	12399	5	385100	213415,1
RC203.50							
RC203.100							
RC204.25							
RC204.50							
RC204.100							
RC205.25	307,600	338,000	3	47	0	1551	57,9
RC205.50	541,842	631,000	5	8938	0	266170	55104,3
RC205.100							
RC206.25	250,390	324,000	3	2465	8	27327	82387,5
RC206.50							
RC206.100							
RC207.25CD	217,964	298,300	3	13395	3	215645	220991,2
RC207.50							
RC207.100							
RC208.25							
RC208.50							
RC208.100							

Table 6: Solution overview for the RC2 instances.

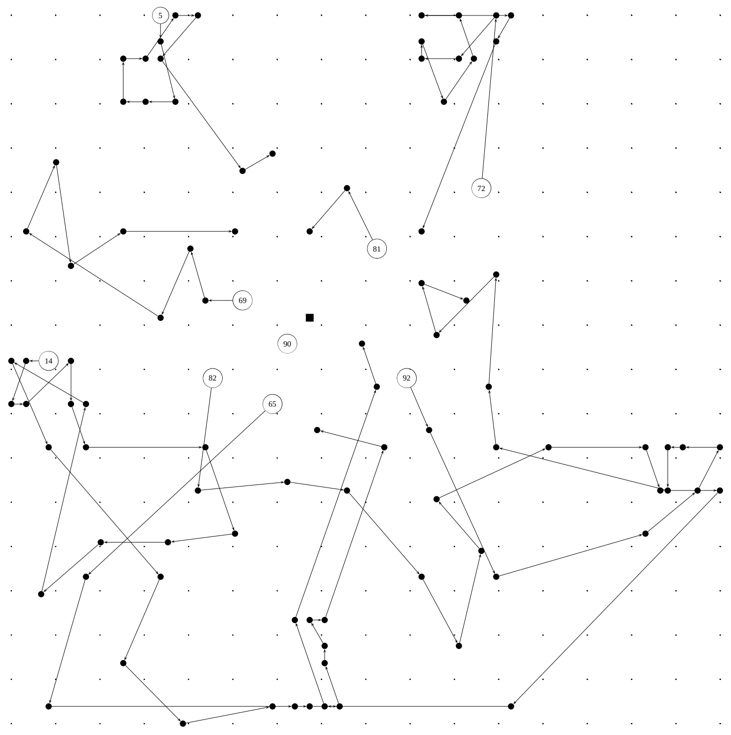

Figure 8: Geographic view of the solution to RC201 with 100 customers.

