A Decomposition Approach to the Inventory Routing Problem with Satellite Facilities*

JONATHAN F. BARD[1]

Graduate Program in Operations Research, Department of Mechanical Engineering, The University of Texas at Austin, Austin, Texas 78712-1063

LIU HUANG

DSC Communications, 11009 Metric Boulevard, Building 822, Austin, Texas 78758-4017

PATRICK JAILLET[2]

Department of Management Science and Information Systems, The University of Texas at Austin, Austin, Texas 78712

MOSHE DROR

College of Business and Public Administration, University of Arizona, Tucson, Arizona 85721

This paper presents a comprehensive decomposition scheme for solving the inventory routing problem in which a central supplier must restock a subset of customers on an intermittent basis. In this setting, the customer demand is not known with certainty and routing decisions taken over the short run might conflict with the long-run goal of minimizing annual operating costs. A unique aspect of the short-run subproblem is the presence of satellite facilities where vehicles can be reloaded and customer deliveries continued until the closing time is reached. Three heuristics have been developed to solve the vehicle routing problem with satellite facilities (randomized Clarke–Wright, GRASP, modified sweep). After the daily tours are derived, a parametric analysis is conducted to investigate the tradeoff between distance and annual costs. This leads to the development of the efficient frontier from which the decision maker is free to choose the most attractive alternative. The proposed procedures are tested on data sets generated from field experience with a national liquid propane distributor.

Inventory routing problems (IRP) arise when a large number of customers rely on a central supplier to provide them with a given commodity on a regular basis (BELTRAMI and BODIN, 1974), (DROR, BALL, and GOLDEN, 1985). These problems usually involve the specification of a sequence of locations that must be visited by a delivery vehicle to restock a storage tank or bin. The objective is to minimize the annual delivery cost while attempting to ensure that no customer stocks out at any time. To avoid the trivial case, it is assumed that the annual demand of each customer is greater than its tank size; that is, each customer requires more than one replenishment per year. Whenever a customer stocks out, a special delivery must be made immediately, which incurs a high cost.

A key feature of this problem is that a customer's consumption rate is a random variable that is often customer-dependent (DROR and TRUDEAU, 1986). In addition, inventories must be continually main-

*Accepted by Gilbert Laporte.
[1]This work was partially supported by the Texas Higher Education Coordinating Board under the Advanced Research Program, grant ARP-003.
[2]Department of Mathematics, ENPC, Paris, France.

Transportation Science
Vol. 32, No. 2, May 1998

0041-1655/98/3202-0189 $01.25

tained at adequate levels to minimize the chances of stock outs. The problem, therefore, combines elements of routing with explicit inventory considerations. The three interrelated tasks include: (1) customer selection: identification of the customers to be visited (serviced on a particular day or during a particular week); (2) customer vehicle assignment: assignment of customers selected for service on a particular day to one of the available trucks; (3) routing: the construction of efficient routes for each truck over the set of its assigned customers. The latter two tasks generally define the classical vehicle routing problem (VRP). The first adds a degree of complexity that makes the IRP considerably more challenging than the VRP.

Whereas the objective of the IRP is to minimize annual delivery costs, actual routing plans can only be issued in the short term. To deal with these two different time scales, we introduce a long-term cost measure to serve as a criterion for customer selection. Short-term routing decisions are based on distance minimization. An acceptable balance is achieved by trading off one measure against the other.

In many IRP applications, particularly those involving fuel deliveries, suppliers have found it beneficial to set up several satellite facilities (usually two or three in large districts). These depots are geographically scattered throughout the service area and permit drivers to refill their vehicles with the commodity during a shift. This leads to an effective increase in vehicle capacity and perhaps an increase in route length. The efficiency of the overall routing may also be increased if the tours are well scheduled. Typically, each vehicle must return to the central depot at the end of the day rather than remain overnight at a satellite facility.

There have been a number of studies on the IRP, but none has specifically addressed the IRP with satellite facilities (IRPSF). The purpose of this paper is to integrate our work on annual costing (JAILLET et al., 1997) with algorithms for finding good solutions to the IRPSF. In so doing, we develop an analytic framework that accommodates the need for real-time decision making. The proposed approach is based on the idea of a rolling horizon. The first step is to identify customers that are to be visited during the planning horizon, say, two weeks. The identification process centers on the fact that, for each customer, there exists an optimal (expected) frequency, and hence, day on which service should be provided. If the next scheduled visit falls within the planning horizon the customer is selected. After an adjustment is made to balance daily demand, customers scheduled for the first week are routed. The planning horizon is then extended and the process is repeated. Performance is measured by two separate functions: distance traveled and total incremental costs (costs incurred by not servicing a customer on his or her optimal day). Rather than trying to combine these two measures into a single value, we take a bi-criteria approach and derive the efficient frontier.

