An Optimization Algorithm for the Vehicle Routing Problem with Time Windows based on Lagrangean Relaxation

Niklas Kohl Oli B.G. Madsen

Institute of Mathematical Modelling
The Technical University of Denmark
DK-2800 Lyngby, Denmark

Revised: August 8, 1995

Our paper presents a new optimization method for the Vehicle Routing Problem with Time Windows (VRPTW). The VRPTW is a generalization of the Vehicle Routing Problem, where the service of a customer must start within a given time interval - a so-called time window. Our method is based on a Lagrangean relaxation of the constraint set requiring that each customer must be serviced. The master problem consists of finding the optimal Lagrangean multipliers and the subproblem is a Shortest Path Problem with Time Windows and Capacity Constraints. The optimal multipliers are found using a method exploiting the benefits of subgradient methods as well as a bundle method. The method has been implemented and tested on a series of well known benchmark problems of size up to 100 customers. Our algorithm turns out to be very competitive compared to algorithms considered in the literature, and we have succeeded in solving several previously unsolved problems.

Introduction:

Routing and scheduling problems are major elements of many logistic systems, and a large amount of research has been devoted to find good solutions to these complex combinatorial optimization problems. One of the major research topics has been the Vehicle Routing Problem (VRP) which involves finding a set of routes, starting and ending at a depot, which together cover a set of customers. Each customer has a given demand, and no vehicle can service more customers than its capacity enables it to. The objective can be to minimise the total distance travelled or the number of vehicles used or a combination of these objectives.

In this paper we consider the Vehicle Routing Problem with Time Windows (VRPTW) which is a generalization of the VRP. A solution to the VRPTW must ensure that the service of any customer starts within a given time interval, a so called time window. The VRPTW can be considered as a special case of Resource Constrained Routing which also contains problems such as the multiple Travelling Salesman Problem with Time Windows, the Dial-a-Ride Problem and Crew Scheduling Problems (see Desrosiers et al. (1995)). The results presented in this paper can be generalised to these problems.

The most successful approaches for the VRPTW are based on constrained shortest path relaxations. Desrochers, Desrosiers and Solomon (1992) have used a column generation (Dantzig-Wolfe decomposition) scheme and Halse (1992) has used a decomposition based on Variable Splitting (also known as Lagrangean decomposition). In both cases the resulting subproblem is a shortest path problem with time and capacity constraints. Even though this problem is $\mathcal{NP}$-hard a pseudo polynomial algorithm exists, and most instances of moderate size are solvable.

We have developed an algorithm exploiting Lagrangean relaxation of the constraint set requiring that all customers must be serviced. The master problem consists of finding the optimal Lagrangean multipliers. The subproblem is a time and capacity constrained shortest path problem for each vehicle. This approach creates relatively easy instances of the subproblem compared to the Dantzig-Wolfe decomposition and is more simple than the Variable Splitting approach, since the number of Lagrangean multipliers is much smaller.

However, it is often a major problem to find the optimal Lagrangean multipliers, and the commonly used subgradient method gives slow convergence. Therefore we have developed a more sophisticated master iteration scheme exploiting a number of results in non-differentiable optimization theory. On a number of benchmark problems by Solomon (1987) the computational times are very competitive to those reported by Desrochers, Desrosiers and Solomon (1992) and Halse (1992). Furthermore we have solved a number of previously unsolved problems.

In Section 1 we present a mathematical model of the VRPTW. Section 2 gives a brief review of existing optimization methods. We focus especially on the methods of Desrochers, Desrosiers and Solomon (1992) and Halse (1992), since these methods are quite similar to our method. Our method is considered in Section 3. We show how a lower bound on the number of vehicles can be exploited and compare the lower bounds obtained with those of Desrochers, Desrosiers and Solomon (1992) and Halse (1992). Section 4 describes how the subproblem, a constrained shortest path problem, is solved and we provide information on how to estimate the difficulty of a particular instance of the problem. In Section 5 the master problem, a non-differentiable concave maximization problem, is considered. Section 6 gives a brief description of the branching strategy implemented and Section 7 reports on the computational experiments on the set of benchmark problems. Finally Section 8 contains our conclusions.

1 A mathematical model of the VRPTW

The VRPTW is given by a set of identical vehicles $\mathcal{V}$, a set of customers $\mathcal{C}$ and a directed network connecting the customers and a depot. For the sake of simplicity the number of customers $|\mathcal{C}|$ will also be denoted n and the customers will be denoted $1, 2, ..., n$. The depot is represented by two nodes denoted 0 and $n+1$. The arcs of the network correspond to connections between the nodes. No arc terminates in node 0 and no arc originates in node $n+1$. All routes start at 0 and end at $n+1$. The set of all nodes is denoted $\mathcal{N}$. A cost c_{ij} and a travel time t_{ij} is associated with each arc (i,j), $i \neq j$ of the network. The travel time t_{ij} may include a service time at customer i. If vehicles do not have to be used, then $t_{0,n+1} = 0$ and $-c_{0,n+1}$ represents the cost of using a vehicle. A non-existing or non-feasible arc (i,j) can be modelled using a large cost c_{ij}, so apart from node 0 and $n+1$ we can assume that the network is fully connected. Each vehicle has a given capacity q and each customer a demand d_i, $i \in \mathcal{C}$.

At each customer, the start of the service must be within a given time interval, a so called time window, $[a_i, b_i]$, $i \in \mathcal{C}$. Also vehicles must leave the depot within the time window $[a_0, b_0]$ and return in the time window $[a_{n+1}, b_{n+1}]$. A vehicle is permitted to arrive before the beginning of the time window, and wait at no cost until service is possible, but it is not permitted to arrive after the end of the time window. With out loss of generality we can assume that $a_0 = b_0 = 0$. We will further assume that all data, ie. q, d_i, c_{ij}, t_{ij}, a_i and b_i are non-negative integers. For technical reasons, which are discussed in Section 4, all t_{ij} must be strictly positive.

The triangle inequality, that is $c_{ij} \leq c_{ih} + c_{hj}$ and $t_{ij} \leq t_{ih} + t_{hj}$, $\forall h, i, j \in \mathcal{N}$, does not have to be satisfied. It is possible to add any scalar to all arc costs c_{ij}, $i \in \mathcal{C}$, without changing the optimal solution, so any instance of VRPTW can be transformed to an instance satisfying the triangle inequality in cost. This is not true for the time triangle inequality, but it seems that most applications will satisfy this property. This is especially true if t_{ij} includes a service time at i.

The model contains two types of decision variables. The decision variable x_{ijk} (defined $\forall i, j \in \mathcal{N}$, $\forall k \in \mathcal{V}$, $i \neq j$, $i \neq n+1$, $j \neq 0$) is 1 if vehicle k drives from node i to node j, and 0 otherwise. The decision variable s_{ik} (defined $\forall i \in \mathcal{N}$, $\forall k \in \mathcal{V}$) denotes the time a vehicle k, $k \in \mathcal{V}$, starts service at customer i, $i \in \mathcal{C}$. If vehicle k does not service customer i, s_{ik} does not mean anything. We may assume that $s_{0k} = 0$, $\forall k$, and $s_{n+1,k}$ denotes the arrival time of vehicle k to the depot. The objective is to design a set of minimal cost routes, one for each vehicle, so that all customers are serviced. The routes must be feasible with respect to the capacity of the vehicles and the time windows of the customers serviced.

