An Optimization-Based Heuristic for Vehicle Routing and Scheduling with Soft Time Window Constraints

YIANNIS A. KOSKOSIDIS

The City College of CUNY, New York, New York, 10031

WARREN B. POWELL

Princeton University, Princeton, New Jersey 08544

MARIUS M. SOLOMON

College of Business Administration, Northeastern University, Boston, Massachusetts 02155

The Vehicle Routing and Scheduling Problem with Time Window constraints is formulated as a mixed integer program, and optimization-based heuristics which extend the cluster-first, route-second algorithm of Fisher and Jaikumar are developed for its solution. We present a new formulation based on the treatment of the time window constraints as soft constraints that can be violated at a cost and we heuristically decompose the problem into an assignment / clustering component and a series of routing and scheduling components. Numerical results based on randomly generated and benchmark problem sets indicate that the algorithm compares favorably to state-of-the-art local insertion and improvement heuristics.

The problem of vehicle routing with time window constraints arises in a variety of applications, including retail distribution, mail and newspaper delivery, municipal waste collection, fuel oil delivery, school bus routing, airline and railroad scheduling, trucking and bargeline fleet scheduling, demand responsive bus systems, and more. The basic elements of the problem are a variable size fleet of vehicles located at a central depot and a set of demand points (customers). Goods need to be transported from the depot to the demand points (or the other way) at minimum cost. Each customer places a fixed size order (shipment) and specifies a time interval within which the service (e.g., delivery) should take place. A vehicle is assigned to each customer to deliver the order. All shipments are assumed to be less than the vehicle capacity (full truckload), and therefore, more than one customer order can be placed on the same vehicle. The problem is to design a complete tour for each vehicle, starting from and ending at the depot, and servicing all the customers within their time window, at minimum routing (transportation) cost.

Most of the early research efforts in the field have concentrated on case studies developing ad hoc procedures for each individual problem.[1, 3, 4, 9, 14, 19, 32, 33, 39] We refer to BODIN et al.[5] for an excellent survey of routing problems and algorithms. More recently SOLOMON[37, 39] designed and analyzed a variety of route construction heuristics, including savings heuristics, a time-oriented nearest neighbor heuristic, insertion heuristics, and a time-oriented sweep heuristic; a sequential time-space insertion algorithm was found to be very successful. BAKER and SCHAFFER[2] have conducted a computational study of route improvement procedures (based on the 2-opt and 3-opt branch exchange procedure of LIN[24]) applied to heuristically generated initial solutions. The results suggest that good quality solutions can be obtained at the expense of computational time requirements.

Optimal approaches for the time constrained

Transportation Science
Vol. 26, No. 2, May 1992

0041-1655/92/2602-0069 $01.25

vehicle routing problem have been proposed by KOLEN et al.[20] who extended the shortest q-path relaxation algorithm of Christofides et al.[8] to the problem with time windows. The largest problem solved to optimality involved 4 vehicles and 14 demand points. Jornsten et al.[16] and MADSEN[26] proposed a promising approach based on variable splitting and Lagrangian relaxation techniques; the approach leads to a generalized assignment problem and a series of shortest path subproblems with time constraints. Computational results were not reported. An extensive review of recent research for the time window constrained vehicle routing problem and closely related problems can be found in the survey by SOLOMON and DESROSIERS[36] and in KOSKOSIDIS.[21]

This paper presents a new methodological approach for the time constrained VRP, based on the Generalized Assignment Heuristic proposed by FISHER and JAIKUMAR.[12] A significant change from the original generalized assignment heuristic is its iterative application where approximate clustering costs are successively improved. We start with a mixed integer programming formulation which treats the time window constraints as "soft" constraints, and develop an optimization based heuristic algorithm for the resulting VRP with Soft Time Window Constraints. The algorithm is tested on randomly generated problems and benchmark problems available in the literature.

1. THE MODEL DEVELOPMENT

WE PRESENT a mixed integer programming formulation for the VRP with Time Window constraints. Let $\mathcal{I}$ be the set of customers $i = 1, \ldots, I$, $\mathcal{K}$ be the set of vehicles $k = 1, \ldots, K$, where the index k denotes also the tour run by vehicle k, and $\mathcal{N}_k$ be the set of customers i serviced by vehicle k, namely $\mathcal{N}_k = \{i / y_{ik} = 1\}$, where $I \equiv \bigcup_k \mathcal{N}_k$. Furthermore, let c_{ij} be the cost of traveling directly from customer i to customer j, q_i be the shipment size of customer i, V_k be the capacity of vehicle k, τ_{ij} be the travel time between customers i and j, s_i be the service time (loading/unloading) at customer i, a_i be the beginning of the time window at i, b_i be the end of the time window at i, and T be a constant larger than the total travel time of any feasible route. Finally, let us define the following variables:

$$x_{ijk} = \begin{cases} 1 & \text{if vehicle } k \text{ travels directly from} \\ & \quad \text{customer } i \text{ to customer } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ik} = \begin{cases} 1 & \text{if vehicle } k \text{ services customer } i \\ 0 & \text{otherwise} \end{cases}$$

t_i = the arrival time at customer i.

The problem can be stated as follows:

(VRP-H)

$$\text{minimize} \quad F(x, y, t) = \sum_{i \subset \mathcal{I}} \sum_{j \subset \mathcal{I}} \sum_{k \subset \mathcal{K}} c_{ij} x_{ijk}$$

$$\text{subject to:} \quad \sum_{i \subset \mathcal{I}} q_i y_{ik} \leqslant V_k \qquad \forall\, k \subset \mathcal{K} \tag{1}$$

$$\sum_{k \subset \mathcal{K}} y_{ik} = \begin{cases} K & i = 0 \\ 1 & i = 1, \ldots, I \end{cases} \tag{2}$$

$$y_{ik} = (0, 1) \qquad \forall\, i \subset \mathcal{I},\ k \subset \mathcal{K} \tag{3}$$

$$\sum_{i \subset \mathcal{I}} x_{ijk} = y_{jk} \qquad \forall\, j \subset \mathcal{I}^0,\ k \subset \mathcal{K} \tag{4}$$

$$\sum_{j \subset \mathcal{I}} x_{ijk} = y_{ik} \qquad \forall\, i \subset \mathcal{I}^0,\ k \subset \mathcal{K} \tag{5}$$

$$x_{ijk} = (0, 1) \qquad \forall\, i, j \subset \mathcal{I}^0,\ k \subset \mathcal{K} \tag{6}$$

$$t_j \geqslant t_i + s_i + \tau_{ij} - (1 - x_{ijk})T \qquad \forall\, i, j \subset \mathcal{I},\ k \subset \mathcal{K} \tag{7}$$

$$a_i \leqslant t_i \leqslant b_i \qquad \forall\, i \subset \mathcal{I} \tag{8}$$

$$t_i \geqslant 0 \qquad \forall\, i \subset \mathcal{I}^0 \tag{9}$$

where $\mathcal{I}^0 = \mathcal{I} \cup \{0\}$.

The objective is to minimize the routing cost subject to vehicle capacity and arrival time feasibility constraints. We can distinguish three groups of constraints in the MIP formulation given above. Constraints (1)–(3) are the assignment/clustering constraints, dealing with the feasible assignment of customer orders to vehicles and dominated by the assignment variables **y**. Capacity constraints (1) ensure that the total freight to be carried by vehicle k is within the capacity of the vehicle. Constraints (2) state that all routes (tours) begin and end at the depot and that each customer i is serviced by one and only one vehicle k.

The second group contains the routing constraints (4)–(6) which include the routing variables **x** and define the itinerary of each vehicle through *all customers* assigned to it.

Finally constraints (7)–(9), incorporating the time variables **t** are the scheduling constraints. The set (7) ensures compatible arrival times between any two consecutive customers i and j, while constraints (8) enforce the service time windows. Note that constraints (7) need to be defined explicitly for the depot, since the arrival time there is always greater than the departure time (unlike the rest of the customers).

Two well-known optimization problems are embedded within the MIP formulation for the problem; the assignment/clustering problem (constraints (1)–(3)) and the time constrained traveling salesman problem (constraints (4)–(9)); this observation is quite significant since it drives our methodology for the solution of the problem (see also FISHER and JAIKUMAR[11, 12]).

The VRP-H, as an extension of the simple VRP where no time constraints are attached, is NP-hard. A medium sized problem could easily involve millions of variables and constraints. CHRISTOFIDES et al.[6, 7] and LENSTRA and RINNOOY KAN[23] have shown that the VRP is NP-hard and by reduction the VRP-H is NP-hard, as well. The findings of SOLOMON[38] and SAVELSBERGH[34] indicated that the time constrained problem is fundamentally more difficult than the simple VRP. As such, one should not expect to obtain optimal solutions within polynomial time; and heuristics have so far offered the most promising results for solving realistic size problems. Heuristic algorithms have been quite successful in dealing with large-scale problems at minimal computational requirements, but they can suffer from serious limitations concerning the quality of the solutions. The tradeoff between the quality of the solution and the computational time requirements motivated the development of an optimization based heuristic for the VRP with Time Windows. The positive experience other researchers have had in the past (Fisher and Jaikumar,[12] CULLEN et al.[10]) is an additional factor justifying the use of this class of algorithms. As MAGNANTI[27] put it, these methods are competitive in running time with most standard heuristics that do not incorporate embedded optimization procedures, e.g., the Clark and Wright savings method, and yet provide better cost solutions in almost all instances.