The paper is organized as follows. In Section 1, we review the pertinent work on inventory routing and related distribution systems. Assumptions are presented in Section 2. The proposed solution methodology is detailed in Section 3 along with three VRP-based heuristics modified to account for satellite facilities. The first is a randomized version of the Clarke–Wright (CW) algorithm (CLARKE and WRIGHT, 1964), the second is a GRASP (KONTORAVDIS and BARD, 1995), and the third is a revised sweep algorithm (GILLETT and MILLER, 1974). Section 4 contains our computational experience. An extensive analysis was conducted to gain an understanding of the tradeoffs between cost and distance, as well as the effectiveness of the routing heuristics. For the larger data sets, the CW algorithm was seen to outperform the others.

1. RELATED WORK

IRP APPLICATIONS FIRST APPEARED in the 1970s (Beltrami and Bodin, 1974) with a focus on modeling and simple solution techniques. The seminal work of GOLDEN, ASSAD, and DAHL, (1984) defined the basic components of the problem in the context of a large energy-products company involved in the distribution of liquid propane to residential and industrial customers. The goal of the distribution system was to maintain an adequate level of inventory for all customers. In particular, each customer had a known tank capacity, and its fuel level was expected to remain above a prespecified critical value. The system forecasted customer demand, selected a subset of customers for service, and generated vehicle routes on a daily basis. The main tool used by the authors was a simulation model that contained routing algorithms for planning of delivery operations. The inventory levels of customers were simulated daily by running the model. The experimental design showed that the performance of the routing algorithm was superior to the firm's current system.

Dror et al. (1985) addressed the same problem and compared several different computational schemes. In their work, they start with a set of customers, where each customer has a storage tank of known capacity and a probability distribution defining daily consumption. Daily consumption rates are as-

sumed to be independent, identically distributed (i.i.d.) normal random variables with known parameters. The problem is formulated as a two-stage integer program that handles customer selection and scheduling of replenishments by route, truck, and day of the week. The primary solution approach proposed is based on hierarchical decomposition. Some computation results are presented.

A number of the specific issues raised by Dror et al. (1985) are investigated in more detail in companion papers. DROR and BALL (1987) address the issue of reducing the annual distribution problem to a single period problem. Dror and Trudeau (1986) and DROR, LAPORTE, and TRUDEAU (1989) discuss the VRP with stochastic demand and DROR and LEVY (1986) explore several new methods for improving a given set of vehicle routes with node exchange and reoptimization. In two of the procedures, a minimum-weight matching problem is solved repeatedly.

The strategic inventory/routing problem (SIRP) is discussed by LARSON (1988) and WEBB and LARSON (1995). The major differences between the IRP and the SIRP are twofold. First, SIRPs have long lead times, i.e., months or even years between delivery operations. Second, SIRPs focus on estimating the minimum vehicle fleet size that is required to supply inventory. Related to the SIRP is what might be called the route-sales distribution problem. As exemplified in the work of ANILY and FEDERGRUEN (1990, 1993), the corresponding system consists of a single depot and numerous geographically dispersed retailers with deterministic demands. The objective is to determine long-term integrated replenishments strategies enabling all retailers to meet their demands while minimizing long-run average system-wide transportation and inventory costs.

The integrated inventory allocation and vehicle routing problem is another application that is similar to the IRP. FISHER et al. (1983) investigated a bulk delivery version of this problem that, in many ways, parallels the VRP; however, because of the bulk nature of the product, there is a major difference in the way customer demand is specified. For each customer, lower and upper limits exist on the amount of product that must be delivered during each period of the planning horizon. Given a value per unit of product, the objective is to maximize the value of product delivered to all customers less fleet operation costs incurred in making the deliveries. The solution specifies a routing plan, the number of deliveries per customer, and how much product to provide on each visit.

FEDERGRUEN and ZIPKIN (1984) examined the combined problem of allocating a scarce resource available at a central depot among several locations (customers). They considered stochastic demand and nonlinear inventory costs comprised of holding and shortage costs. The problem is to decide which deliveries are to be made by each set of vehicles and in which order.

CHIEN, BALAHRISHMAN, and WONG, (1989) similarly dealt with the problem of allocating and delivering a limited amount of a commodity to customers using a fleet of vehicles based at a single depot. First, customers were selected and then the fleet was routed with the objective of maximizing profit. The problem was formulated as a mixed-integer program and Lagrangian relaxation was used to solve several small instances.