The VRPTW can be stated mathematically as:

$$
\begin{aligned}
min \quad & \sum_{k\in\mathcal{V}}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} c_{ij}x_{ijk}, && s.t. && \\
& \sum_{k\in\mathcal{V}}\sum_{j\in\mathcal{N}} x_{ijk} && = 1 && \forall i \in \mathcal{C} && (1)\\
& \sum_{i\in\mathcal{C}} d_i \sum_{j\in\mathcal{N}} x_{ijk} && \leq q && \forall k \in \mathcal{V} && (2)\\
& \sum_{j\in\mathcal{N}} x_{0jk} && = 1 && \forall k \in \mathcal{V} && (3)\\
& \sum_{i\in\mathcal{N}} x_{ihk} - \sum_{j\in\mathcal{N}} x_{hjk} && = 0 && \forall h \in \mathcal{C},\ \forall k \in \mathcal{V} && (4)\\
& \sum_{i\in\mathcal{N}} x_{i,n+1,k} && = 1 && \forall k \in \mathcal{V} && (5)\\
& s_{ik} + t_{ij} - K(1 - x_{ijk}) && \leq s_{jk} && \forall i \in \mathcal{N},\ \forall j \in \mathcal{N},\ \forall k \in \mathcal{V} && (6)\\
& a_i \leq s_{ik} \leq b_i && && \forall i \in \mathcal{N},\ \forall k \in \mathcal{V} && (7)\\
& x_{ijk} \in \{0,1\} && && \forall i \in \mathcal{N},\ \forall j \in \mathcal{N},\ \forall k \in \mathcal{V} && (8)
\end{aligned}
$$

The objective function states that costs should be minimised. For the benchmark problems used (cf. Section 7), costs reflect the distance travelled, but other objectives are easily modelled. Constraint set (1) states that each customer must be assigned to exactly one vehicle. We will denote this set of constraints the assignment constraints. Constraint set (2) states that no vehicle must service more customers than its capacity enables it to. Constraint sets (3), (4) and (5) are the flow constraints requiring that each vehicle k leaves node 0 once, leaves node i, $i \in \mathcal{C}$ if and only if it enters the node and that the vehicle returns to node $n+1$. Note that constraint set (5) is redundant, but is maintained in the model to underline the network structure. Since the arc $(0, n+1)$ is included in the network the "empty" tour is permitted. Constraint set (6) states that vehicle k cannot arrive at j before $s_{ik} + t_{ij}$ if it travels from i to j. The scalar K can be any large number. Constraint set (7) ensures that all time windows are respected and (8) are the integrality constraints. Desrosiers et al. (1995) introduce the nonlinear formulation $x_{ijk}(s_{jk} - s_{ik} - t_{ij}) \leq 0$ of constraint set (6). In their model the constraint set (8) is not necessary because there will always exist an optimal solution to the continuous relaxation of the problem which is integer, but since the feasible region is non-convex, this formulation is not useful in practical computations.

The model permits a fixed number of vehicles or an upper limit on the number of vehicles given by $|\mathcal{V}|$. If this upper limit is sufficiently high, it means that the number of vehicles is free. A lower bound on the permitted number of vehicles is also easily added. A free number of vehicles is modelled by setting $c_{0,n+1} = 0$. If the number of vehicles is fixed at $|\mathcal{V}|$ one can set $c_{0,n+1}$ or $t_{0,n+1}$ to a large value ensuring that this arc will not be used in any optimal solution. Another modelling option is to permit a free number of vehicles, but put a cost c_v on each vehicle used. This can be done by setting $c_{0,n+1} = -c_v$, $\forall k \in \mathcal{V}$. If c_v is sufficiently large, the model will primarily minimise the number of vehicles and secondarily minimise travel costs.

The VRPTW includes several well known problems such as the Travelling Salesman Problem, the Vehicle Routing Problem and the Bin Packing Problem. Finding an optimal solution to the VRPTW is a $\mathcal{NP}$-hard problem, and even finding a feasible solution when $|\mathcal{V}| < |\mathcal{C}|$ is $\mathcal{NP}$-hard, since this includes the problem of packing demand on $|\mathcal{V}|$ vehicles (equivalent to bin packing).

2 Review of existing optimization methods

Since the VRPTW is $\mathcal{NP}$-hard early research has focused on case studies and heuristic approaches. The first paper on an optimization method for the VRPTW appeared as late as 1987. Heuristics for the VRPTW is still an active research field, but the methods and techniques applied have very little in common with the research on optimization methods. Desrosiers et al. (1995) contains a review of these heuristic methods, but we will leave the issue at this point since the focus of this paper is on optimization methods. In what follows we will only discuss these methods.

Kolen, Rinnooy Kan and Trienekens (1987) have presented the first optimization method for the VRPTW. The method calculates lower bounds using dynamic programming and state space relaxation. Branching decisions are taken on route-customer allocations. The largest problem solved contained 15 customers.

Fisher, Jörnsten and Madsen (1992) describe an algorithm based on the K-tree relaxation of the VRPTW. Capacity constraints are handled by introducing a constraint requiring that some set $\mathcal{S}$, $\mathcal{S} \subset \mathcal{C}$, of customers must be serviced by at least $k(\mathcal{S})$ vehicles. This constraint is Lagrangean relaxed and the problem resulting is still a K-tree problem with modified arc costs. Time windows are treated similarly. A constraint, requiring that not all arcs in a time violating path can be used, is generated and Lagrangean relaxed. The method has solved a few problems with up to 100 customers from the Solomon benchmark problems (Solomon 1987).

At present, the most successful algorithms for the VRPTW are based on shortest path decompositions of the problem. The fundamental observation is, that only constraint (1) is "linking" the vehicles together. The problem consisting of (2), ... (8) is an Elementary Shortest Path Problem with Time Windows and Capacity Constraints (ESPPTWCC) for each vehicle. This problem is $\mathcal{NP}$-hard in the strong sense (Dror 1994), but very efficient Dynamic Programming algorithms for the slightly relaxed problem SPPTWCC exist. Furthermore most of the integrality gap between the LP- and IP-solution of VRPTW is often explored in the SPPTWCC. Two decompositions have been investigated computationally: Dantzig-Wolfe decomposition and Variable Splitting.

Desrochers, Desrosiers and Solomon (1992) (see also Desrosiers et al. (1995)) have applied column generation to the VRPTW with a free number of vehicles. As noted in Desrosiers et al. (1995) this column generation is in fact equivalent to Dantzig-Wolfe decomposition, so we will denote their approach Dantzig-Wolfe decomposition even though this term is not used in Desrochers, Desrosiers and Solomon (1992). The Dantzig-Wolfe decomposition exploits the fact that only constraint set (1) is linking the vehicles together. The method starts out with a solution consisting of $|\mathcal{C}|$ paths, each servicing one customer. The master problem consists of finding a minimum cost set of paths, among all generated paths, that together ensures that all customers are serviced. The number of times a path is used is not necessarily an integer, but can be any number in the interval $[0,1]$. Since the generated paths are not elementary paths, that is the same customer can be serviced more than once on a path, the master problem is the LP-relaxation of a Set Partitioning type problem. For computational reasons Desrochers, Desrosiers and Solomon (1992) solves the LP-relaxation of a Set Covering type problem instead. This does not change the final result, if the triangle inequality on time and cost is satisfied.

The simplex multipliers of the optimal solution are used to modify costs in the subproblems, which are shortest path problems with time windows and capacity constraints. The subproblems are identical, so only one problem is solved. If a path with negative marginal cost is found, it is included in the set of paths in the master problem. Since one might find several negative marginal cost paths, each iteration can generate several paths.

The method terminates when no negative marginal cost paths can be generated in the subproblem. At this point all paths used in the optimal solution to the master problem will have marginal cost 0 and all other paths will have a non-negative marginal cost. If the master problem turns out to have an integer solution, it is optimal. Otherwise a lower bound on the optimal objective value of VRPTW has been found and some branch-and-bound strategy must be employed to find the optimal solution. The method has solved problems with up to 100 customers from the Solomon data set, including many problems other methods have not been able to solve.