The optimization based approach we propose is heuristic in the sense that it solves an approximation of the original mathematical model (1)–(9). However, an iterative scheme is employed, which tends to drive the approximation close to the original problem. The solution of the problem is obtained by solving several very well established optimization problems, namely the Capacitated Clustering Problem and a series of time constrained TSPs.

There are three basic steps associated with the development of the procedure. First, the time window constraints (8) are relaxed into the objective function, and then the complicated objective function is approximated using an objective function that is easier to work with. In the following sections we describe these steps in details. Finally, the entire process is run iteratively to improve the solution.

1.1. Turning "Hard" Time Window Constraints to "Soft"

The time window constraints in VRP-H constitute "hard" constraints, which should not be violated in any feasible solution. The first step for the solution of the time constrained VRP involves the relaxation of the constraints (8) into the objective functions in a Lagrangean Relaxation fashion. In this case constraints (8) are perceived as "soft" time window constraints and 100% on-time service (i.e., 100% of the customers are serviced within their time windows) is not required. Introducing the constraints into the objective function, multiplied by the appropriate penalty coefficients, we create a new soft time window model. The result is a variant of the time constrained VRP, namely the Vehicle Routing Problem with Soft Time Window constraints (VRP-S), which can be stated as follows:

(VRP-S) minimize

$$\bar{F}(x, y, t) = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} \bar{c}_{ij} x_{ijk} \quad (10)$$

subject to: (1)–(7), (9) where:

$$\bar{c}_{ij} = w_D d_{ij} + w_T u_{ij} \quad (11)$$

$$u_{ij} = \frac{1}{2}\left[(a_i - t_i)^+ p_e + (t_i - b_i)^+ p_l\right] + \frac{1}{2}\left[(a_j - t_j)^+ p_e + (t_j - b_j)^+ p_l\right] \quad (12)$$

$$(a_i - t_i)^+ = \max\{0, (a_i - t_i)\}$$

$$(t_i - b_i)^+ = \max\{0, (t_i - b_i)\}$$

and d_{ij} is the Euclidean distance between i and j, w_D is the spatial weighting factor, w_T is the temporal weighting factor, p_e is the early delivery penalty, and p_l is the late delivery penalty.

The new composite cost coefficients $\bar{c}$ are weighted sums of the spatial (routing) component and the temporal (scheduling) component of the problem. Term u_{ij} measures the penalty due to the violation of the time windows at the end of each link (i, j), divided by 2 to avoid double counting. Early penalties are imposed, if the arrival time $t_i \leqslant a_i$, i.e. if the term $(a_i - t_i)$ is positive. Similarly, late penalties are imposed when $t_i \geqslant b_i$. If both $a_i \leqslant t_i \leqslant b_i$ and $a_j \leqslant t_j \leqslant b_j$ then no penalties are imposed and $u_{ij} = 0$.

The VRP-S model constitutes a weighted relaxation of the two-sided constraints (8) in the objective function (1) using prefixed multipliers p_e and p_l. As such, the VRP-S is a generalization of the VRS-H model. The two models become equivalent when $w_D = 1$, $w_T = \infty$ and/or $p_e = p_l = \infty$. There are several reasons for selecting the soft time window model:

- The VRP-S model is more general and encompasses the VRS-H model. Furthermore, an algorithm for the soft time window program would be more flexible (due to fewer

constraints) and hence would be less likely to get caught in local optima. If 100% on-time service is required, raising appropriately the penalties should encourage a solution with fewer time window violations.

- In real world time situations windows are usually soft by nature. Typically, there is a limit on the amount any carrier will pay to meet a delivery constraint, although this will depend on the customer. The time penalty coefficients can be adjusted to reflect the situation. High penalties are attached to customers with strict delivery time requirements, while customers who can bear time window violations receive small penalty coefficients.
- The VRP-S model can be very useful in a strategic planning level where tradeoffs between the size, the operating costs of the fleet, and the level of service need to be considered. By adjusting the weight coefficients w_D and w_T we can assign priority to minimize the routing cost (which is usually obtained with a small fleet and high vehicle utilization) or the service violation penalty costs (which usually requires a large fleet and few customers per vehicle).
- Finally, the VRP-S model is capable of finding solutions in cases where a hard time window formulation would have failed. Problems with tight time windows and a small fleet might not have a solution satisfying all customers on-time. In this case, the VRP-S model would yield a solution where some of the customers would not be serviced on time. Naturally, the solution would be infeasible for the hard time window model but the user would, at least, have a solution at hand. The solution can be either accepted as is, or it can be improved by adjusting the appropriate time windows to produce routes to service more or all the customers on-time. The VRP-S solution would provide ample information on the customers causing the infeasibility in the schedule; the "trouble makers" would be easily identified from the routes at hand.

1.2. An Approximation Model for the VRP-S

The basic difficulty in solving the VRP-S stems from the complicated objective function $\bar{F}(x, y, t)$. The cost coefficients $\bar{\mathbf{c}}$ are non-linear functions of $\mathbf{t}$, the routing variables $\mathbf{x}$ are functions of $\mathbf{y}$ and the $\mathbf{t}$ variables are functions of $\mathbf{x}$; in addition, $\mathbf{x}$ and $\mathbf{y}$ are integer. The objective function is non-linear, discontinuous and non-differentiable, and as a result of the integrality constraints, nonconvex; furthermore, the routing and scheduling constraints are non-linear, due to the binary nature of the assignment and routing variables. Fortunately, the mathematical model has a hierarchical structure, which enables us to break the problem into individual subproblems and then solve each subproblem separately.

Extending the analysis by Fisher and Jaikumar[12] for the VRP, we can write the VRP-S as a nonlinear generalized assignment model, namely:

(NGAP) minimize

$$\bar{F}(x(y), t(x(y))) = \Sigma_{k \subset \mathcal{K}} f(y_k) \qquad (13)$$

subject to: (1)–(3).

The function $f(y_k)$ is defined for each vehicle k as:

(TSP-S)$_k$

$$f(y_k) = \text{minimize} \quad \Sigma_{i \subset \mathcal{N}_k} \Sigma_{j \subset \mathcal{N}_k} \bar{c}_{ij} x_{ijk} \qquad (14)$$

$$\text{subject to:} \quad \Sigma_{i \subset \mathcal{N}_k} x_{ijk} = 1 \qquad \forall\, j \subset \mathcal{N}_k \qquad (15)$$

$$\Sigma_{j \subset \mathcal{N}_k} x_{ijk} = 1 \qquad \forall\, i \subset \mathcal{N}_k \qquad (16)$$

$$x_{ijk} = (0,1) \qquad \forall\, i, j \subset \mathcal{N}_k,\ k \subset \mathcal{K} \qquad (17)$$

$$t_j \geq t_i + s_i + \tau_{ij} - (1 - x_{ijk})T \qquad \forall\, i, j \subset \mathcal{N}_k \qquad (18)$$

$$t_i \geq 0 \qquad \forall\, i \subset \mathcal{N}_k \qquad (19)$$

where $\mathcal{N}_k = \{i / y_{ik} = 1\}$ is the set of customers assigned to vehicle k through (13), (1)–(3).

Clearly the TSP-S model defines a Traveling Salesman Problem with time window constraints over the customers assigned to vehicle k. The optimal assignment of customers to vehicles, in other words, the optimal partitioning of $\mathcal{I}$ into K sets $\mathcal{N}_k$, is performed through NGAP. However, in order to solve NGAP the optimal values of the K functions $f(y_k)$ are needed, which are obtained through the K TSP-S's. The master-slave relation between NGAP and the TSP-S's can be approached using an iterative approximation procedure. The approach is based on constructing a linear approximation of the objective function (13). Note that NGAP contains only assignment variables $\mathbf{y}$ in the constraint sets, while the $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{t}$, variables are present directly or indirectly in the objective function. If we decouple the $\mathbf{x}$ and $\mathbf{t}$ variables from the objective function then the partitioning problem could be easily solved through state-of-the-art methods. This motivated the use of an approximate objective function, which minimizes the assignment cost instead of the routing cost, and it is justified if we consider the approximate cost coefficients $\hat{\mathbf{c}}$ as approximate derivatives of $\bar{\mathbf{F}}(\cdot)$. Then, NGAP turns out to be the well studied Generalized Assignment

Problem (Fisher and Jaikumar,[12] FISHER et al.[13]), namely:

$$\text{(GAP)} \quad \text{minimize} \quad \hat{F}(y) = \sum_{k \subset K} \sum_{i \subset I} \hat{c}_{ik} y_{ik} \tag{20}$$

subject to: (1)–(3) where $\hat{c}_{ik}$ expresses the approximate cost of assigning customer i to vehicle k.

The solution of GAP defines a feasible assignment of customers to vehicles in terms of the capacity constraints. Model GAP has a well defined linear objective function and state-of-the-art algorithms exist for its solution. We will not dwell on this point, since in the place of GAP we solve the Capacitated Clustering Problem (CCP). The method for approximating the function $\bar{F}(x, y, t)$ and determining the approximate cost matrix $\hat{\mathbf{c}}$ is described in Section 3.