Period routing addresses the problem of designing a set of routes for each day of a given p-day period (CHRISTOFIDES and BEASLEY, 1984), a feature common to the IRP. In the former, each customer may require a number of visits by a vehicle during the period. In contrast, only one visit per period is permitted in our approach to the IRP. This, and the fact that it is assumed that customer visits follow a p-day cycle, are the main differences between the two problems, at least from a routing point of view. Cost issues further complicate the IRP.

2. MODEL DEVELOPMENT

IN REDUCING THE PROBLEM from an annual time base to a weekly time base, Dror et al. (1985) defined two subsets of customers. The first set, denoted by M, are those customers who must be replenished during the given planning period. The second set, denoted by M', are only replenished if there exists a cost-savings opportunity to do so.

In our approach to the IRPSF, we build on the decomposition ideas proposed by Dror and Ball (1987) and Dror et al. (1985), but, rather than divide the customers into two sets and solve a series of single period problems, we examine two periods (weeks) at a time and identify all customers whose *optimal* delivery day falls within this timeframe. Only the solution for the first week is implemented and the process is repeated for the next two weeks. Customer assignments to days of the week are based on a combination of a desire to balance demand and to minimize incremental costs; i.e., the cost of serving a customer on a day other than his optimal day. Our method of computing incremental costs is different from, and more accurate than, what has been proposed in the literature so far (see Jaillet et al., 1997).

Objective and assumptions The IRP is defined by a set of n customers with a non-negative demand for a given commodity that varies daily in accor-

dance with a known probability distribution. Each customer scheduled for service on a given day must be visited by exactly one of m homogeneous vehicles located at a central depot. The depot and the s satellite facilities hold an unlimited supply of the given commodity. The vehicles can be reloaded at each of these locations. The only distinction between the depot and a satellite facility is that the depot serves as the origin and final destination of each vehicle.

A feasible vehicle-delivery sequence starts at the depot, visits a subset of customers dropping off the required amount of commodity to each, reloads perhaps at one of the satellite facilities, continues to another subset of customers, reloads again at a satellite facility, and at the end of the route returns to the depot. An additional requirement for feasibility is that the deliveries on a given day be completed within T hours.

In deciding on the appropriate objective function for the IRP, we are faced with two distinct criteria that measure different aspects of the problem. The first is associated with assigning customers to days of the week. One way to do this is to make the assignments based on the incremental costs alone. Such an approach, however, ignores vehicle capacities and travel-time restrictions. The second criterion is associated with routing customers on a given day. Typically, the objective is to minimize travel time or distance, which are surrogates for minimizing short-term costs. Although it may be possible to translate distance into cost, the results may not be directly additive with the incremental costs because the former are strictly associated with operations and maintenance while the latter may have a nontangible aspect corresponding to quality of service and customer retention.

As such, on a given day, we wish to construct a set of non-overlapping (disjoint in terms of the customers) feasible tours that delivers all the required commodity without violating the vehicle capacity and route closing-time constraints. We wish to do this by simultaneously minimizing total distance and incremental assignment costs. This leads to a bi-criteria mixed-integer linear programming formulation. The details are presented in (HUANG, 1997).

3. SOLUTION APPROACH TO IRP WITH SATELLITE FACILITIES

THE LIMITATIONS OF AVAILABLE computational techniques make it impractical to try to solve the mixed-integer linear programming formulation of the IRPSF directly for all but the smallest of instances. The structure of the problem, though, argues for some type of decomposition. As mentioned, there are two subproblems embedded in the IRP. The first involves clustering of customers and the second involves routing.

Our decomposition scheme for the IRPSF is outlined in the following steps. The details are discussed in subsequent sections.

Step 0. Identify those customers in the service area whose optimal replenishment day falls within the current planning horizon.
Step 1. Assign each customer identified in Step 0 to a given day by solving a balanced assignment problem with the objective of minimizing incremental costs.
Step 2. For each day in the planning horizon, try to find a good feasible solution by solving a VRP with satellite facilities (VRPSF). The objective at this step is to minimize total distance.
Step 3. Attempt to improve the solution by swapping customers between routes.
Step 4. Exam the tradeoff between incremental costs and route lengths by swapping customers between different days of the planning horizon.

In the initialization step, customers are selected for the current period. At Step 1, a generalized assignment problem with constraints that set a range for expected demand on any day is solved. The objective is to minimize the incremental cost of serving all customers. The details are given in Section 3.1.