Jörnsten, Madsen and Sørensen (1986) suggested the decomposition of the VRPTW by Variable Splitting. In Variable Splitting the variables in some of the constraints are renamed. A new type of constraint, coupling the original and the new variables, is introduced and Lagrangean relaxed. This decomposes the problem into two or more independent problems. In the VRPTW this means that $\sum_j x_{ijk}$ is replaced by the binary variable y_{ik} in some constraints. The constraint $\sum_j x_{ijk} = y_{ik}, \forall i \in \mathcal{C}, \forall k \in \mathcal{V}$ is introduced and Lagrangean relaxed. Now the problem separates into two problems, one in the x_{ijk} and s_{ik} variables and one in the y_{ik} variables. The Variable Splitting method is some times called Lagrangean decomposition, and has been described by Jörnsten, Näsberg and Smeds (1985) and Guignard and

Kim (1987). For the VRPTW there are three natural ways of separating the problem.

The model VS_1 consists of a Shortest Path Problem with Time Windows (SPPTW), that is constraints (3), ... (8), and a Generalised Assignment Problem (GAP), i.e. constraints (1) and (2). Model VS_2 consists of a Shortest Path Problem with Time Windows and Capacity Constraints (SPPTWCC), all constraints except (1), and a so called Semi Assignment Problem (SAP) consisting of constraints (1) only. SAP is basically a GAP without capacity constraints. In VS_3 constraints (2) are duplicated and modelled into both problems. A SPPTWCC and a GAP are obtained.

Halse (1992) has implemented the VS_2 method and solved problems from the Solomon data set with up to 100 customers. Furthermore he solved a 105 customer problem composed of data from two of the Solomon problems. Olsen (1988) has implemented the VS_1 method and solved problems with up to 16 customers. The VS_3 model has not been implemented.

3 A new approach based on Lagrangean Relaxation

3.1 The Lagrangean Relaxation

In this section we present a new solution approach for the VRPTW based on Lagrangean Relaxation. The problem decomposes as in the Dantzig-Wolfe decomposition, but the way the dual variables (in this context Lagrangean multipliers) are optimised is new in this context. Furthermore we will show how a lower bound on the number of vehicles can be exploited and we will compare the quality of the lower bounds obtained using this decomposition with the lower bounds obtained by the Dantzig-Wolfe decomposition and the VS_2 Variable Splitting method. Our method uses a Lagrangean Relaxation of the assignment constraint (1).

The method is closely related to the Dantzig-Wolfe decomposition of Desrochers, Desrosiers and Solomon (1992) and the VS_2 Variable Splitting method of Halse (1992) since the subproblem - a SPPTWCC - is the same for all methods.

Using Lagrangean relaxation is not a new idea in the context of time constrained routing problems. Desrosiers, Sauvé and Soumis (1988) has used Lagrangean relaxation of the assignment constraint to estimate the minimum fleet size in the multiple Travelling Salesman Problem with Time Windows (m-TSPTW). The m-TSPTW is just a VRPTW without capacity constraints, i.e. constraint set (2). In their recent review of time constrained routing problems Desrosiers et al. (1995) focus on Dantzig-Wolfe decomposition based methods. They conclude that Dantzig-Wolfe decomposition has produced better computational results than other approaches. This is because the information obtained in the subproblem is exploited best in the Dantzig-Wolfe master problem (LP-relaxation of a Set Covering type problem), since all generated paths can be exploited in this master problem. Desrosiers et al. (1995) also note that the Dantzig-Wolfe decomposition provides better information to design the branch-and-bound method. This paper shows how this information can be used in a Lagrangean relaxation scheme.

If constraint set (1) is Lagrangean relaxed the objective function can be written as:

$$minimize \quad \sum_{k \in \mathcal{V}} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} (c_{ij} - \lambda_j) x_{ijk} + \sum_{j \in \mathcal{C}} \lambda_j, \;\; subject\, to$$

$$(2),\ (3),\ (4),\ (5),\ (6),\ (7)\ and\ (8) \tag{9}$$

Here λ_j is the multiplier associated with the constraint requiring that customer j must be serviced. We will denote the modified (or marginal) arc costs $\hat{c}_{ij} = c_{ij} - \lambda_j$. The multiplier

λ_j reflects the costs of servicing a customer, and will generally be positive. However, since constraint set (1) are equality constraints, the multipliers may take negative values as well. If multipliers are very large, long routes servicing many customers will be attractive in the subproblem. If multipliers are small (or negative) very short routes will generally be attractive in the subproblem.

For any given value of the multiplier vector $\underline{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_{|\mathcal{C}|})$, the resulting model decomposes into one subproblem for each vehicle. However, the subproblems are identical since their constraint sets are identical (recall that the vehicles are identical) and the arc costs are independent of the vehicle. The subproblem is an elementary shortest path problem, i.e. constraint sets (3), (4) and (5), from the depot (node 0) back to the depot (node $n+1$) with the additional constraint sets (2), (6), (7) and (8). Constraint set (2) are the capacity constraints. Constraint sets (6) and (7) are the time window constraints and (8) is the integrality constraint, i.e. the problem is an ESPPTWCC. The arc costs have been modified by the multipliers, so we can no longer assume that they are neither integers nor non-negative.

The ESPPTWCC is unsolvable for practical purposes, but the slightly relaxed problem SPPTWCC is solvable in most cases by dynamic programming, cf. Section 4. The SPPTWCC permits a path servicing the same customer more than once as long as the accumulated time and capacity consumption permits this. Solving the SPPTWCC instead of the ESPPTWCC may decrease the lower bound obtained, but the computational experiments of Desrochers, Desrosiers and Solomon (1992) shows that the bound is still of excellent quality.

The Lagrangean multipliers associated with the assignment constraints can be interpreted as a price given for servicing the customer associated with the constraint. Each vehicle chooses its own path, minimising the cost of the path modified by the multipliers of the customers serviced. If the number of vehicles is free or bounded from above, any vehicle has the option of using the $(0)-(n+1)$ path, i.e. no route, at no cost. If the number of vehicles is bounded from below, for example by the constraint $\sum_k \sum_{j \in \mathcal{C}} x_{0jk} \geq v_{min}$, this constraint is Lagrangean relaxed. The corresponding Lagrangean multiplier modifies the cost of arcs originating in 0 and terminating in j, $j \in \mathcal{C}$, by the formula $\hat{c}_{ij} = c_{ij} - \lambda_j - \gamma_{min}$, where γ_{min} denotes the multiplier associated with the minimum number of vehicles constraint.

For any multiplier vector $\underline{\lambda}$, the objective function value of the optimal solution to the relaxed problem, denoted $f(\underline{\lambda})$, gives a dual lower bound on the optimal objective value for the VRPTW (Geoffrion 1974). We will denote the cost of a path in the SPPTWCC problem the *marginal* cost. If z^* denotes the marginal cost of the shortest path, i.e. $z^* = \sum_i \sum_j \hat{c}_{ij} x_{ijk} = \sum_i \sum_j (c_{ij} - \lambda_j) x_{ijk}$ for the optimal solution to the relaxed problem for any vehicle k, then (9) yields the formula $f(\underline{\lambda}) = |\mathcal{V}| z^* + \sum_i \lambda_i$. We will denote a maximiser of f by $\underline{\lambda}^*$.

If the SPPTWCC has a unique optimum, all vehicles will choose the same path, since vehicles are identical. Unless the problem contains only one vehicle, this cannot be a feasible solution to the VRPTW, since customers on the shortest path are serviced $|\mathcal{V}|$ times and all other customers are not serviced at all. However, we will show that the solution to the SPPTWCC is generally not unique, when the best Lagrangean multipliers have been found.

Proposition 1.

Assume f is bounded from above. If $\underline{\lambda} = \underline{\lambda}^*$ then for any node i, $i \in \mathcal{C}$, there exists a path of minimal marginal cost servicing customer i.

Proof.