The procedure for the solution of the VRP-S model iterates between solving the assignment problem defined by the GAP/CCP and a series of routing and scheduling problems defined by the TSP-S's. The proposed algorithm starts with some initial approximate cost matrix $\hat{\mathbf{c}}^0$ and, based on that, partitions the set $\mathcal{I}$ into K clusters and produces a set of routes satisfying the constraints (1)–(7), (9). These routes are used to update the approximate cost matrix necessary for the next iteration. The algorithm iterates producing customer clusters and tours, and updating the cost matrix, until no more improvements can be found in the produced tours or until the maximum allowable number of iterations has been exceeded.

The resulting solution to the VRP-S is expected to be suboptimal, since it is based on a suboptimal partitioning. However, the resulting tours can reveal information useful to better approximate the objective function (20) at subsequent iterations, through an updating mechanism we describe later. The new approximation is used to resolve the assignment problem and produce new tours which are aimed to be closer to the optimal solution of the VRP-S. The algorithm is outlined in the following:

An Iterative Approximation Algorithm for the VRP-S

Step 0. Initialization.
Set the iteration counter $n = 0$.
Initialize the approximate cost matrix $\hat{\mathbf{c}}^0$.
Go to step 2.

Step 1. Cost Update.
Set $n = n + 1$.
Update the approximate cost matrix: $\hat{\mathbf{c}}^n = f(\hat{\mathbf{c}}^{n-1})$.

Step 2. Clustering.
Solve the Capacitated Clustering Problem.
Let: $\mathcal{N}^n = \{\mathcal{N}_1^n, \ldots, \mathcal{N}_k^n, \ldots, \mathcal{N}_K^n\}$ be the partition of $\mathcal{I}$ at the nth iteration.

Step 3. Routing and Scheduling.
Solve the Traveling Salesman Problem with Soft Time Window constraints for each cluster $\mathcal{N}_k$.
Let: $\mathcal{T}^n = \{\mathcal{T}_1^n, \ldots, \mathcal{T}_k^n, \ldots, \mathcal{T}_K^n\}$ be the set of routes produced in the nth iteration, satisfying (1)–(7), (9), and $f(y_k^n)$ be the cost of the tour T_k^n.
Set: $\bar{F}(x^n, y^n, t^n) = \sum_{k \subset K} f(y_k^n)$.

Step 4. Termination check.
If $\bar{F}(\cdot)$ does not improve for some number of iterations or if $n > n_{\max}$ STOP.
Otherwise go to Step 1.

At the initialization step the algorithm constructs the initial approximate cost matrix $\hat{\mathbf{c}}^0$ using some preliminary notion about the nature of the costs; details are given in Section 3.

Step 1 corresponds to determining the approximate derivatives of the objective function $\bar{F}(\cdot)$, which are necessary for the solution of the approximate function $\hat{F}(\cdot)$. The information on the obtained solution, conveyed through $\mathcal{T}$, are used here to update the cost matrix $\hat{\mathbf{c}}$ to better approximate (10).

Step 2 deals with the optimal assignment/clustering of customers to vehicles. The problem is to minimize the assignment/clustering cost in (20) subject to vehicle capacity constraints. What this does, in essence, is to use the approximate costs $\hat{\mathbf{c}}$ to decouple the VRP-S. The solution of the clustering problem fixes the assignment variables $\mathbf{y}$ independently of the $\mathbf{x}$ and $\mathbf{t}$ variables. The result is K individual mutually exclusive and collectively exhaustive clusters.

Having fixed the $\mathbf{y}$ variables, all we need to do next is optimize the itinerary of each vehicle k assigned to service cluster $\mathcal{N}_k$. Since the vehicle capacity feasibility is guaranteed from step 2, the problem reduces to solving K TSPs with soft time window constraints. The objective is to minimize the routing cost and the time penalties for each $\mathcal{N}_k$, namely:

$$\text{(TSP-S)}_k \quad f(y_k) = \text{minimize} \sum_{i \subset \mathcal{N}_k} \sum_{j \subset \mathcal{N}_k} \bar{c}_{ij} x_{ijk}$$

subject to: (15)–(19).

The solution of the TSP-S's yields a set $\mathcal{T}$ of K tours $\mathcal{T}_k$ starting from and terminating at the depot, which service all customers $i \subset \mathcal{I}$. Each tour $\mathcal{T}_k$ defines a feasible itinerary of vehicle k over the customers in $\mathcal{N}_k$. The set $\mathcal{T}$ defines a feasible solution for the VRP-S.

The algorithm terminates when the objective function (20) has not improved significantly for a given number of iterations or when some maximum number of iterations has been reached.

The procedure shares several attractive features with the original Fisher and Jaikumar[12] algorithm for the VRP:

- The decomposition of the VRP-S yields problems which are easier to solve, compared to the VRP-H, and have been fairly well studied in the operations research literature.
- The algorithm has the capability of producing feasible solutions (with respect to vehicle capacities) at each iteration, a feature which is not found in tour construction algorithms. This allows the user to terminate the algorithm at any desirable point and still have a feasible solution at hand. Furthermore, by devoting additional computational time the solution can be improved in most instances, by sharpening the initial approximation through the iterative procedure we employ. To the contrary, most of the existing heuristics (including [12]) are one-shot affairs and whatever solution is determined after one application of the algorithm must be accepted.
- The algorithm preserves a global view of the optimization problem solved, attained at the clustering phase. The assignment of customers to vehicles is made in light of any other possible assignment, through the optimization problem solved at the assignment/clustering level. This avoids a problem faced by sequential assignment or limited adjustment heuristics that can "paint themselves into a corner" by unknowingly making initial assignments that eventually force very expensive ones in order to maintain feasibility.
- The algorithm imitates the natural steps followed by vehicle dispatchers and goes even further. Dispatchers, when faced with pure spatial problems, will first locate cohesive groups of customers and then route the vehicles through those groups. The problem becomes intractable to the human, as soon as the time dimension is introduced. Obviously, this is no obstacle for the computerized algorithm.
- The algorithm lends itself to parametric analysis, concerning tradeoffs between the size of the fleet of vehicles, the operating costs and the level of service offered. This kind of analysis would be useful in a strategic planning level when evaluating, for example, vehicle acquisition and territorial expansion decisions.

2. SOLVING THE SUBPROBLEMS

THE DECOMPOSITION of the VRP-S model results in the assignment/clustering problem (20) and k TSPs (14). The solution of these subproblems in the context of the VRP-S is discussed next.

2.1. The Capacitated Clustering Problem

The solution of the assignment/clustering problem is obtained by solving a Capacitated Clustering Problem (CCP). Fisher and Jaikumar[12] solve a Generalized Assignment Problem in the corresponding phase of their algorithm for the VRP, which requires the external specification of "seed" customers that become approximate centroids of each cluster. To the contrary, when solving the CCP the selection of the seeds is imbedded within the optimization procedure.

The CCP is a special case of the facility location problem and therefore it is closely related to the generalized assignment and the p-median problem. As such, it has been shown to be NP-complete (MULVEY AND BECK[30]). Heuristic and approximation procedures are, therefore, the only practical techniques for handling large problems. We refer the interested reader to Mulvey and Beck,[30] MULVEY AND CROWDER,[29] Koskosidis[21] and their references for the mathematical formulation and analytic description of the CCP model.

We present here a heuristic iterative algorithm based on the Mulvey and Beck[30] primal heuristic algorithm for the CCP. Our algorithm consists of four basic steps, which are briefly described next:

Phase 0: Initialization. Select K initial seed customers. Let $\mathcal{J}^*$ be the set of seed customers.

Phase I: The greedy assignment. Given the set of seeds $\mathcal{J}^*$ the non-seed customers are ranked and assigned to seeds according to their regret function:

$$\text{REGRET}(i) = \hat{c}_{im'} - \hat{c}_{im} \qquad (21)$$

where m is the closest seed to i and m' the second closest seed. The regret function represents the "penalty" of assigning customer i to its second closest seed instead of the closest seed. Customers are ordered in decreasing order of regret and assigned to the closest available seed. The assignment of customers with the largest regret function values is undertaken first to avoid assigning a customer to a very "distant" second or kth closest seed. At the end of the assignment phase all customers have been assigned to a seed and all K clusters have been formed.

Phase II: The median seed relocation. For each cluster k with seed j, we seek to improve the

internal clustering cost defined as:

$$\hat{C}_{j_k} = \sum_{i \subset \mathcal{N}_k} \hat{c}_{ij_k} \qquad j \subset \mathcal{N}_k \tag{22}$$

by identifying the customer which, as a seed, generates the minimum cost "hub-and-spoke" spanning tree; this is the customer $l \subset \mathcal{N}_k$, such that:

$$\hat{C}_{l_k} = \min\{\hat{C}_{m_k},\ m \subset \mathcal{N}_k\} \tag{23}$$

If new seeds have been found for one or more clusters the algorithm iterates between phases I and II until no more improvements are possible. Then the algorithm proceeds with phase III.