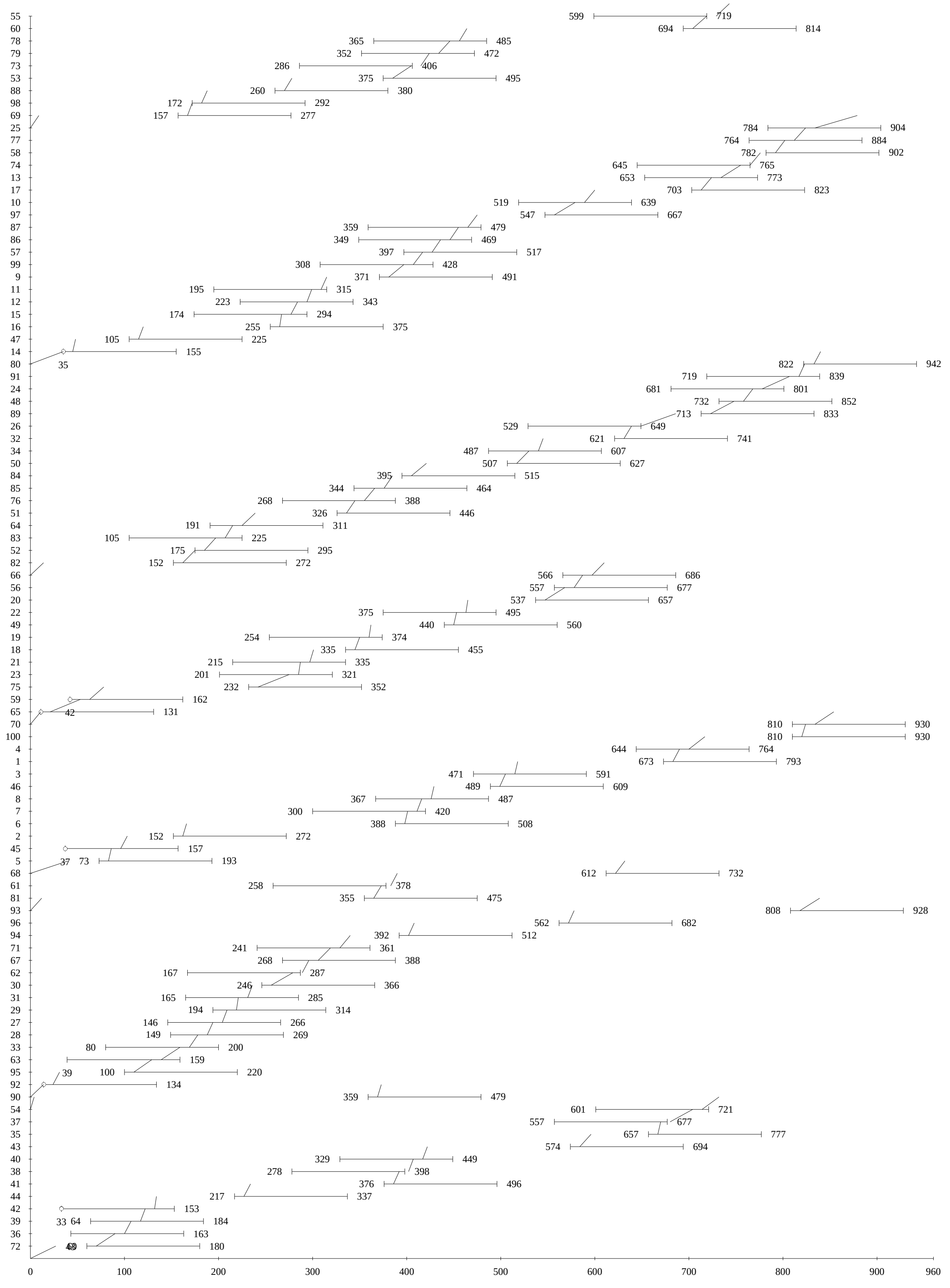

Figure 9: The solution to RC201 with 100 customers from a time perspective.

4.7 The Homberger instances

We have solved 9 problems from the Homberger testsets, 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 placed.

Problem	LB$_{opt}$	IP$_{opt}$	veh	no	VI	iter	Beregningstid
R1_2_1.200	4654,900	4667,200	23	469	21	1314	4562,1
C1_2_1.201	2698,600	2698,600	20	1	0	149	121,3
C1_2_2.202	2682,187	2694,300	20	95	6	1113	4569,4
C1_2_5.203	2694,900	2694,900	20	1	0	152	116,5
C1_2_6.204	2694,900	2694,900	20	1	0	155	94,8
C1_2_7.205	2694,900	2694,900	20	1	0	181	134,6
C1_2_8.206	2667,870	2684,000	20	129	13	967	1970,6
C1_4_1.400	7138,767	7138,800	40	1	1	17472	9242,0
C110_1.1000	42444,400	42444,800	100	3	2	996	28952,2

Table 7: Overview of the solved Homberger instances.

4.8 An instance with 1000 customers

The solution to an instance with 1000 customers is shown in figure10. The objective value is 42444.8 and the solutions needed 100 vehicles. The Bundle-method solved the root node in 704 iterations in 4116 seconds. The Dantzig-Wolfe algorithm used 292 master iterations, returning at most 200 column for every call of the SPPTWCC subproblem and used a total of 24836 sekunder to find the optimal solution. 2 valied 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 8: Programoutput for solving C110_1.1000.

In table 8 one can see that the main part of the running time is used in the LP-solver

(23872 sekundes out of a total of 24836 seconds). This 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 [Kal] p.82).