Let $\underline{\lambda} = \underline{\lambda}^*$. Let z_i denote the marginal cost of the shortest path passing node i, $i \in \mathcal{C}$, i.e. $z^* = min_{i \in \mathcal{C}}(z_i)$. Assume that node i is not on any shortest path, i.e. $z_i - \epsilon = z^*$, $\epsilon > 0$. Now increase λ_i with ϵ, ie. $\lambda_i := \lambda_i + \epsilon$. This decreases z_i down to z^*, but does not change z^*. Since z^* is unchanged the first term of the relaxed objective function is unchanged, but the second term is increased by ϵ. This contradicts the assumption that $\underline{\lambda}$ is optimal, and therefore shows that i must be on a shortest path if $\underline{\lambda} = \underline{\lambda}^*$. End of proof.

If there exist a set of minimal cost paths, which together cover each customer exactly once, a feasible and optimal solution has been found. Whether this is the case, can in principle be determined by solving a set partitioning problem with columns corresponding to the minimal marginal cost paths and rows corresponding to customers. If the number of vehicles is fixed or bounded this must be included as a constraint. If and only if a feasible solution to this set partitioning problem exists, the VRPTW has been solved to optimality. This procedure is not directly implementable, since we generally do not know *all* minimal marginal cost paths, but only one. This suggests that a naive implementation of Lagrangean relaxation only provides a lower bound on the optimal objective, but is not capable of producing an optimal integer solution (or information for a branch-and-bound procedure). At this point we will not show how to overcome this difficulty. This issue is closely related to solution methods for the master problem, which is the topic of Section 5.

The master problem consists of finding the multipliers $\underline{\lambda}^*$ that yield the best lower bound, i.e. solving the model $maximize_{\underline{\lambda} \in \mathbb{R}^n} f(\underline{\lambda})$. The function f is concave and non-differentiable (see e.g. Lemaréchal 1989). If f is not bounded from above, the original problem is clearly infeasible. The opposite is, however, not true. The problem may be infeasible even if f is bounded from above. A subgradient for $-f(\underline{\lambda})$ is given by $-\underline{g} = -(g_1, g_2, ..., g_n)$, where $g_i = 1 - \sum_j \sum_k x_{ijk} = 1 - |\mathcal{V}| \sum_j x_{ijk}$, since $\sum_j x_{ijk}$ counts how many times node i is serviced on the shortest path used by all $|\mathcal{V}|$ vehicles. If the optimal solution to the SPPTWCC is unique for a given $\underline{\lambda}$, f is differentiable at $\underline{\lambda}$ and $\underline{g}$ is the gradient. If it is not unique, f is not differentiable, and a subgradient $-\underline{g}$ for $-f$ can be calculated using any optimal solution to the SPPTWCC or any linear combination of optimal solutions.

For any given value of the x-variables, f is linear in $\underline{\lambda}$. The number of x-variables is bounded by $|\mathcal{V}||\mathcal{N}|^2$. Since they can only take the values 0 and 1, the number of possible solutions in the x-variables is bounded by $2^{|\mathcal{V}||\mathcal{N}|^2}$ which is generally very large but finite. Since the ESPPTWCC is relaxed to a SPPTWCC, each arc of the network may be used several times (in case of cycles), so x_{ijk} is not bounded by 1, but is still integer and bounded, so the analysis is not fundamentally changed by this. Since the number of solutions in the $x-$variables is finite, f is the minimum of a very large number of linear functions. Consequently f is piecewise linear.

3.2 Exploitation of a lower bound on the number of vehicles

The model does permit a free number of vehicles, but it turns out that the problem is easier to solve if the number of vehicles is fixed, i.e. if the arc $(0, n+1)$ is omitted. Therefore we will present a result relating the two problems to each other.

Consider the fixed number of vehicles problem. We will introduce the function $f_m(\underline{\lambda})$ which measures the lower bound obtained using multipliers $\underline{\lambda}$ and exactly m vehicles. $f_m(\underline{\lambda})$ is maximised by $\underline{\lambda}_m^*$ and z_m^* is the cost of any corresponding shortest path. It is easily seen that $f_m(\underline{\lambda}_m^*) = mz_m^* + \sum_i \lambda_{m(i)}^*$, where $\lambda_{m(i)}$ is the i'th element of $\underline{\lambda}_m$. We are now ready to present the following proposition.

Proposition 2.

$f_p(\underline{\lambda}_p^*) \geq f_m(\underline{\lambda}_m^*) + (p-m)z_m^*$

Proof.

$f_p(\underline{\lambda}_p^*) \geq f_p(\underline{\lambda}_m^*) = pz_m^* + \sum_i \lambda_{m(i)}^* = (p-m)z_m^* + mz_m^* + \sum_i \lambda_{m(i)}^* = f_m(\underline{\lambda}_m^*) + (p-m)z_m^*$

In the Dantzig-Wolfe decomposition, this result is easily obtained from a parametric right hand side analysis of a constraint on the number of vehicles. Proposition 2 can be useful in several ways. In this work we are using it to prove that an optimal solution to the VRPTW with a fixed number of vehicles is also optimal for the corresponding problem with a free number of vehicles. First, we have calculated a lower bound, denoted m on the number of vehicles needed to serve all the customers. A simple lower bound is calculated by finding the smallest integer not smaller than total demand divided by the capacity of the vehicles. A possibly tighter bound can be found by solving the corresponding bin-packing problem (Fisher 1994), but this has not been necessary for the instances considered in this work. Second, we have solved the problem with exactly m vehicles. In all cases z_m^* was positive and using the bounds in proposition 2, we were able to conclude that $|\mathcal{V}| > m$ could not be optimal. This is not a surprising result, since it often will be optimal to use the smallest possible number of vehicles.

3.3 Comparison of lower bounds

It is well known, that Dantzig-Wolfe and Lagrangean relaxation methods are equivalent primal and dual methods, if they use the same subproblem in their decomposition mechanisms. Therefore we cannot hope to obtain better bounds than Desrochers, Desrosiers and Solomon (1992). In the free number of vehicles case it is clear that there exists a dual solution $\underline{\lambda}$ maximising $f(\underline{\lambda})$ for which $z^* = 0$. In that case $f(\underline{\lambda}) = \sum_i \lambda_i$. This is exactly the objective function value of the LP-dual of the Dantzig-Wolfe master problem, since all the right hand side values of the primal are 1. The fixed number of vehicles case is only slightly more complicated. In the Dantzig-Wolfe decomposition a fixed number of vehicles appears as an additional constraint in the master problem. The simplex multiplier on this constraint, which can be interpreted as the marginal cost of an additional vehicle, is used to modify the cost of the arc $(0, n+1)$ in the SPPTWCC subproblem. At the optimum this marginal cost will be equal to the modified cost of the shortest paths z^*. Therefore the optimal value of the LP-dual is $|\mathcal{V}|z^* + \sum_i \lambda_i$ which is also the value of $f(\underline{\lambda})$.

Now consider the VS_2 method. The two subproblems are SAP and SPPTWCC. The SAP posses the integrality property, i.e. there exists an optimal solution to the LP-relaxation of SAP which is naturally integral. Therefore the lower bound obtained by Variable Splitting is, in this case, not better than the lower bound obtained by an ordinary Lagrangean relaxation of constraint set (1) (Guignard and Kim 1987).

4 Solution method for the subproblem

Recall from Section 3 that the subproblem is given by

$$minimize \quad \sum_{k \in \mathcal{V}} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} (c_{ij} - \lambda_j) x_{ijk} + \sum_{j \in \mathcal{C}} \lambda_j, \;\; subject\, to$$

$$(2),\ (3),\ (4),\ (5),\ (6),\ (7)\; and\, (8) \tag{9}$$

with the modification that cycles are permitted, cf. Section 3. Compared to the usual shortest path problem with non-negative arc costs, which can be solved in $O(n^2)$ time (Nemhauser and Wolsey 1988), this problem is complicated for two reasons. The existence of side constraints, (i.e. constraint set (2), (6) and (7)) is the first reason. The possible existence of negative costs cycles is the other reason. Methods to overcome the first difficulty are well known, but not much is known on how to deal with negative costs cycles.