At Step 2, we use several heuristics to solve a VRPSF for each day in the planning horizon with the objective of minimizing total distance. The mathematical programming formulation for this problem is similar to the one for the IRPSF and is contained in Huang (1997).

3.1 Customer Selection and Assignment

In a complementary paper (Jaillet et al., 1997), we derived the expected total cost of restocking a customer over a given time period, say a year, as a function of the frequency d between refills. The underlying assumptions were that the demand U_k on day k is a random variable with finite expected value μ independent of k, and that, over the given time period, the U_k values are i.i.d. Let the minimum point on the expected total-cost curve be obtained at $d = d^*$. The optimal strategy, then, is to replenish the customer every d^* days. Suppose that, on the first visit, it is necessary to replenish the customer on day d ($\neq d^*$), and, after that, the optimal strategy is resumed. The difference between the expected total cost per year incurred by following this strategy and the minimum expected total cost per year is the incremental cost for customer i, denoted by c_{id}.

The procedure used to select customers for possi-

ble service over the planning horizon of, say two weeks, is based on the frequency d^*. If the best day to visit a customer is within the two weeks, then the customer is selected. Note that a weekend assignment must be shifted to a week day. We define the set S_1 as the group of customers that needs to be restocked in the current week and the set S_2 as the group of customers that needs to be serviced in the following week. Assuming five working days per week and a fixed number of vehicles, it is desirable to balance the demand from one day to the next. This is achieved by solving a generalized assignment problem.

In the model, c_{id} is the incremental cost for customer i on day d, r_{id} is the expected demand of customer i on day d, L is a lower bound on expected total customer demand per day (say, 10% below the expected daily demand), and U is the upper bound on expected total customer demand per day (say, 10% above the expected daily demand). The parameters L and U are designed to take into account the fact that demand is really a random variable, and so, may differ from the forecast. Now let z_{id} be a binary variable equal to 1 if customer i is selected for day d, 0 otherwise. The generalized assignment problem is stated as follows

$$\min \sum_i \sum_d c_{id} z_{id} \tag{1}$$

subject to

$$\sum_d z_{id} = 1 \quad i = 1, \ldots, n \tag{2}$$

$$L \leq \sum_i r_{id} z_{id} \leq U \quad d \in \mathrm{D} \tag{3}$$

$$z_{id} \in \{0,1\} \quad \forall i, d. \tag{4}$$

The model assigns customers to 1 of $|D|$ days according to their incremental cost. Equation (2) assures that each customer is assigned to exactly one day. Constraint (3) assures that expected demand of all customers on a particular day is within the lower bound L and upper bound U. Although problem (1)–(4) is an integer linear program (ILP), it has the property that at least $n - 2|D|$ of the variables in the solution of the associated LP relaxation assume integer values (see LASDON, 1970). Recall that the LP relaxation is obtained by replacing (4) with $0 \leq z_{id} \leq 1$, $\forall i, d$. In our case, $|D| = 10$ corresponding to the 10 working days during the two-week period and $n = 400$. GAMS with OSL solves this ILP in about one minute.

3.2 Heuristics Development for VRPSF

3.2.1 Current VRP Algorithms

The objective of a VRP is to design optimal delivery or collection routes originating from one or more depots to a number of geographically scattered customers, subject to side constraints (LAPORTE, 1992). The most common constraints include (1) capacity restrictions, where a non-negative weight is assigned to each customer, and the resultant sum associated with each vehicle route may not exceed its capacity, (2) the number of customers on any route is bounded, (3) total time restrictions, where the length of any route may not exceed a prescribed value, (4) time windows, where customers must be visited within a time interval, and (5) precedence relations between customers, where customer i may have to be visited before customer j.

According to LAPORTE and NOBERT (1987), exact algorithms for the VRP can be placed into three broad categories: (1) direct tree search methods, which have been used by CHRISTOFIDES, MINGOZZI, and TOTH, (1981) to solve problems ranging in size from 10 to 25 nodes and by LAPORTE, MENURE, and NOBERT, (1986) to solve to optimality asymmetric capacity-constrained VRPs with up to 260 nodes; (2) dynamic programming, where optimal solutions have been obtained for problems with up to 25 nodes; and (3) integer linear programming, where DESROCHERS, DESROSIERS, and SOLOMON (1992) solved VRPs with time windows containing up to 100 nodes using column generation.