Phase III: The local exchanges. In this phase all the possible combinations of customer pairs belonging to different clusters are considered for exchange. Vehicle capacities permitting the profitable exchanges are identified, ranked in terms of savings in clustering cost and materialize. If one or more local exchanges are performed the algorithm returns to phase II and so on. The algorithm terminates either when no more improvements can be identified in phases II and III or when a prespecified number of iterations has been reached.

Concerning the seed initialization procedure several different methods have been proposed over the years, including human expertise, random seed selection (Mulvey and Beck[30]), geometrical and shipment size-related criteria (Fisher and Jaikumar[12]). In an effort to account for both the spatial and the temporal aspects of the our cost function and to use more information from the problem we have developed two methods for the selection of the initial seeds (see Koskosidis[21] for a discussion of a third method, the Bayesian updating procedure). The idea is to keep these algorithms simple, and avoid overloading the VRP-S model with computational requirements disproportional to the significance of the initialization phase. At the same time the algorithms should be sophisticated enough to provide the CCP model with a reasonably good starting solution. A bad initial set of seeds could trigger a lengthy clustering procedure and affect the quality of the final clustering solution. These procedures are based in solving some form of a knapsack problem and are described in detail in Koskosidis.[21]

The iterative heuristic described is based on the Mulvey and Beck [30] primal heuristic algorithm for the CCP. However, several modifications and extensions have been introduced which increase the efficiency of the algorithm. First, we develop special algorithms for the seed selection phase in order to avoid using randomly generated seeds. Second, the CCP algorithm employs an iterative, self-correcting scheme, that improves the initial set of seeds and the existing clusters, while iterating between the assignment, seed relocation and local improvement phases. To the contrary, the Mulvey and Beck (M&B) heuristic terminates after the local switch phase and restarts with a new set of seeds, making no use of the information generated before.

The evaluation of the performance of the iterative CCP algorithm has been based on the development of tight lower bounds through Lagrangean relaxation techniques. Several relaxations have been developed, but the relaxation of the unique assignment constraints (2) has been the most successful in providing tight lower bounds. The update of the Lagrangean multipliers has been based on a typical subgradient optimization scheme.

Five sets of randomly generated problems (including 76 problems in total) have been used to evaluate the performance of the iterative CCP algorithm and to compare it with the M&B heuristic. Detailed results and comparisons can be found in Koskosidis[21] and KOSKOSIDIS et al.[22] In summary, the iterative heuristic we have developed has been able to find the optimal solution for 45 out of the 76 problems tested. The solutions have been within 1% from the lower bound for 61 out of the 76 problems and for only 2 problems was the solution more than 3% from the lower bound. Compared to the M&B heuristic it managed to provide systematically better quality solutions, while the computational time requirements have been consistently lower.

2.2. The TSP with Soft Time Windows

Once the clusters have been formed we solve a series of time constrained TSPs, as stated in (14)–(19). The scheduling constraints of TSP-S make the model equivalent to an asymmetric TSP (Koskosidis[21]), but at the same time they are used as subtour elimination constraints (Solomon,[37] MILLER et al.[28]). Karp[17, 18] and PAPADIMITRIOU and STEIGLITZ[31] established that the asymmetric TSP is NP-complete; as a result, considerable effort has been devoted to develop good heuristics for this class of problems. We have developed two approaches for the solution of the TSP-S. The first one combines a total enumeration algorithm with a reduced gradient scheduling algorithm. The second is a heuristic procedure based on the branch exchange algorithm by LIN and KERNIGHAN.[25]

In the total enumeration approach the TSP-S is solved using a two-tiered procedure. At the first level tours are created based only on the routing variables $\mathbf{x}$ with no regard to the scheduling variables $\mathbf{t}$; at the second level the optimal scheduling

of these tours are sought preserving the customer sequencing obtained in the first level. In principle, all feasible tours (in terms of the routing variables) are constructed in the first level. Then, a reduced gradient-based optimization algorithm is applied to each tour to identify the optimal scheduling of the tour. Since the optimal solution to TSP-S should be feasible in $\mathbf{x}$, and from all tours feasible in $\mathbf{x}$ we choose the lowest cost one in $\mathbf{x}$ and $\mathbf{t}$, the obtained tour is the optimal solution to the TSP-S of the cluster under consideration. The procedure is accelerated by applying weak pruning rules at the first level, aiming at early detection of unrealistic tours, which are consequently abandoned. For a description of the procedure the reader is referred to Koskosidis[21] for details on the total enumeration procedure and the reduced gradient scheduling algorithm.

The total enumeration method would not be effective for larger problems (more than six customers) but was very efficient for problems with a relatively few customers per tour. For larger tours we developed a branch exchange heuristic based on Lin and Kernighan's[25] heuristic for the symmetric TSP. Although the L&K heuristic provides an excellent framework for the asymmetric TSP, several differences introduced by the temporal aspect of the TSP-S created the need to modify and extend the original approach, in order to accommodate the temporal asymmetric nature of our model. First, in order to identify profitable exchanges (i.e., link exchanges that would reduce the total cost of the tour) one needs to know the cost on each link $\bar{c}_{ij}$. Since this is not known in advance the approximate costs $\hat{c}_{ik}$ have been used instead. Then, when it comes to evaluate a candidate tour (one emerging from a link exchange) one has to work with directed tours, since the temporal component of the cost is very likely to depend on orientation. In both cases, the time window constraints are handled implicitly through the composite cost coefficients $\hat{c}_{ik}$. When time windows are violated, this reflects on the cost of the tour under consideration. The branch exchange heuristic for the TSP-S has been based on the modification of a FORTRAN code for the L&K heuristic developed by Frantzeskakis [15]. We refer the interested reader to Koskosidis[21] and Frantzeskakis[15] for more details.

Both solution algorithms for the TSP-S model have been tested and compared using sets of randomly generated test problems with structure similar to that of problem set RP1 (described below). All the problems have a one day planning horizon and time windows with a maximum length of two hours. The comparative results presented here are based on problems with five, seven and ten customers per tour, since the total enumeration approach failed to solved problems with more than ten customers within reasonable time. The results are listed in Table I.

The branch exchange heuristic was able to match the performance of the total enumeration approach in most of the problems tested. It failed to match the total enumeration solutions in four out of the fifteen problems solved, but the worst deviation from these solutions was no more than 4%. In general, since the original L&K heuristic is able to produce high quality solutions, and the branch exchange heuristic we developed follows the same basic structure, we believe that the solutions obtained are of excellent quality. As expected, the CPU times (seconds on an IBM 3081/VM370) increase sharply for the total enumeration approach rising from 0.017 second for the five customer problems to 36.0 seconds for the ten customer problems. To the contrary, the branch exchange heuristic exhibited a much smoother behavior; the average CPU time ranged from 0.025 second for the five customer problems to about 2.9 seconds for twenty customer problems and to less than 9.0 seconds for thirty customer problems (not shown in Table I). Given the performance of the two approaches and their respective CPU time requirements, we have used the total enumeration algorithm for problems with up to five customers per tour and the branch exchange heuristic for larger problems.

3. CALCULATING THE APPROXIMATE COSTS

The approximate cost coefficients $\hat{\mathbf{c}}$ represent the cost of including customer i into the cluster serviced by vehicle k, which depends on the other customers included in the same tour. During the clustering phase there is very little information available on the customers that has been or will be included in each cluster. Furthermore, the effect of including a customer in a given cluster is not known until later when the routing and scheduling problem for each cluster is solved. Therefore, in order to calculate th approximate costs, the algorithm starts with some initial cost matrix $\hat{\mathbf{c}}^0$ and proceeds to the clustering, routing and scheduling phases. Once a set of feasible tours $\mathcal{T}^0$ has been formed, the cost matrix $\hat{\mathbf{c}}$ is updated using information developed previously. The tours are evaluated to reveal information on how successful the current clustering configuration of customers has been and try to identify alternative potentially successful configurations. We describe next the initialization

TABLE I

Solution Results for the TSP-S

		Total Enumeration		Branch Exchange	
n	Problem	COST	CPU secs	COST	CPU secs
5	1	882	0.017	882	0.023
	2	902	0.017	902	0.027
	3	876	0.017	876	0.022
	4	949	0.016	989	0.030
	5	548	0.017	548	0.024
7	1	953	0.493	953	0.102
	2	1037	0.467	1076	0.098
	3	928	0.505	928	0.096
	4	1107	0.487	1107	0.109
	5	628	0.512	628	0.101
10	1	992	36.06	1014	0.216
	2	1101	35.35	1109	0.225
	3	998	37.00	998	0.203
	4	1194	36.24	1194	0.212
	5	789	36.17	789	0.219

algorithm and three update algorithms for the approximate cost matrix $\hat{\mathbf{c}}$.

3.1. The Initial Approximate Cost Matrix

During the cost initialization procedure we develop the cost matrix $\hat{\mathbf{c}}^0$, which gives the initial clustering costs between any pair of customers. The direct cost and the cost averaging update schemes share the same cost initialization algorithm. The basic idea behind the calculation of the initial costs $\hat{c}^0_{ij_k}$ is to consider the single vertex tour through j (*depot–j–depot*) and then calculate the cost of inserting i into this tour in the least expensive way. The resulting tour would be either *depot–i–j–depot* or *depot–j–i–depot*. The difference between the single vertex tour and the two vertex tour is the cost of associating customer i with seed j, namely:

$$\hat{c}_{ij} = \min\big[w_D(d_{0i} + d_{ij} + d_{j0}) + w_T(u_{0i} + u_{ij} + u_{j0}), w_D(d_{0j} + d_{ji} + d_{i0}) + w_T(u_{0j} + u_{ji} + u_{i0})\big] - (d_{0j} + d_{j0}).$$

An example is shown in Figure 1 for customer i and seed j.