The time window constraints and capacity constraints are modelled into the network by making one copy of each node in the original network for each permitted combination of time and (used) capacity. To keep the network finite, we must require that time and capacity take values only from a discrete set, such as the natural numbers $\mathbb{N}$. The number of nodes in the new network is given by $\sum_{i \in \mathcal{N}} (b_i - a_i + 1)(q + 1)$. For the benchmark problems studied in Section 7 this approximately means between 500,000 and 17,000,000 nodes in the new expanded network. The networks must not necessarily be created explicitly, but can be treated implicitly by lists of labels. However, the number of nodes gives an upper bound on the computational time of the algorithm.

The shortest path in the new expanded network does not contain cycles since time is increasing on every arc. However, a node (in the original network) might still be visited more than once, i.e. customers may be serviced more than once on a path. Fortunately the cycles will end sooner or later, because time or capacity runs out. We have implemented the forward dynamic programming (DP) algorithm for the SPPTW by Desrochers (1988). The algorithm has been modified to handle capacity constraints as well as time windows.

Houck et al. (1980) has suggested a method to eliminate 2-cycles (i.e. cycles of the form $... - (i) - (j) - (i) - ...$) but the method cannot easily be generalised to cycles of higher order. We have implemented this 2-cycle elimination, but are forced to accept cycles of higher order. Several strategies to decrease the computational time of the algorithm has been proposed. Desrochers, Desrosiers and Solomon (1992) suggest a method to decrease the width of the time windows and also proposes to use a pulling algorithm instead of the more straight forward reaching algorithm usually used in DP-algorithms for shortest path problems. We have not implemented any of these suggestions, since we believe that they will only lead to minor improvements.

The solution time for the SPPTWCC is extremely dependent on the concrete instance. Computational experience reveals, that the following factors are of importance:

- The size of the network (number of customers)
- The ratio between the vehicle capacity and the demand of the customers.
- The width of the time windows
- The size of the arc costs $\hat{c}_{ij}$

The first three factors are quite obvious, since they determine the number of different feasible paths. If just capacity or time is very constraining only (relatively) few paths will be possible, and (relatively) few labels will be considered. The fourth factor is less obvious, but is due to the fact that only small (generally negative) arc costs can make long paths attractive. If most arc costs are large, most labels corresponding to long paths will be dominated by labels corresponding to short paths.

Recall that arc cost in the subproblem are calculated by the formula $\hat{c}_{ij} = c_{ij} - \lambda_j$. Therefore, large multipliers imply difficult subproblems. Figure 1 illustrates this point. For 4 different SPPTWCC problems constructed from the Solomon problems (Solomon 1987) we have fixed the multipliers at $\underline{\lambda}_0 = (\lambda_0, \lambda_0, ..., \lambda_0)$ and found the computational time of the SPPTWCC-algorithm on a HP-9000/735 computer. The arc $(0, n+1)$ was not included, so any path must service at least one customer. This is a very primitive experiment, since the multipliers generally vary from customer to customer, but it illustrates the general point - that large multipliers make the SPPTWCC much more difficult.

The problem c101.100 has very narrow time windows, c102.100 and c103.100 have wider time windows, and in c104.100 the time windows are hardly constraining. For c101.100 the computational time increases by a factor 5 when λ_0 increases from 0 to 300. For the c104.100 the computational time is increased by a factor 106. This is because, the problems with very wide time windows are especially sensitive to large multipliers, since the number of time feasible paths (possibly including cycles) is virtually infinite.

The optimal size of the multipliers (i.e. the dual solution) is clearly not controllable. However, there may be several courses to the optimal dual solution. It is preferable to solve a number of subproblems with relatively low multipliers rather than solving a series of problems with relatively large multipliers.

In the Dantzig-Wolfe decomposition method the multipliers are not directly controlled, but are determined by the simplex multipliers in the Set Covering master problem. If the "good" paths have not yet been generated, the simplex multipliers will tend to be too high, thus creating difficult subproblem instances. The routes generated will generally service too many customers.

In a Lagrangean relaxation based method, one can choose a starting point with relatively small multiplier values and gradually increase the multipliers to the optimal level. This is likely to create easier instances of the subproblem. In the first iterations paths may service too few customers, but this is much less dangerous (in terms of computational time) than servicing too many.

5 Solution method for the master problem

5.1 Subgradient and bundle methods

The master problem is a concave non-differentiable maximization problem. For a general introduction to non-differentiable optimization, see e.g. Lemaréchal (1989). In non-differentiable optimization it is common to study a minimization problem. Therefore we formulate the master problems as

$$minimize_{\underline{\lambda}}\ h(\underline{\lambda}) = -f(\underline{\lambda})$$

Both subgradient methods and bundle methods use the current iterate $\underline{\lambda}$, a step direction $\underline{d}$, $\underline{d} \in \mathbb{R}^n$, and a step size t, $t \in \mathbb{R}$, to calculate the new iterate $\underline{\lambda}$. If the suffix denotes the iteration number, the master iteration number u consists in finding $\underline{d}_u$ and t_u and setting

$$\underline{\lambda}_{u+1} := \underline{\lambda}_u + t_u \underline{d}_u$$

In subgradient optimization one uses a current subgradient as step direction. Note that the solution of SPPTWCC automatically provides us with a subgradient. The step size must be chosen a priori, i.e. without a line search in the direction $\underline{d}_u$, since a line search may lead to convergence to a non-optimal solution. Several rules for choosing t_u exist, see Shor (1985). A major drawback of the subgradient optimization method is, that it only yields guaranteed convergence to optimum if t_u is a divergent series, or if the optimal objective function value is known a priori (which is obviously not the case in our application). In the first case guaranteed convergence even requires an infinite number of iterations. In practical computation this means that one must often settle for a solution which is not guaranteed to be optimal. Furthermore subgradient optimization does not provide a stopping criterion and no primal solution is obtained.

The advantages of subgradient optimization are, that it is easy to implement and usually gives rapid improvement in the first iterations. Later, the convergence slows down, and for some difficult problems convergence in a reasonable time is not possible.

In a bundle method a convex combination of previously obtained subgradients (the bundle of subgradients) is used to calculate $\underline{d}_u$. This yields a better approximation of the dual function f than exploiting a single subgradient only. We have chosen to use the bundle algorithm of Lemaréchal, Strodiot and Bihain (1981)(for other bundle algorithms see Kiwiel 1985). In this algorithm a line search in the direction $\underline{d}_u$ is performed and a step is taken if $\underline{d}_u$ is a direction of "sufficient" descent. Otherwise a so called null step is taken. In both cases the new obtained subgradient is included in the bundle, and a new step direction is calculated. The algorithm terminates at optimum after a finite number of iterations (Lemaréchal, Strodiot and Bihain (1981)).

One can show (see Lemaréchal 1989) that the direction finding problem is equivalent to finding the minimal element of the so called *ϵ-subdifferential* $\partial_\epsilon f(\lambda)$ which is given by

$$\partial_\epsilon f(\underline{\lambda}) \;=\; \{\underline{g} \in \mathbb{R}^n \mid f(\underline{\lambda}) + \underline{g}^T \cdot (\underline{\lambda}' - \underline{\lambda}) \leq f(\underline{\lambda}') + \epsilon \;\; \forall \underline{\lambda}' \in \mathbb{R}^n\}$$

The ϵ-subdifferential can not be computed, but an inner approximation is given by

$$\{\sum_u \zeta_u \underline{g}_u \mid \sum_u \zeta_i = 1, \; \sum_u \zeta_u p_u \leq \epsilon, \; \zeta_u \geq 0\}$$

where $p_u = f(\underline{\lambda}_v) - f(\underline{\lambda}_u) - \underline{g}_u^T \cdot (\underline{\lambda}_v - \underline{\lambda}_u)$ and v is the current iteration number. One may consider p_u as an estimate of the quality of the approximation of the dual function at $\underline{\lambda}_v$ given by the subgradient obtained in iteration u. The direction finding problem is given by

$$\begin{array}{lll} minimize & \| \sum_u \zeta_u \underline{g}_u \|, & subject\,to \\ & \sum_u \zeta_u & = \; 1 \\ & \sum_u \zeta_u p_u & \leq \; \epsilon \\ & \zeta_u \geq 0 & \qquad\qquad \forall u \end{array}$$

This is a quadratic program and the number of variables is equivalent to the number of subgradients. If the bundle grows large, one may choose to delete some of the subgradients, to keep the size of the quadratic program limited. One may for example delete all subgradients for which ζ_u (approximately) zero if the number of subgradients reaches some threshold.