Heuristics for the VRP can often be derived from procedures used for the TSP. Several heuristics, though, have been specifically developed for the VRP. The first of three that we mention is the CW algorithm, which starts with n routes, each containing the depot and one other node. At each step, two routes are merged according to the largest saving that can be generated. The second is the sweep algorithm (Gillett and Miller, 1974), (WREN, 1971), also called sectorial region partitioning. Using polar coordinates (θ_i, ρ_i), where θ_i is the angle and ρ_i is the distance to customer i, the algorithm begins by assigning a value $\theta_{i^*} = 0$ to an arbitrary node i^* and then computes the angles of the remaining $(n - 1)$ nodes. The nodes are then ranked in increasing order of their θ_i values. Next, an unused vehicle is chosen and starting from an unrouted node having the smallest angle, customers are assigned to the vehicle as long as its capacity is not exceeded. The procedure is repeated until all customers are routed.

Tabu search has also been used to solve the VRP. From a computational point of view, the implementation by GENDREAU, HERTZ, and LAPORTE, (1994)

provides the best solutions among existing heuristics; however, its running time can be excessive. From a practical point of view, the CW algorithm strikes a good balance between solution quality and running time.

3.2.2 *Heuristics for VRPSF*

At Step 2 of our decomposition scheme, a VRPSF must be solved for each day in the planning horizon. In this subsection, we describe three heuristics developed for this purpose. The first is a modification of the CW algorithm, the second is an extension of the GRASP developed by Kontoravdis and Bard (1995) for the VRP with time windows, and the third is a revised version of the sweep algorithm. Depending on the heuristic, a subset of the feasible solutions found are post-processed in an attempt to improve the results. The algorithm for doing this is the same in each case and is presented in the next section. Throughout the discussion, the words "route" and "tour" will continue to be used interchangeably.

The ability to refill a vehicle during the day implies that the length rather than the cumulative demand is the real limitation of a route. The procedures used to insert satellite facilities into partial routes during tour construction is specific to the individual heuristic. In each case, we investigated a number of different schemes but report only the alternative that yielded the best results for the heuristic under study. Similarly, the parameter settings used in the final calculations were arrived at after extensive experimentation.

Note that the heuristics are presented under the assumption that feasible solutions to the VRPSF requiring m or fewer vehicles can always be found. If this is not the case for a particular instance of the IRPSF, the procedure discussed in Section 3.4 is invoked using the best feasible solution obtained regardless of the number of vehicles required. If feasibility is still not achieved, customers are removed from the planning horizon one at a time until a solution is found that uses, at most, m vehicles per day. The removal procedure is a function of individual demand, the day in question, and the number of customers assigned to each vehicle on that day.

Revised CW Algorithm. In adapting the CW algorithm, we introduced a degree of randomness into the construction phase. We start with one route for each customer and calculate the savings associated with merging each pair of customers into a single route. Instead of merging two routes with the largest savings as is usually done, we rank the customer pairs according to their savings and then construct a list L of size r. The list is in nonincreasing order of savings and only has to be constructed once because only customers linked directly to the depot can be merged at any stage of the CW methodology. We then randomly select one pair of customers from L and record the savings. When this merger results in a violation of the vehicle-capacity constraint, a satellite facility is inserted between the two customers as long as the maximum tour time is not exceeded. Which satellite facility to select is determined by calculating insertion costs for each candidate. The one with the smallest insertion cost that does not violate the maximum tour time is then inserted in the appropriate position.

Note that, at this point, it is not necessary to satisfy the capacity constraint because other customers may still be added to the tour. In the post-processing phase, we adjust the tour to assure that no vehicle's capacity is exceeded.

Step 1. Construct n distinct tours $(0, i, 0)$, each containing the depot and one customer only, where $i = 1, \ldots, n$, and 0 is the depot.

Step 2. Let τ_{ij} be the distance from customer i to customer j. Compute the savings of merging each pair of customers i and j: $s_{ij} = \tau_{i0} + \tau_{j0} - \tau_{ij}$ for i, $j = 1, \ldots, n$ and $i \neq j$.

Step 3. Place the savings $s_{ij} \geq 0$ on a nonincreasing list $\hat{L}$.

Step 4. Construct a list L of size r from the first r entries of $\hat{L}$. Put $\hat{L} \leftarrow \hat{L} \backslash L$

Step 5. Randomly select one element s_{ij} from list L. Suppose it is from the pth position of L. Merge routes with arcs $(i, 0)$ and $(0, j)$, respectively, when the resulting route is feasible and then go to Step 7. Otherwise go to Step 6 and try to insert a satellite facility in the route.