The x-axis represents time and the horizontal line segments represent the length and positioning of the time window for each customer. The vertical distance between the customers is a measure of the distance between them. The single vertex tour through j is shown with the solid line. The dashed line shows the two-customer tour *depot–i–j–depot*. The best place to insert i into the tour of j, given the size and the positioning of the time windows, is in the leg from the depot to j. The resulting tour

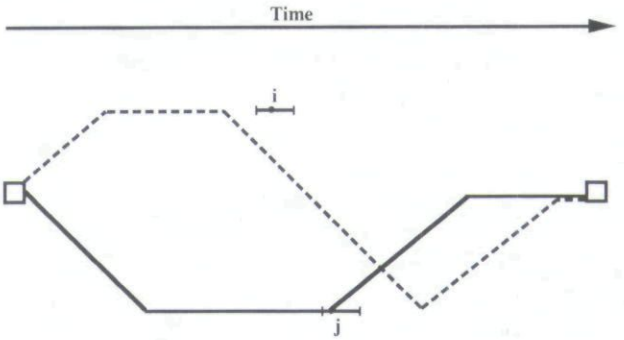

Fig. 1. Insertion of customer i into one-vertex tour.

forces the vehicle to miss the time window at j. If the delay is, for example, 5 time periods, then:

$$\hat{c}^0_{ij_k} = w_D \times (d_{0i} + d_{ij} + d_{j0} - (d_{0j} + d_{j0})) + w_T \times 5p_l$$

Alternatively, the vehicle could have delivered at i 5 time periods earlier, catching j's time window right on b_j, resulting in time penalties equal to $5p_e$. If $p_e < p_l$ the second scenario is more attractive. In general, the cost initialization algorithm will insert i in the leg resulting in the least cost $\hat{c}_{ij_k}$.

This procedure is performed for all pairs of customers (i, j), constructing the matrix $\hat{\mathbf{c}}^0$. Note that $\hat{\mathbf{c}}^0$ will be asymmetric since, in general $\hat{c}_{ij_k} \neq \hat{c}_{j_k i}$.

3.2. The Direct Cost Update Procedure

The update of the approximate cost matrix $\hat{\mathbf{c}}$ at any subsequent iteration follows the same principals with the initialization procedure, estimating the insertion cost of customers into potential clusters. The difference is that the customers are now inserted into existing tours and not the artificial one vertex tours. We distinguish between two cases depending on whether customer i is a member of the tour under consideration or not.

Case a. $i \subset \mathcal{N}_k$, $\mathcal{N}_k = \{l / y_{lk} = 1,\ l \subset \mathcal{I}\}$.

In this case customer i is a member of the tour k under consideration. The cost of keeping i in $\mathcal{N}_k$ is the difference in the costs of tour k with and without i, namely:

$$\hat{c}^{n+1}_{ik} = f(y^n_k) - f'(y^n_k) \tag{24}$$

where $f(y^n_k)$ is the cost of the TSP-S tour (14), (15)–(19) through the customers in $\mathcal{N}_k$ during iteration n, and $f'(y^n_k)$ is the cost of the tour through $\mathcal{N}_k - \{i\}$, keeping the sequence of the remaining customers as it was in $f(y^n_k)$.

Case b. $i \not\subset \mathcal{N}_k$, $\mathcal{N}_k = \{l / y_{lk} = 1,\ l \subset \mathcal{I}\}$.

In this case we want to insert the customer i in the tour k and measure the additional burden imposed on vehicle k, namely:

$$\hat{c}_{ik}^{n+1} = f'(y_k^n) - f(y_k^n) \tag{25}$$

where $f(y_k^n)$ is as before and $f'(y_k^n)$ is the cost of the augmented tour over the customers in $\mathcal{N}_k \cup \{i\}$, holding fixed the sequence of the rest of the customers in $\mathcal{N}_k$. Customer i is inserted in the appropriate leg to minimize the cost of the augmented tour.

After calculating the cost $\hat{c}_{ik}^{n+1}$, the cost of associating customer i with any other customer l in cluster k would be:

$$\hat{c}_{il}^{n+1} = \hat{c}^{n+1} \qquad \forall\, l \subset \mathcal{N}_k \tag{26}$$

Note that in both cases we reevaluate the cost of the tour after adding or removing customer i, without reoptimizing the itinerary of the vehicle, in order to avoid resolving the TSP-S for $\mathcal{N}_k - \{i\}$ or $\mathcal{N}_k \cup \{i\}$. The arrival times are readjusted using a reduced gradient-based scheduling procedure (named SCHED see [21]), to minimize the time penalties. The new tour may be suboptimal, but that had to be accepted in order to reduce the computational burden. Since the costs $\hat{\mathbf{c}}$ are approximate anyway, it was felt that 100% accuracy was not justified, since the attempt to calculate those costs exactly would be computationally prohibitive.

An example is shown in Figure 2a with a nine customer (plus depot), 2-vehicle problem. The solution at iteration n is the two tours shown as solid lines (Tour 1: 0-1-2-3-4-5-0 and Tour 2: 0-9-8-7-6-0).

The cost of including customer 5 into tour 2 would be:

$$\hat{c}_{52}^{n+1} = f'(y_2^n) - f(y_2^n)$$

where $f'(y_2^n)$ is the cost of Tour 2′: 0-9-8-7-5-6-0. Note that in addition to the change in routing, we also had to reschedule a portion of the tour to optimize the service penalties. Thus, inserting customer 5 in tour 2 required an earlier departure from customer 8. Customer 5 is delivered very early so that customer 6 is not served too late.

Clearly, in this example the best place to insert 5 in tour 2 is in the leg (7,6), shown in Figure 2b. This choice is not necessarily obvious, and a search is required to determine the best leg in which to perform an insertion.

3.3. The Average Cost Update Procedure

The cost averaging update scheme is identical to the direct cost update scheme with the exception of the calculation of the costs $\hat{c}_{ij_k}$. While in the direct update scheme the clustering costs $\hat{c}_{il}^{n+1}$ for customer i with all customers $l \subset \mathcal{N}_k$ are identical and equal to $\hat{c}_{ik}$, in the averaging scheme the costs are the average over all previous iterations, namely:

$$\hat{c}_{il}^{n+1} = \hat{c}_{il}^{n} + \frac{1}{n}\left(\hat{c}_{ik}^{n+1} - \hat{c}_{il}^{n}\right) = \frac{n-1}{n}\hat{c}_{il}^{n} + \frac{1}{n}\hat{c}_{ik}^{n+1} \tag{27}$$

where $\hat{c}_{ik}^{n+1}$ is given from equation 24 or 25, respectively.

This way the approximate costs at a given iteration are standard convex combinations of the costs at all previous iterations and the algorithm is allowed to use more information from the past iterations. In the cost averaging scheme the update is based on the tours produced in all previous tours and not just the last one, as is the case with the direct cost update scheme. The purpose of this procedure is to stabilize the algorithm from one iteration to the next.

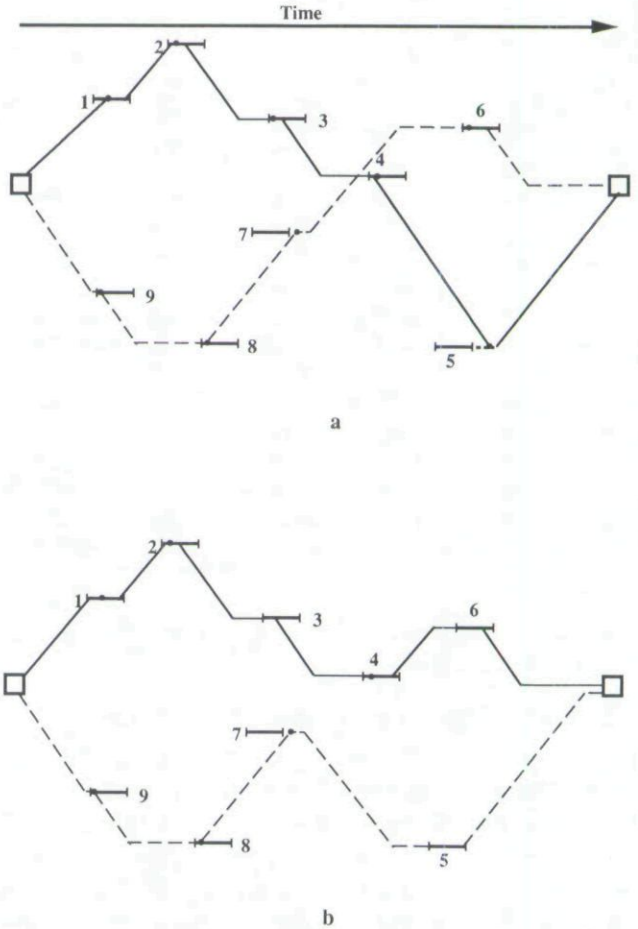

Fig. 2. Cluster improvement through approximate insertion costs.