The search direction d_p is given by $-\sum_u \zeta_u \underline{g}_u$ and it is possible to derive the optimality criterion $\sum_u \zeta_u \underline{g}_u \;=\; \underline{0}$, where $\underline{0} = (0, 0, ..., 0)$. When this condition is satisfied the (dual) function is optimised with an absolute accuracy of ϵ on the function value.

The advantages of the bundle methods are, that they secures reasonably fast convergence to the proven optimum, when one is relatively close to optimum. Far from optimum the performance is relatively poor because the line search often requires several iterations of little value.

5.2 The implemented method

We have implemented a method exploiting the advantages of both subgradient optimization and a bundle methods. First a number of subgradient iterations were performed and then the optimum was found using the bundle method outlined above. We have implemented a very simple version of this idea, which turns out to produce very competitive results in terms of computational time. In Phase I, we executed a fixed number of subgradient iterations using step size $t_u = \alpha\beta^u$, where α and β are parameters. This series is not divergent, but since we are not going to run an infinite number of iterations anyway, this does not matter. The starting point $\underline{\lambda}_0 = (\lambda_0, \lambda_0, ..., \lambda_0)$ was chosen slightly larger than 0. Depending on problem size we used the values stated in table 1.

The parameters are in no way optimised, but the relationship between the number of iterations and β secures that the last step size is reasonably small. The objective is to get close to the optimum in Phase I. Since one cannot easily estimate the distance to optimum, we have chosen parameters which will give a fair chance of getting close to optimum. As Section 7 shows, this sometimes means that Phase I is carried on too long, and other times means that it is ended prematurely.

In Phase II we execute the bundle algorithm from the best (in terms of lower bounds) solution obtained during the subgradient optimization. Phase II ends when the proven optimal solution to the dual problem have been found. The stopping criterion is given by $\| \sum_u \zeta_u \underline{g}_u \| \leq \delta$, where δ is a small tolerance. Whether the solution is feasible to the VRPTW or not, can be determined by inspecting the result of the last solved quadratic program. The optimal weights ζ_u are equivalent to the variables of the Dantzig-Wolfe master problem (except for a factor $|\mathcal{V}|$). This means that one can obtain exactly the same information for branching decisions as by the Dantzig-Wolfe decomposition.

The line search was performed using the algorithm of of Lemaréchal (1981). The quadratic program was solved using the function `dqprog` of the IMSL callable library. This quadratic programming solver is known to be poor compared to other commercial solvers, but was the only one we had access to. At most 50 subgradients were kept in the bundle. If this value was exceeded, the bundle reduction technique described in Lemaréchal, Strodiot and Bihain (1981) was applied.

Compared to the Dantzig-Wolfe decomposition master problem (LP relaxation of Set Covering), our master problem in Phase II (quadratic programming) is more difficult, and each iteration is more time consuming. This is, however, not very important since anyway most of the time is spent solving the subproblem.

6 Branching strategy

Often a large proportion of the integrality gap is explored in the subproblem, i.e. SPPTWCC. In many cases it is not even necessary to apply a branch-and-bound method, when the optimal Lagrangean multipliers have been found. When branching is necessary, we have chosen to branch on the time windows. If we choose to branch on the time window of customer i we separate $[a_i, b_i]$ into two time windows $[a_i^1, b_i^1]$ and $[a_i^2, b_i^2]$ by setting $a_i^1 = a_i$, $b_i^2 = b_i$ and $b_i^1 + 1 = a_i^2$. A simple option, which we used, is to choose a_i^2 as the largest integer not larger than $(a_i + b_i)/2$. More sophisticated strategies are suggested in Gélinas et al. (1994).

Two subproblems are created. If the (fractional) primal solution corresponding to the parent node of the branch and bound tree contains one or more paths servicing customer i in as well the first part of the time window as in the second, then both subproblems are restrictions of the parent problem.

To select the time window to branch on, we determine which customers are serviced more than once on a path used in the optimal primal solution. The first one detected is selected as the customer to branch on. Next branching node is chosen as the one with minimal lower bound (best-first strategy). As starting point for the Lagrangean multipliers in any branching node, we used the optimal multipliers from the parent node. Only Phase II (cf. Section 5) was carried out. It turns out that this simple branching strategy is sufficient to solve the problems considered in Section 7.

7 Numerical results

The benchmark problems we have used are the Solomon benchmark problems (Solomon 1987), which Fisher, Jörnsten and Madsen (1992), Desrochers, Desrosiers and Solomon (1992) , Halse (1992) and others have tried to solve as well. The Solomon problems seem to have been accepted as standard benchmark problems by most researchers working on optimization methods or heuristics for the vehicle routing problem with time windows. We have only considered the 27 problems with clustered customers (denoted the C1 problems), because these problems are known to require very little branching. Since our work has been focused on the calculation of lower bounds and does not contain any new results on branching strategy, we find that these problems provide the best basis for evaluation of our method.

In the C1 problems the geographical data and demand data are those of problem 12 in Christofides, Mingozzi and Toth (1979). Solomon (1987) modified these problems by adding time windows and by changing the capacity of the vehicles. Some of the problems have very narrow time windows (problems c101, c105, c106 and c107) and other have wider time windows (problems c102-c104 and c108-c109). The time windows are generated and centered around the arrival times which were obtained by a 3-optimal cluster-by-cluster routing solution without time windows. This makes the problems in some sense ”easy”, because even problems with narrow time windows have good solutions. Problems with less than 100 customers have been constructed by taking subsets of the corresponding 100 customer problems. The 50 customer problems are created by taking the first 50 customers from the 100 customer problem. The 25 customer problems consists of the first 25 customers in the 100 customer problems.

Of the 27 problems Desrochers, Desrosiers and Solomon (1992) succeeded in solving 21 by Dantzig-Wolfe decomposition. Fisher, Jörnsten and Madsen (1992) solved 12 by the K-tree method and Halse (1992) solved 20 using Variable Splitting. In total optimal solutions for 23 problems were known, when we started this work. Since Fisher, Jörnsten and Madsen (1992) does not report any computational times we will not consider their results for further comparison. As table 2 shows, we have now found the optimal solution to all 27 problems. 25 of the 27 problems can be solved without branching. For the two remaining problems the branch-and-bound strategy of Section 6 was executed. The optimal solution was found within a few branching nodes, cf. table 3.

One should note that the instances considered by Desrochers, Desrosiers and Solomon (1992), Halse (1992) and us are slightly different. Halse considers a problem with a fixed number of vehicles, calculates t_{ij} and c_{ij} with one decimal and uses ordinary rounding. The number vehicles is fixed at the number of clusters, which is also the optimal number of vehicles for all 27 instances. We have considered the problem with free number of vehicles, calculated t_{ij} and c_{ij} with one decimal and used truncation. Desrochers, Desrosiers and Solomon (1992) also solved the problem with free number of vehicles, calculates c_{ij} with one decimal and truncation, but calculates t_{ij} with no decimals and truncation (Desrosiers and Gélinas 1993). This reduces the worst case number of nodes in the expanded network (cf. Section 4) by a factor 10, since t_{ij} takes value from a 10 times smaller set. Potentially the worst case computational time is reduced by a factor 100. However, we doubt that the actual reduction in computational time is very large.