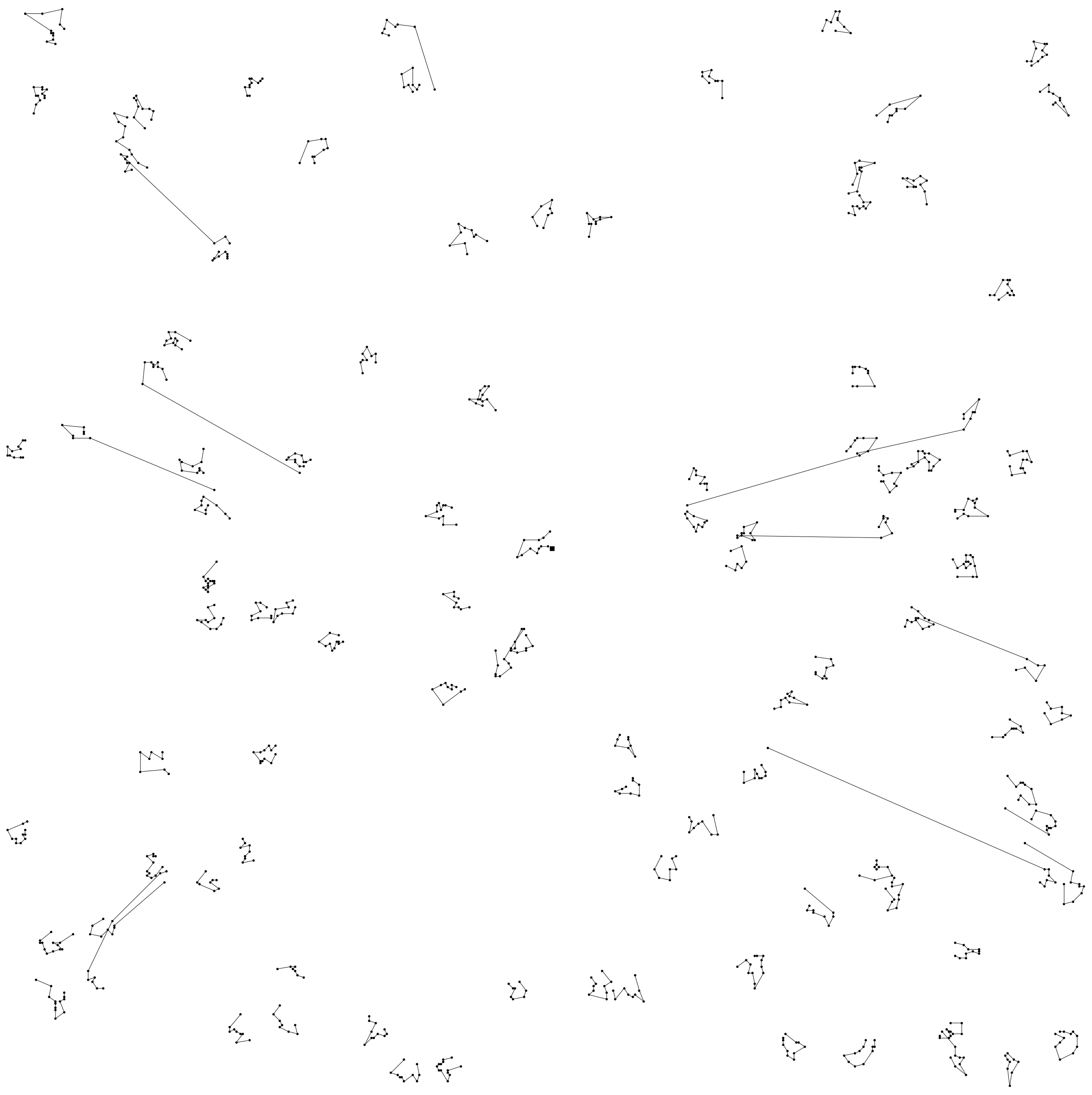

Figure 10: The solution to C110_1.1000.

References

[BGAB83] Lawrence Bodin, Bruce Golden, Arjang Assad, and Michael Ball. Routing and scheduling of vehicles and crews - the state of art. *Computers & Operations Research*, 10(2):62 – 212, 1983.

[Bre95] Alex Van Breedam. Vehicle routing: Bridging the gap between theory and practice. *Belgian Journal of Operations Research, Statistics and Computer Science*, 35(1):63 – 80, 1995.

[CL98] Teodor Gabriel Crainic and Gilbert Laporte. *Fleet Management and Logistics*. Kluwer, 1998.

[CR99] William Cook and Jennifer L. Rich. A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Departement of Computational and Applied Mathematics, Rice University, 1999.

[DDSS93] Jacques Desrosiers, Yvan Dumas, Marius M. Solomon, and F. Soumis. Time constrained routing and scheduling. Technical report, GERAD, September 1993.

[Fis97] Marshall Fisher. Vehicle routing. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, chapter 1, pages 1 – 79. North-Holland, 1997.

[Hal92] Karsten Halse. *Modeling and Solving Complex Vehicle Routing Problems*. PhD thesis, Department for Mathematical Modeling, Technical University of Denmark, 1992.

[Hom] Jörg Homberger. Extended solomon's vrptw instances. Available on the web at `www.fernuni-hagen.de/WINF/touren/inhalte/probinst.htm`.

[Kal] Brian Kallehauge. Solutions to the solomon instances for vrptw. Supplementary report for Masters Thesis no. 13, [in Danish].

[Koh95] Niklas Kohl. *Exact methods for Time Constained Routing and Related Scheduling Problems*. PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, 1995. IMM-PHD-1995-16.

[Lar99] Jesper Larsen. *Parallellization of the Vehicle Routing Problem with Time Windows*. PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, 1999. IMM-PHD-1999-62.

[The98] The Danish Ministry of Transport. Trafikredegørelse 1997, January 1998. [in danish].