4. COMPUTATIONAL STUDY

WE HAVE performed two sets of computational experiments. The first, based on data we have randomly generated, is intended to compare the cost update algorithms and investigate the efficiency of the iterative scheme we used for the solution of the VRP-S model. The second set of experiments has been based on benchmark problems available in the literature and aimed to compare the VRP-S to state-of-the-art heuristic algorithms.

4.1. Randomly Generated Data Sets

We have generated five sets of random problems, consisting of five problems per set. Each set has its own distinct characteristics and all problems in the same set have been created using identical parameters but different seeds for the random number generator. The key characteristics of the problems are *the number of customers* to be serviced, *the size of the service area*, the number of vehicles to be used, *the size of shipment* delivered to each customer, *the planning horizon* which defines the time points (dates) within which the delivery dates should be placed, *the delivery date* which defines the target date for the delivery of the shipment to the customer, and *the length of the time window* which in coordination with the delivery date defines the time interval for no penalty delivery. The time windows are centered around the delivery dates. The number of available vehicles should be at least as big as the total volume of the shipments delivered, expressed in truckloads. It is variable and it is initially set by the user. However, it can be increased by the model, as needed, during the clustering phase, if the number of vehicles is not adequate.

The random problem generation procedure is based on drawing pseudo-random numbers from a (0, 1) uniform distribution; the procedure is as follows. First, the user defines the desired number of customers, the size of the service region, the maximum allowed shipment size, the length of the planning horizon and the maximum allowed time window length. Next, the computer generates random pairs of numbers, as many as the desired number of customers, each one representing x- and y-coordinates for the customers. These pairs should fall within the dimensions of the service area. For each customer the computer generates a random demand (shipment size) bounded by the maximum demand, a random delivery date within the planning horizon, and a randomly long time window. Special care is taken to ensure that duplicate demand points are not generated.

Each problem includes 50 customers; problems of this size are large enough to explore the capabilities of the model without imposing excessive computational requirements. We generated problems with short, medium and long planning horizons, with emphasis on medium sized ones. Problem set RP1 has a short planning horizon (one day) and time windows which can be at most two hours, which result in a problem with tight time window constraints. Problem sets RP2, RP3, and RP4 have medium planning horizons (5 days) and time windows with maximum length 4, 24, and 48 hours respectively. Finally, problem set RP5 has a long planning horizon (14 days) and time windows with maximum length 48 hours, generating problems with loose time constraints. In general, the parameters for the generation of the random problems have been chosen such that the problems would represent a variety of real world situations ranging from tightly constrained daily deliveries (e.g., perishable goods), to loosely constrained problems. We refer the interested reader to [21] for more details on the random problem generation and the problems themselves which are available from the authors.

Each problem has been solved using both of the cost update methods. The results are shown in the following tables. Table II compares the cost update algorithms in terms of the quality of the solution obtained and the CPU time requirements. It lists average solution values for each one of the five problem sets. For each algorithm the first column lists the average values of the initial total cost (routing cost plus time penalty cost), the total cost of the best solution found, and the improvement between the initial and the best solution as a percentage of the first one. The second column lists the average percentage of customers that are not serviced within their time window in the best solution found, and the third column lists the CPU time requirements in seconds for 100 iterations of the VRP-S model on an IBM 3081/VM370. Table III lists for each of the cost update algorithms the number of problems that the corresponding algorithm found the lowest total cost solution and the average percentage above the best solution found. Finally, Table IV lists, for a given number of iterations n, the average percentage of the best solution found within n iterations above the best solution.

Since the optimal solutions or any bounds for the random problems are not available, the performance evaluation of the VRP-S model is based on the relative improvement of the final solution obtained compared to the initial solution. Recall that the initial solution is based on the Fisher and Jaikumar[12] one-pass insertion logic and the

TABLE II

Comparison of the Cost Update Algorithms in Terms of (a) Average Solution Values, (b) Average % of Violated Windows, and (c) Average CPU Time Requirements

	Direct Cost			Average Cost		
Problem Set	Initial Best % imp.	% of viol. wind.	CPU secs	Initial Best % imp.	% of viol. wind.	CPU secs
RP1	7282 5828 19.1	29.2	63.3	7282 6037 16.4	31.2	63.1
RP2	17576 12892 26.1	14.0	102.7	17567 13270 24.2	13.6	104.5
RP3	13653 11494 15.5	3.2	109.0	13653 11558 15.1	6.4	108.0
RP4	12820 11189 12.2	3.6	106.8	12820 11297 11.5	4.0	103.7
RP5	15483 11773 21.9	0.8	109.0	15483 11699 22.5	2.0	109.0

CPU seconds on IBM 3081 for 100 iterations.

TABLE III

Comparison of the Cost Update Algorithms in Terms of (a) Number of Problems the Algorithm Found the Best Solution and (b) Average % Above the Best Solution

	Direct Cost		Average Cost	
Problem Set	# of problems	% above best	# of problems	% above best
RP1	4	0.30	1	4.38
RP2	2	2.87	3	5.57
RP3	4	0.09	1	1.68
RP4	2	1.23	3	2.04
RP5	2	1.65	3	0.97
All problems	14	1.58	11	2.97

TABLE IV

Comparison of the Cost Update Algorithms in Terms of Average % above the Best Solution after n Iterations

Algorithm	$n \leqslant 10$	$n \leqslant 20$	$n \leqslant 50$	$n \leqslant 75$	$n \leqslant 100$
Direct	7.4	5.0	3.0	2.4	1.58
Average	5.4	4.3	3.3	3.0	2.97

comparison of the initial to the final solution would give a measurement of the effectiveness of the iterative scheme we employ. We can see in Table II that the *average improvements* range from 10% to over 25% while on a problem by problem basis the improvements range from a low of about 5% to a high of over 40% for the total cost. The average improvements on the number of violated windows (not shown here) range from a low of about 10% to over 26% depending on the cost update algorithm and the tightness of the time constraints. However, these numbers can be treated as indicative performance measurements only and cannot reveal any information concerning the optimal values for the problems tested. They justify, however, the use of an iterative algorithm for the VRP-S model.

Considering the violation of the time windows, one should expect that servicing all the customers within their time windows becomes more difficult as the time constraints become tighter. If we compare the percentages of violated windows from Table II, independently of the cost update algorithm used, we see that the looser the time constraints become, the more customers receive on-time service. The percentage of violated windows rises up to 45% for problem set RP1, which features a very short planning horizon (1 day) and very tight time windows (maximum length 2 hours). We can clearly observe the percentage of violated windows dropping gradually, as the planning horizon and the width of the time windows increase, and eventually reaching 100% on-time service for the loose time constraints problems in RP5.

Comparing the two cost update algorithms, the direct cost update algorithm slightly edged the average cost method, but the two were very close. It is clear from Table III that the direct cost update and the average cost update algorithm came very close to each other providing the best solution in terms of total cost in 11 and 10 problems respectively, out of the 25 problems tested. In terms of the quality of the obtained solutions the solutions of the direct cost update algorithms are 1.58% over the best solution found. The detailed results available in Koskosidis[21] show only 4 problems (out of the 25 tested) with solutions above 3% of the best solution. The corresponding numbers for the average cost update algorithm are 2.97% with 7 problems above 3%. Clearly, the direct cost update algorithm worked better than the average cost update algorithm. Although, they managed to provide the best solution to about the same number of problems, the first one managed to provide solutions closer to the best, when not providing the best. It is interesting to observe however the "dynamic" behavior of the algorithms. It is clear from Table IV that on the average the direct cost update algorithm required more iterations to obtain its best solution compared to the average cost update algorithm. Although, the first one provided better "final" solutions, the average cost update algorithm performed better in the short run. Comparing the cost update algorithms in terms of the percentage of customers that receive on-time service, the direct cost update

algorithm performed again better than the other two, as shown in Table II.

4.2. Benchmark Problems

One of the major objectives of this research has been to investigate the performance of global algorithms (such as the optimization based heuristic for the VRP-S) versus local improvement heuristics. A side by side comparison would only be possible by testing the algorithms on a common problem database. Solomon and Desrosiers[36] recognize the lack of standardized test problems for vehicle routing and scheduling with time windows, and emphasize the need for standardization of the computational experiments conducted. Solomon[37] has proposed an extensive data bank, which has been already utilized by several researchers such as Baker and Schaffer[2] and SORENSEN.[40] This data base includes standard VRP problems (see for example Christofides et al.[7]), which received randomly generated time window constraints, and cover a wide range of routing and scheduling environments in terms of the length and positioning of the time windows.