The solutions obtained are in all cases identical. Note however that Desrochers, Desrosiers and Solomon (1992) contains one tiny misprint. Problem c103.50 requires 5 vehicles due to the capacity constraints. Desrochers, Desrosiers and Solomon (1992) reports a solution with 4 vehicles. Halse obtains a slightly different objective value due to the rounding of c_{ij}. In c109.25 Desrochers, Desrosiers and Solomon (1992) must branch because the optimal solution cannot be obtained in the first node of the branching tree. This is because the number of vehicles in this node is fraction, and could have been avoided if a lower bound on the number of vehicles was calculated initially.

Desrochers, Desrosiers and Solomon (1992) used a SUN SPARK 1 computer and coded their algorithm in Fortran. Halse (1992) used a HP 9000-835 computer and coded the algorithm in Pascal. Our algorithms has been coded in the C programming language and executed on a HP 9000-735 computer. The HP 9000-835 is about 10% faster than the SUN SPARK 1 (SPEC Release 1.0 1991). We have executed some of the problems on a HP 9000-835 as well, and the results indicate, that the HP 9000-735 is about 8 times faster. Table 2 shows that in general our computational times are more than 8 and 9 times faster than the results obtained by Halse (1992) and Desrochers, Desrosiers and Solomon (1992) , respectively. The computational times of Halse (1992) are best for 7 of the most easy problems. We do not think this shows very much since I/O-times and other program overheads might dominate when the computational time is only a few seconds. The computational time of Desrochers, Desrosiers and Solomon (1992) is only best for one instance. However the results, compared to ours, are better for large problems than for small problems. We think this is due to inefficiencies in our master iteration method. It is hard to draw any final conclusions on the relative efficiency of the methods compared, but it seems that the method suggested in this paper is very competitive.

Table 4 shows how the cpu-time in the initial node is distributed between the three phases. In the easy cases up to about 80% of the time is spent in Phase I. This indicates that these problems might have been solved faster, if a fewer number of iterations in Phase I was chosen. For the c101.100 problem we tried a method consisting of 100 subgradient iterations (instead of 500 iterations) and the bundle method afterwards. This reduced the cpu-time from 51.44 sec. to 28.33 sec. A similar improvement is probably possible for the other easy problems. On the other hand, Phase I seems to be ended prematurely for the most difficult problems. This suggests that a more flexible stopping criterion in Phase I would improve the results.

Due to the similarity between the benchmark problems it is clear that some of the problems will have identical solutions. For example, this happens in fact for problems c101.100, c102.100, c105.100, c106.100, c107.100, c108.100 and c109.100. The solution to these problems is probably equivalent to the 3-optimal solution used to construct the time windows. Problem c103.100 and in particular problem c104.100 have time windows so wide that it is possible to find a better solution than in the problems mentioned above. However, the computer does obviously not know that the problems are very similar which can also be observed on the differences in CPU-times required. As earlier mentioned, we have tested the algorithm on some widely known benchmark problems among those that are available for the problem class. However, we feel that there is a need to extend the number of "standardized" benchmark problems in the future.

8 Conclusions

We have presented a new optimization method for the VRPTW based on a Lagrangean relaxation of the constraint requiring service of all customers. The method is closely related to previously suggested successful algorithms, which decomposes the problem into a series of Shortest Path Problems with Time Windows and Capacity Constraints.

Theoretically our method produces the same lower bounds on the objective function value as the methods of Desrochers, Desrosiers and Solomon (1992) and Halse (1992), but our method creates relatively easy instances of the subproblem, and therefore we have been able to solve a number of previously unsolved problems, and have obtained very competitive computational times even though many possible improvements of our algorithm exist.

Acknowledgements

We thank the two anonymous referees and the associate editor for a large number of valuable comments which improved the quality of this paper.

References

[1] Christofides, N., A. Mingozzi and P. Toth (1979), The Vehicle Routing Problem, in *Combinatorial Optimization*, eds.: Christofides, N., A. Mingozzi, P. Toth and C. Sandi, John Wiley and Sons, New York, U.S.A.

[2] Desrochers, M. (1988), A Generalized Permanent Labelling Algorithm for the Shortest Path Problem with Time Windows, *INFOR 26*, pp. 191-211.

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

[4] Desrosiers, J., M. Sauvé and F. Soumis (1988), Lagrangean Relaxation Methods for Solving the Minimum Fleet Size Multiple Travelling Salesman Problem with Time Windows, *Management Science 34*, pp. 1005-1022.

[5] Desrosiers, J., Y. Dumas, M.M. Solomon and F. Soumis (1995), Time Constrained Routing and Scheduling, in *Handbooks in Operations Research and Management Science*, vol 8: Network Routing, eds.: Ball, M.O, T.L. Magnanti and G.L. Nemhauser, North-Holland, Amsterdam, The Netherlands.

[6] Desrosiers, J. and E. Gélinas (1993), *Private Communication.*

[7] Dror, M (1994), Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW, *Operations Research 42*, pp. 977-978.

[8] Fisher, M.L. (1994), Optimal Solution of Vehicle Routing Problems using Minimum K-Trees. *Operations Research 42*, pp. 626-642.

[9] Fisher, M.L., K.O. Jörnsten and O.B.G. Madsen (1992), *Vehicle Routing with Time Windows*, Research Report 4C/1991, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, DK-2800 Lyngby, Denmark.

[10] Gélinas, M, M. Desrochers, J. Desrosiers and M.M. Solomon (1994), *A New-Branching Strategy for Time Constrained Routing Problems with Application to Backhauling*, Research Report G-92-42, GERAD, Montréal, to appear in *Annals of Operations Research.*

[11] Geoffrion A.M. (1974), Lagrangean Relaxation and Its Uses in Linear Programming, *Mathematical Programming Study 2*, pp. 82-114.

[12] Guignard, M. and S. Kim (1987), Lagrangean Decomposition: A Model Yielding Stronger Lagrangean Bounds, *Mathematical Programming 39*, pp. 215-228.

[13] Houck Jr., D.J, J.-C. Picard, M. Queyranne and R.R. Vemuganti (1980), The Travelling Salesman Problem as a Constrained Shortest Path Problem: Theory and Computational Experience, *Opsearch 17*, pp. 93-109.

[14] Halse, K. (1992), *Modelling and Solving Complex Vehicle Routing Problems*, Ph.D. Dissertation no. 60, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, DK-2800 Lyngby, Denmark.

[15] Jörnsten, K., O.B.G. Madsen and B. Sørensen (1986), *Exact Solution of the Vehicle Routing Problem with Time Windows*, Research Report 5/1986, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, DK-2800 Lyngby, Denmark.

[16] Jörnsten, K., M. Näsberg and P. Smeds (1985), *Variable Splitting - A new approach to some Mathematical Programming Models*, Report LITH-MAT-85-04, Department of Mathematics, Linköping Institute of Technology, Sweden.

[17] Kiwiel, K.C. (1985), *Methods of Descent for Nondifferentiable Optimization*, Lecture Notes in Mathematics 1133, Springer-Verlag, Berlin, Germany.

[18] Kolen, A.W.J., A.H.G. Rinnooy Kan and H.W.J.M. Trienekens (1987), Vehicle Routing with Time Windows, *Operations Research 35*, pp. 256-273.

[19] Lemaréchal, C., J.J. Strodiot and A. Bihain (1981), On a Bundle Algorithm for Nonsmooth Optimization, *Nonlinear Programming 4*, eds.: Mangasarian, O.L., R.R. Meyer and S.M. Robinson, Academic Press, New York, USA.

[20] Lemaréchal, C. (1981), A view of line searches, in *Optimization and Optimal Control*, eds.: Auslender et al., Lecture Notes in Control and Information Sciences 30, Springer-Verlag, Berlin, Germany.