Step 6. Calculate the cost of inserting each satellite facility between arcs $(i, 0)$ and $(0, j)$. For the candidate with the smallest insertion cost that does not violate the maximum tour time, merge the routes with arcs $(i, 0)$ and $(0, j)$ and insert the satellite facility between arcs $(i, 0)$ and $(0, j)$.

Step 7. If $\hat{L} \neq \varnothing$, take the first element in $\hat{L}$ and put it in position p in L. Go to Step 5. Otherwise, eliminate the pth position from L. If $L \neq \varnothing$, go to Step 5; otherwise go to Step 8.

Step 8. Starting with the n distinct tours constructed in Step 1, repeat Steps 3 to 7 a predetermined number times (say, M) and save the best route after each iteration.

Step 9. Call post-processor after the M iterations to try to improve the solution.

Figure 1 illustrates the logic of inserting a satellite facility between arcs $(i, 0)$ and $(0, j)$, and hence, merging the corresponding routes. The cost of this

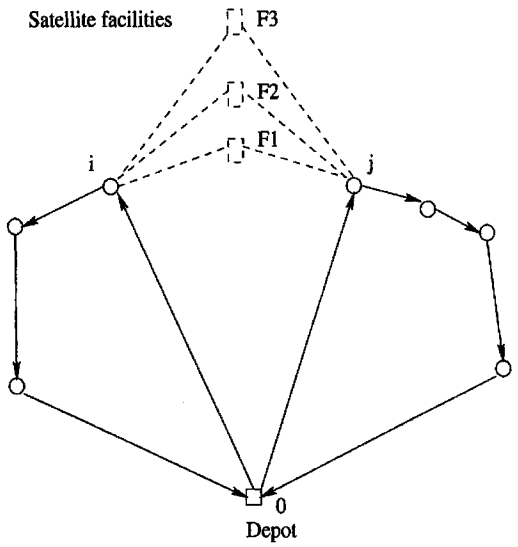

Fig. 1. Insertion of satellite facilities.

merger when satellite facility F_1 is inserted is $\tau_{iF_1} + \tau_{jF_1} - \tau_{i0} - \tau_{j0}$. Because routes are directional, the initial order in which the customers in the partial route containing i was to be visited is now reversed. In our implementation, $r = 3$ at Step 4 and $M = 50$ at Step 8.

GRASP. Kontoravdis and Bard (1995) addressed the problem of finding the minimum number of vehicles needed to service n customers subject to time window and capacity constraints. We have modified their greedy randomized adaptive search procedure to solve the VRPSF.

Step 1. Find a lower bound v on the number of routes needed.
Step 2. Select v seed customers to form a set of initial routes.
Step 3. Calculate the cost of inserting each customer into these routes.
Step 4. Calculate penalties for each insertion.
Step 5. Construct a list L that contains the r largest penalties. From L, randomly choose one customer.
Step 6. Try to insert the customer in the corresponding route. If successful, go to Step 8. If insertion leads to a time violation, start a new route between the depot and the customer and go to Step 4. If insertion leads to a capacity violation, go to Step 7.
Step 7. Find the closest satellite facility to the customer, and then find the closest route to that satellite facility. Now, identify the best position in this route to insert the satellite facility. If the maximum tour time is not violated, insert the satellite facility in this position, revise the insertion costs for all unrouted customers with respect to this route, and go to Step 4; otherwise, add a new route between the depot and the customer, compute the insertion costs for all unrouted customers with respect to this route, and go to Step 4.
Step 8. If every customer is routed, then stop; otherwise, update the costs and go to Step 5.
Step 9. Repeat Steps 1 to 8 a predetermined number (say, M) times and save the best results. During these M iterations, run the post-processor every N ($\leq M$) iterations on the best of the N solutions.

In Step 1, the minimum number of routes needed is determined. When the fleet is homogeneous, a simple way to compute this value is to divide the customers' total demand by the truck capacity and round the result up to the nearest integer. A better way is to solve a bin packing problem in which each truck is viewed as a bin and customer demand is equivalent to the size of items to be packed. Although the bin packing problem is *NP*-hard it can be solved quickly for instances with several hundred demand points.

Seed customers are selected at Step 2 by trying to identify customers who are least likely to appear in the same route in an optimal solution. One way to do this is to find customers who are geographically far apart. A second way is to take into consideration customer demands. Because vehicles have capacity limits, it is not possible to serve two customers whose total demand is more than the vehicle capacity without a visit to a satellite facility.