The three sets of problems (named R1, C1, RC1) include 100 customers each and have different spatial and temporal characteristics. For the set R1 the customer coordinates, demand data and time windows have been randomly generated and attached to the customers.[7, 37] From this set we have solved problems R101, R102, R103, R104, R108 and R109. The first four have been chosen because they all have the same characteristics except for the percentage of customers which receive time window constraints, ranging from 100% for R101 to 25% for R104. That would enable us to draw conclusions on the effect of the density of the time constraints on the quality of the solutions. The other two problems have been chosen randomly. In the problems of set C1 the customers are clustered in ten distinctive groups of 10 customers each. The best cluster by cluster solution found (Solomon[37]) requires 10 vehicles with a routing cost of 829 units. We have solved all nine problems in this class, because our algorithm showed an exceptional capability to deal with cluster data and we would like to verify that this has not been coincidental. Finally, the problems in RC1 are combinations of problems in R1 and C1. A fleet of 10 vehicles is required when no time constraints are imposed. We have chosen problems RC101 through RC104 for the same reasons as in class R1 and problems RC106 and RC108 at random. For more detailed description of the problems the reader is referred to Christofides et al.[7] and Solomon.[37]

4.2.1. *Soft vs. Hard Time Windows, Once Again*

We have seen that the VRP-S model treats time window constraints quite differently than the Solomon heuristics.[37, 39] Solomon treats the windows as hard constraints with no time window violation allowed. To the contrary, the VRP-S model has been designed on the basis that 100% on-time service is not strictly guaranteed, but is obtained through the time penalties in the objective function. This turned out to be a problem for the testing, because we could only do the comparison between the two algorithms if the VRP-S model produced solutions with 100% on-time service. Since 100% on-time service cannot be enforced through our iterative approximation algorithm, we had to guide the algorithm to exclude customers with incompatible time windows from the same cluster. This is done by identifying pairs of customers which cannot be serviced on time, if included in the same cluster.

Two customers i and j have incompatible time windows if

$$a_i + s_i + \tau_{ij} > b_j$$
$$a_j + s_j + \tau_{ji} > b_i$$

Then if i and j are in the same cluster, at least one of them is serviced outside of the time window.

The proof is trivial.

In order to exclude customers with incompatible windows from the same cluster, we preprocessed the data sets to identify all the pairs of incompatible customers. Then, we modified the clustering algorithms such that incompatible customers are not allowed in the same cluster. This turned out to be very useful in avoiding time window violations. However, the identification of incompatible pairs was not enough to guarantee solutions with 100% on-time service. The fact that all the customers in a cluster are pairwise compatible does not always prevent window violations. There might exist three customers, pairwise compatible, which cannot be all serviced on-time if they are in the same cluster. As a result, the VRP-S model was not always able to solve a problem with 100% on-time service.

4.3. Numerical Results

Several adjustments were made to the VRP-S model to comply with the given data sets. First we preprocessed the data to identify and mark pairs of customers with incompatible time widows. Then we used non-linear penalty coefficients in an additional effort to guide the algorithm to 100% on-time service solutions. The algorithm starts with low

penalty coefficients, which are gradually increased. This way, we keep a balance between the spatial and the temporal costs during the early iterations of the algorithm. As the algorithm proceeds and the configuration of the clusters and the tours is stabilizing, the penalty coefficients are gradually increased to create solutions with emphasis on the time window satisfaction. Finally, the vehicle capacities have been reduced appropriately to ensure a vehicle utilization of about 85%, which has been found convenient for the iterative approximate heuristic. This has been deemed necessary since the data sets produce low vehicle utilization which favors local improvement algorithms.

The vehicle capacity constraints can play an important role in the quality of the obtained solution in terms of the percentage of violated windows. Loose capacity constraints could allow too many customers to be assigned to a vehicle and thus turning on-time service into a difficult (if at all possible) task. As a result, reducing the vehicle capacities appropriately would help to spread the customers more evenly among the clusters. As it turned out, the capacity constraints on the benchmark problems we have solved were loose and thus artificial reduction of the capacities was necessary.

The results on the benchmark problem testing are shown on Tables V through VII. The solutions obtained using the VRP-S model are compared to those obtained by Solomon[39] and Baker and Schaffer.[2]

The tables report the total routing cost (distance), number of vehicles used and CPU time in seconds for each problem solved. Column one lists the best solution in terms of total routing cost over all heuristics tested by Solomon. The numerical tests were performed on a DEC-10 computer. Column two lists the best solution found by Baker and Schaffer using 2-opt and 3-opt improvement heuristics applied to initial solutions generated by the insertion and nearest neighbor heuristic of Solomon. The computational time reported is CPU seconds on UNIVAC 1100/80. Finally, the third column lists the best solution obtained through the VRP-S model, using the direct cost update algorithm. The CPU time reported is seconds on a IBM3081/VM370 computer. The additional entry (fourth line) for the VRP-S model indicates the percentage of customers that receive on-time service, which is occasionally less than 100%.

The results in Tables V, VI and VII show a significant improvement of the VRP-S algorithm over Solomon's procedure. Out of the 21 problems, the VRP-S algorithm produced lower routing costs in 14 cases, higher costs in 2 cases, and 5 ties. However, in 4 out of the 14 cases, VRP-S produced time window violations for two to eight percent of the customer. Total routing costs summed over the 21 problems were 26,174 for Solomon and 24,504 for VRP-S, a 6% reduction.

TABLE V

Solution Results for Benchmark Problem set R1

Problem	Solomon	B&S	VRP-S
R101	1831	1672	1856
	21	20	21
	2.7	156.6	209.1
			100%
R102	1705	1531	1628
	19	19	19
	2.8	257.7	325.6
			100%
R102	2064		
	18		
	3.0		
R103	1484	1414	1428
	14	13	14
	3.0	76.4	369.3
			100%
R104	1188	1119	1047
	11	11	11
	3.4	304.5	856.4
			100%
R104			1114
			10
			903.3
			100%
R108	1137	1036	975
	10	10	10
	3.5	514.4	872.5
			100%
R109	1355	1258	1244
	13	13	13
	2.9	343.6	685.6
			98%

First row: routing cost; second row: Number of vehicles used; third row: CPU seconds; and fourth row: Percentage of nonviolated windows.

The comparison to Baker and Schaffer, on the other hand, is not conclusive. B&S report results for only nine problems, with VRP-S outperforming B&S on four problems, losing on three with two ties.

One conclusion is clear from these numbers: VRP-S is significantly slower than Solomon's procedure. While the development of the VRP-S software did not emphasize execution times and a number of steps could be taken to accelerate the algorithm, it is unlikely that the run times would come close to those of the Solomon heuristics.

TABLE VI

Solution Results for Benchmark Problems Set C1

Problem	Solomon	B&S	VRP-S
C101	829	829	829
	10	10	10
	1.1	130.1	3.0
			100%
C102	966		829
	10		10
	3.0		3.2
			100%
C103	1026		829
	10		10
	1.6		2.9
			100%
C104	952		829
	10		10
	1.6		3.0
			100%
C105	829	829	829
	10	10	10
	1.1	118.6	3.0
			100%
C106	834		829
	10		10
	1.5		3.3
			100%
C107	829		829
	10		10
	1.1		3.2
			100%
C108	829	855	829
	10	10	10
	1.1	118.6	2.9
			100%
C109	829		829
	10		10
	1.1		2.9
			100%

First row: routing cost; second row: Number of vehicles used; third row: CPU seconds; and fourth row: percentage of nonviolated windows.

TABLE VII

Solution Results for Benchmark Problem Set RC1

Problem	Solomon	B&S	VRP-S
RC101	1866		1815
	16		16
	2.7		628.3
			95%
RC102	1881		1605
	14		14
	2.9		570.0
			94%
RC102	1697		
	15		
	2.8		
RC103	1640		1390
	13		13
	2.9		673.8
			100%
RC104	1300		1199
	11		11
	3.3		834.7
			100%
RC104			1353
			10
			834.7
			100%
RC106	1611		1541
	13		13
	2.9		693.1
			92%
RC108	1253		1315
	11		11
	3.2		736.6
			99%

First row: routing cost; second row: Number of vehicles used; third row: CPU seconds; and fourth row: percentage of nonviolated windows

5. DISCUSSION

WE WOULD like to highlight several points at the conclusion of this article. To start with, the computational experiments with the VRP-S model, revealed some limitations of the optimization based heuristic. The computational time requirements of the VRP-S model have been one to two orders of magnitude in excess of the CPU time of the local heuristic algorithms. The CPU requirements of the improvement heuristics are almost as high as those of the VRP-S model. These results come at no surprise however, since both the global heuristic and the solution improvement heuristics do much more work than the local heuristics.

On the other hand, the performance of the VRP-S model on the highly structured data sets C1 has been really impressive. The algorithm managed to find solutions of all the problems tested with minimal computational requirements. Based on the structure of these problems, the solutions obtained are believed to be optimal. The quality of the solutions can be attributed to the clustering problem solved during the assignment/clustering phase of the algorithm. Clearly, the clustering algorithm can very quickly identify the clustered structure of the problems.

Concerning the occasional difficulty of the VRP-S model to provide solutions with 100% on-time service, note that the hardest problems to solve are those that combine loose capacity constraints with loose time window constraints. Recall the clustering algorithm from Section 2. During the

assignment phase, customers are assigned to seeds in a greedy manner, according to the REGRET function and the capacity constraints. The procedure results in the assignment of as many customers as the capacity constraints allow to a few very attractive seeds, yielding unbalanced clustering solutions, i.e., solutions where some clusters include too many customers and some too few. Given an unbalanced clustering solution, it is very hard to satisfy all the time windows. In situations like these, where some seeds tend to be very desirable and attract too many customers, tight capacity constraints are quite useful to balance the solution. The capacity constraints for the problems in the Solomon data base are generally loose. For example, the total demand for the problems in set R1 is 7.3 truckloads, which yields vehicle utilization between 0.35 and 0.65. This is obviously quite low compared to the over 0.85 vehicle utilization obtained when testing the random problems we generated.