[21] Lemaréchal, C. (1989), Nondifferentiable Optimization, in *Handbooks in Operations Research and Management Science*, vol 1: Optimization, eds.: Nemhauser, G.L, A.H.G. Rinnooy Kan and M.J Todd, North-Holland, Amsterdam, The Netherlands.

[22] Nemhauser, G.L. and R.A. Wolsey (1988), *Integer and Combinatorial Optimization*, Wiley - Interscience, New York, USA.

[23] Olsen, B. (1988), *Routing with Time Constraints: Exact Solutions*, in Danish, M.Sc. Thesis, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, DK-2800 Lyngby, Denmark.

[24] Shor, N.Z. (1985), *Minimization Methods for Non-Differentiable Functions*, Springer-Verlag, Berlin, Germany, Originally published in Russian in 1979.

[25] Solomon, M (1987), Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints, *Operations Research 35*, 254-265.

[26] *SPEC Release 1.0 (1991) Summary of Results.*

Customers	Iterations	α	β	λ_0
25	100	5.00	0.9500	3.0
50	200	5.00	0.9750	3.0
100	500	5.00	0.9875	3.0

Table 1: Parameter values in Phase I.

Problem	Optimal solution	Vehicles	Branching nodes	Time	Time_{VS}	Time_{DW}	Rel_{VS}	Rel_{DW}
c101.25	191.3	3	1	0.82	6.3	18.6	1.0	2.5
c101.50	362.4	5	1	6.60	46.6	67.1	0.9	1.1
c101.100	827.3	10	1	51.44	616.0	434.5	1.5	0.9
c102.25	190.3	3	1	2.11	35.0	79.7	2.1	4.2
c102.50	361.4	5	1	12.96	289.4	330.3	2.8	2.8
c102.100	827.3	10	1	151.95		1990.8		1.5
c103.25	190.3	3	1	5.95	47.4	134.7	1.0	2.5
c103.50	361.4	5	1	23.87		896.0		4.2
c103.100	826.3	10	1	241.49				
c104.25	186.9	3	1	9.51	151.9	223.9	2.0	2.6
c104.50	358.0	5	10	890.71				
c104.100	822.9	10	1	2270.58				
c105.25	191.3	3	1	0.96	3.6	25.6	0.5	3.0
c105.50	362.4	5	1	7.86	73.1	99.1	1.2	1.4
c105.100	827.3	10	1	60.55	968.1		2.0	
c106.25	191.3	3	1	0.83	3.5	20.7	0.5	2.8
c106.50	362.4	5	1	7.47	46.4	91.3	0.8	1.4
c106.100	827.3	10	1	79.27	4241.6	724.8	6.7	1.0
c107.25	191.3	3	1	0.84	3.9	31.7	0.6	4.2
c107.50	362.4	5	1	7.58	56.4	170.6	0.9	2.5
c107.100	827.3	10	1	63.36	2914.4	1010.4	5.7	1.7
c108.25	191.3	3	1	1.42	6.9	43.1	0.6	3.4
c108.50	362.4	5	1	12.43	158.2	245.6	1.6	2.2
c108.100	827.3	10	1	100.29		1613.6		1.8
c109.25	191.3	3	1	1.91	18.2	585.4	1.2	
c109.50	362.4	5	1	17.52	235.8		1.7	
c109.100	827.3	10	3	367.85				

Table 2: Results on the clustered data from the Solomon benchmark problems (Solomon 1987). Times are given in seconds and include I/O times. Time_{DW} denotes the time reported by Desrochers, Desrosiers and Solomon (1992). Time_{VS} denotes the time reported by Halse (1992). Rel_{VS} and Rel_{DW} denotes the relative performance of the methods, i.e. $\text{Rel}_{VS} = \text{Time}_{VS}/(8\ \text{Time})$ and $\text{Rel}_{DW} = \text{Time}_{DW}/(9\ \text{Time})$.

Problem	Optimal solution	Lower bound in init. node	Branching nodes	Tree depth	Time in init. node	Time after init. node	Total time
c104.50	358.0	357.25	10	5	234.82	655.89	890.71
c109.100	827.3	825.65	3	2	233.09	134.76	367.85

Table 3: The problems solved by Branch-and-Bound. Times in seconds and include I/O times.

	Phase I		Phase II		Total	
Problem	Iter.	Time	Iter.	Time	Iter.	Time
c101.25	100	0.59	24	0.23	124	0.82
c101.50	200	4.48	63	2.12	263	6.60
c101.100	500	41.13	89	10.31	589	51.44
c102.25	100	1.54	25	0.57	125	2.11
c102.50	200	10.10	36	2.86	236	12.96
c102.100	500	111.70	133	40.25	633	151.95
c103.25	100	2.93	77	3.02	177	5.95
c103.50	200	18.63	39	5.25	239	23.88
c103.100	500	216.82	42	24.67	542	241.49
c104.25	100	4.68	73	4.83	173	9.51
c104.50	200	58.39	387	176.43	587	234.82
c104.100	500	502.77	1103	1767.81	1603	2270.58
c105.25	100	0.64	24	0.32	124	0.96
c105.50	200	5.37	63	2.49	263	7.86
c105.100	500	48.55	89	12.00	589	60.55
c106.25	100	0.58	24	0.25	124	0.83
c106.50	200	5.14	63	2.33	263	7.47
c106.100	500	63.90	89	15.37	589	79.27
c107.25	100	0.60	24	0.24	124	0.84
c107.50	200	5.16	63	2.42	263	7.58
c107.100	500	48.98	106	14.38	606	63.36
c108.25	100	0.92	34	0.50	134	1.42
c108.50	200	8.55	63	3.88	263	12.43
c108.100	500	82.55	79	17.74	579	100.29
c109.25	100	1.45	23	0.46	123	1.91
c109.50	200	12.67	54	4.85	254	17.52
c109.100	500	131.43	291	101.66	791	233.09

Table 4: Number of iterations and time in seconds in the two phases in the initial Branch-and-Bound node.

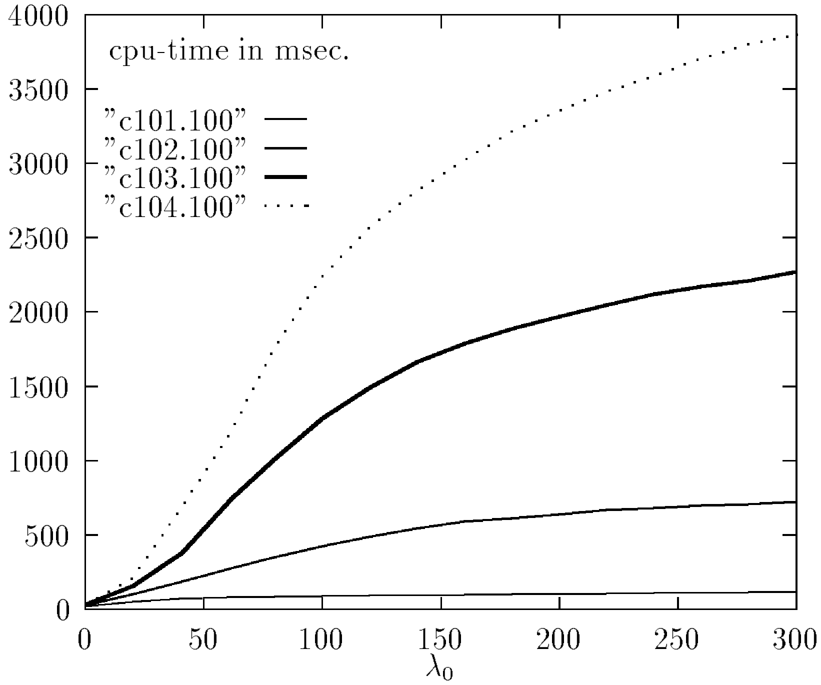

Figure 1: cpu-time in msec. for the SPPTWCC (with 2-cycle elimination) as function of λ_0 (HP 9000-735)