After obtaining v initial routes, some customers can be inserted into the routes. In Step 3, the cost of inserting each customer into each position of each current route is calculated. Suppose γ^{ρ}_{ijk} is the cost of inserting customer i into route ρ between customers j and k. Then $\gamma^{\rho}_{ijk} = \tau_{ij} + \tau_{ik} - \tau_{jk}$. Let $\gamma_{\rho i} = \min_{j,k} \gamma^{\rho}_{ijk}$, where $\gamma_{\rho i}$ is the minimal insertion cost for node i on route ρ. Let $\gamma_{\rho^* i} = \min_{\rho} \gamma_{\rho i}$, the overall minimal insertion cost for customer i. In Step 4, a penalty associated with not assigning customer i at this iteration is calculated. Let $\Pi_i = \Sigma_{\rho} (\gamma_{\rho i} - \gamma_{\rho^* i})$ be the penalty for customer i. The customers with larger penalties are assigned first.

In Step 5, penalties are sorted in nonincreasing order, and one is chosen from the first r on the list (again, $r = 3$ in the implementation). In Step 6, the selected customer is inserted into the route if doing so does not violate the capacity and tour-time restrictions. A violation of the latter requires that a new route be started. In this case, the algorithm returns to Step 4 to update the penalties of all unrouted customers. Alternatively, a violation of the

capacity restriction places us at Step 7, where an attempt is made to find a route and accompanying position in which to insert a satellite facility. If the candidate identified does not violate the time restriction, it is inserted and the insertion costs for all unrouted customers are revised with respect to this augmented route. Otherwise, a new route is created, insertion costs of unrouted customers with respect to this route are computed, and the algorithm returns to Step 4.

Note that when a satellite facility is inserted, we do not insert the current customer but return to Step 4 and begin anew. Inserting the customer as well as the satellite facility was found to be an inferior strategy. The entire procedure is repeated M times and the best result saved. After every N iterations, the post-processor is run in an effort to improve the most promising tour in that subset. In the implementation, M is set to 50 and N to 5.

Revised Sweep Algorithm. To use the sweep algorithm, all customers are represented in polar coordinates (θ_i, ρ_i) as measured from the depot, where θ_i is the angle and ρ_i is the ray length. A value $\theta_{i*} = 0$ is assigned to an arbitrary node and the remaining $n - 1$ angles are computed moving in a clockwise direction. The nodes are ranked in increasing order of their θ_i. The algorithm is then applied beginning with node i^* with appropriate logic added so that satellite facilities can be inserted when needed. Because the starting node is arbitrary, the procedure is repeated $2n$ times, once for each node, both in the clockwise and counterclockwise direction. The best result is saved.

Step 1. Let the depot be the origin and calculate polar coordinates of each customer node. Sort all nodes in non-decreasing order of their polar angles. Set $i = 1$.

Step 2. Let k be the index for vehicles and set $k = 1$. Begin assigning customers to vehicles, starting from the ith position on the ordered list.

Step 3. Construct route k in the clockwise direction. That is, consider the list as a circle and go through the list in the clockwise direction.

Step 4. Include customers one by one as long as both capacity and maximum tour time are not exceeded. If the remaining capacity cannot serve the next customer on the list, go to Step 5. If the total tour time is exceeded and there are some unrouted customers, put $k \leftarrow k + 1$ and go to Step 3; otherwise, update the incumbent and go to Step 6.

Step 5. Find the satellite facility with the smallest insertion cost. Insert it in the current route if the maximum tour time is not violated and go to Step 4. Otherwise, put $k \leftarrow k + 1$ and go to Step 3.

Step 6. If $i < n$, put $i \leftarrow i + 1$ and go to Step 2; otherwise put $i \leftarrow 1$, go to Step 2 and repeat the process in the counterclockwise direction. Run post-processor after every N iterations. Stop when finished.

In the description of the algorithm, it is convenient to consider the list of polar angles as a circle that is constructed by first ordering the customers in a line from smallest to largest θ_i and then bending the line to form a circle. The tour construction is primarily done in Steps 4 and 5. In Step 4, we start with an empty vehicle and assign customers on the list to the current route as long as the capacity and maximum tour-time constraints are not violated and there is still one satellite facility that can be visited. If the maximum tour time is exceeded, a new route is initiated and the algorithm returns to Step 3. If only the capacity constraint is violated and conditions permit, we go to Step 5 to find a satellite facility with the minimal insertion cost. This facility is appended to the route and we continue to add customers by returning to Step 4.

The danger in this procedure is that it can lead to poor results when two customers with the same angle are not close to one another (see HAIMOVICH and RINNOOY KAN, 1985). In such instances, we rely on the post-processor to improve the initial solution by swapping customers within or between tours.