Loose capacity constraints are very desirable for local insertion and improvement heuristics, since they allow more flexibility to perform local operations, such as insertion of a customer in an existing tour or pairwise switches between tours. To the contrary, tight capacity constraints would be very desirable to the VRP-S model, since they tend to balance the greed of the clustering algorithms, and avoid bad clustering solutions.

In the benchmark problems we solved the lack of tight capacity and time constraints by means of artificially restricting the capacity of the vehicles and preventing customers with incompatible windows from being in the same cluster. Furthermore, we hoped that solutions with 100% on-time service would be obtained by appropriately increasing the penalty coefficients and/or the time weight factor. However, this was not always possible, since the increase in time cost should be kept within limits. If the temporal costs overpower significantly the spatial costs, the algorithm would produce solutions which could be counter intuitive in terms of the routing. In addition, the solution of the TSP-S problems is based on the approximate cost coefficients $\hat{\mathbf{c}}$. If the $\hat{\mathbf{c}}$ matrix becomes unrealistic, due to an irrational increase of the temporal cost, then the routing solutions would be irrational as well.

Summarizing, the VRP-S model compares favorably to the state-of-the-art local heuristic and improvement heuristic algorithms. The computational results have been quite encouraging in terms of the quality of the solutions produced by the global optimization based heuristic, and they convincingly show that an optimization based algorithm should and would match or exceed in performance local and improvement heuristics. The computational requirements of the algorithm, although they exceed those of the local heuristics, they have been kept within reasonable limits for realistic size applications, and they compare to those of the solution improvement heuristic. Some recent studies of the author suggest that the CPU time requirements of the algorithm can be reduced by more than 50% by using simple pruning rules on real world data from the motor carrier industry.

ACKNOWLEDGMENTS

THIS RESEARCH was supported in part by the National Science foundation under a Presidential Young Investigator Award, ECE-8451466.

REFERENCES

1. A. ASSAD, M. BALL, L. BODIN AND B. GOLDEN, "Combined Distribution Routing and Scheduling in a Large Commercial Firm," in *Proceedings of Northeast AIDS Conference 1981*, R. Paran and P. Anderson (eds.), pp. 99–102, Boston, 1981.
2. E. BAKER AND J. SCHAFFER, "Solution Improvement Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," *Am. J. Math. Mgmt. Sci.* **6**, 261–300 (1986).
3. M. BALL, B. GOLDEN, A. ASSAD AND L. BODIN, "Planning for Truck Fleet Size in the Presence of a Common Carrier Option," *Decision Sci.* **14**, 103–120 (1983).
4. E. BELTRAMI AND L. BODIN, "Networks and Vehicle Routing for Municipal Waste Collection," *Networks* **4**, 65–94 (1974).
5. L. BODIN, B. GOLDEN, A. ASSAD AND M. BALL, "The State-of-the-Art in the Routing and Scheduling of Vehicles and Crews," *Comput. Opns. Res.* **10**, 63–211 (1983).
6. N. CHRISTOFIDES AND S. EILON, "An algorithm for the Vehicle Dispatching Problem." *Opns. Res. Quart.* **20**, 309–318 (1969).
7. N. CHRISTOFIDES, A. MIGNOZZI AND P. TOTH, "The Vehicle Routing Problem," in *Combinatorial Optimization*, P. Toth, N. Christofides, R. Mignozzi and C. Sandi (eds.), John Wiley, New York, 1979.
8. N. CHRISTOFIDES, A. MIGNOZZI, AND P. TOTH, "Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations," *Math. Program.* **20**, 255–282 (1981).
9. T. COOK AND R. RUSSELL, "A Simulation and Statistical Analysis of Stochastic Vehicle Routing with Timing Constraints," *Decision Sci.* **9**, 673–687 (1978).
10. F. CULLEN, J. JARVIS AND H. RATLIFF, "Set Partitioning Based Heuristics for Interactive Routing," *Networks* **11**, 125–144 (1981).
11. M. FISHER AND R. JAIKUMAR, "A Decomposition Algorithm for Large-Scale Vehicle Routing," Working Paper 78-11-05, University of Pennsylvania, 1978.

12. M. FISHER AND R. JAIKUMAR, "A Generalized Assignment Heuristic for the Vehicle Routing Problem," *Networks* **11**, 109–124 (1981).
13. M. FISHER, R. JAIKUMAR AND L. VAN WASSENHOVE, "A Multiplier Adjustment Method for the Generalized Assignment Problem," *Mgmt. Sci.* **23**, 1095–1103 (1986).
14. M. FISHER AND M. ROSENWEIN. "An interactive optimization system for bulk cargo ship scheduling," Working Paper 85-08-07, University of Pennsylvania, *Decision Sciences*, 1985.
15. L. FRANTZESKAKIS, Untitled Report, Princeton University, 1986.
16. K. JORNSTEN, O. MADSEN AND B. SORENSEN, "Exact Solution for the Vehicle Routing and Scheduling Problem with Time Windows by Variable Splitting," Working Paper, IMSOR, Denmark, 1986.
17. R. KARP, "Reducibility among Combinatorial Problems, in *Complexity of Computer Computations*, R. Miller and J. Thacher, editors, (eds.), Plenum Press, New York, 1972.
18. R. KARP, "On the Complexity of Combinatorial Problems," *Networks* **5**, 45–68 (1975).
19. K. KNIGHT AND J. HOFER, "Vehicle Scheduling with Timed and Connected Calls: A Case Study," *Opns. Res. Quart.* **19**, 299–310 (1968).
20. A. KOLEN, A. RINNOOY KAN AND H. TRIENEKENS, "Vehicle Routing with Time Windows," *Opns. Res.* **35**, 266–273 (1987).
21. I. KOSKOSIDIS, "Optimization Based Models and Algorithms for Routing and Scheduling with Time Window Constraints," Ph.D. thesis, Princeton University, 1988.
22. I. KOSKOSIDIS AND W. POWELL, "Clustering Algorithms for Consolidation of Customer Orders into Vehicle Shipments," *Transportation Research* (to appear).
23. J. LENSTRA AND A. RINNOOY KAN, "Complexity of the Vehicle Routing and Scheduling Problems," *Networks* **11**, 221–228 (1981).
24. S. LIN, "Computer Solutions of the Traveling Salesman Problem," *Bell Syst. Tech. J.* 44:2245–2269, 1965.
25. S. LIN AND B. KERNIGHAN, "An Effective Heuristic Algorithm for the Traveling Salesman Problem," *Opns. Res.* **21**, 498–516 (1973).
26. O. B. G. MADSEN, "Variable Splitting and Vehicle Routing Problems with Time Windows," Paper presented at the EURO IX/TIMS XVIII International Conference, Paris, France, 1988.
27. T. MAGNANTI, "Combinatorial Optimization and Vehicle Fleet Planning: Perspectives and Prospects," *Networks* **11**, 179–214 (1981).
28. C. MILLER, A. TUCKER AND R. ZEMLIN, "Integer Programming Formulations and Traveling Salesman Problems," *J. Assoc. Comput. Mach.* **7**, 326–329 (1960).
29. J. MULVEY AND H. CROWDER, "Cluster Analysis: An Application of Lagrangian Relaxation." *Mgmt. Sci.* **25**, 329–340 (1979).
30. J. M. MULVEY AND M. P. BECK, "Solving Capacitated Clustering Problems," *Eur. J. Opn. Res.* **18**, 339–348 (1984).
31. C. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, N.J., 1982.
32. H. PULLEN AND M. WEBB, "A Computer Application to a Transport Scheduling Problem," *Comput. J.* **10**, 10–13, 1967.
33. R. RUSSELL AND W. IGO, "An Assignment Routing Problem," *Networks* **9**, 1–17 (1979).
34. M. SAVELSBERGH, "Local Search in Routing Problems with Time Windows," Technical report, 1985.
35. N. L. SCHWARTZ, "Discrete Programs for Moving Known Cargos from Origins to Destinations on Time at Minimum Bargeline Fleet Cost," *Trans. Sci.* **2**, 134–145 (1968).
36. M. SOLOMON AND J. DESROSIERS, "Time Window Constrained Routing and Scheduling Problems," *Trans. Sci.* **22**, 1–13 (1988).
37. M. M. SOLOMON, "Vehicle Routing and Scheduling with Time Window Constraints: Models and Algorithms," Ph.D. thesis, University of Pennsylvania, 1984.
38. M. M. SOLOMON, "On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," *Networks* **16**, 161–174 (1986).
39. M. M. SOLOMON, "Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints," *Opns. Res.* **32**, 254–265 (1987).
40. B. SORENSEN, "Interactive Distribution Planning," Ph.D. thesis, IMSOR, Lyngby, Denmark, 1986.

(Received, March 1989; revision received June 1990; accepted September 1990)