When all the customers have been routed, we update the incumbent and go to Step 6. The entire process is repeated starting with the second customer on the list and so on until n sets of routes have been constructed. We then repeat the process another n times, again starting with customer i^*, but scanning the list in the reverse (counterclockwise) direction. The post-processor is run after every N (= 6) iterations (i.e., after completing 3 iterations in one direction and 3 in the other).

3.3 Post-Processing Procedure

As is true with virtually all heuristics designed to solve difficult combinatorial optimization problems, there is no guarantee that the solutions obtained in the construction phase of our heuristics will be locally optimal with respect to common neighborhood definitions. In an effort to improve the results, we have implemented a post-processor that executes arc interchanges within a particular tour and node interchanges between any two tours. In all cases, we work with the directed graph. Before giving the steps, each procedure is outlined.

Arc Interchange. The arc interchange heuristic was originally developed by LIN (1965) to improve TSP tours. In his work, he introduced the concept of

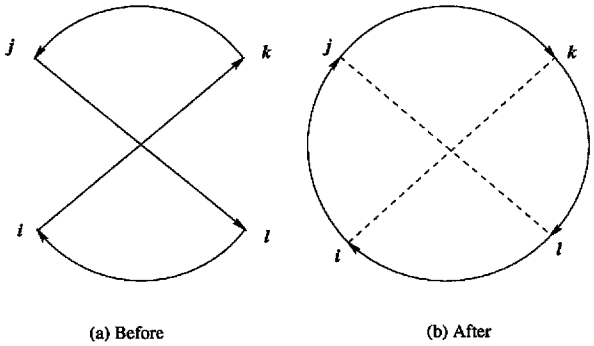

Fig. 2. Two-arc exchange in a route.

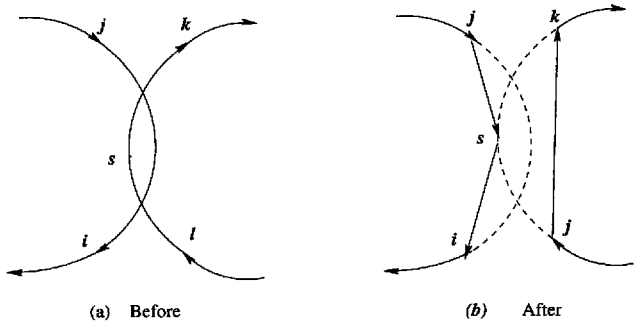

Fig. 4. One-node exchange between two routes.

r-optimality. A given tour is r-optimal if it is impossible to obtain a tour with smaller cost by replacing a set of r of its arcs by any other set of arcs. By doing arc interchanges, only subsequences of nodes in a tour will change. Lin found that for the TSP the 4-arc exchange heuristic did not find noticeably better solutions than the 3-opt procedure.

Two-arc interchange. In this approach, exchanges between two arcs within the same route are performed and the savings calculated. The idea is illustrated in Figure 2. Before the interchange, the route is $i - k - j - l - i$ (Fig. 2*a*). The interchange takes place between arcs ik and jl, and arcs kl and ij. As shown in Fig. 2*b*, the new route is $i - j - k - l - i$. The dash lines represent the old arcs. Notice that there is a direction change on arc kj.

The savings for the 2-arc interchange is calculated as follows,

$$\mathrm{OS}(i, j, k, l) = \tau_{ik} + \tau_{jl} - \tau_{ij} - \tau_{kl}.$$

If OS $(i, j, k, l) > 0$, the interchange reduces the routing cost.

Three-arc interchange. In this case, exchanges among three arcs within the same route are performed and the savings calculated. An example is shown in Figure 3. The original route, $i - l - m - j - k - n -$ i, is displayed at the left (Fig. 3*a*).

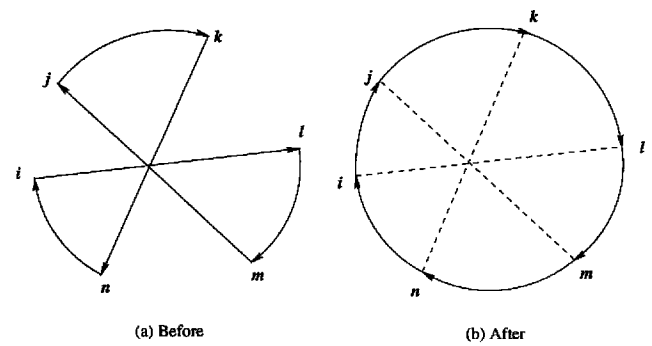

Fig. 3. Three-arc exchange in a route.

Interchanging $i - l$ with $i - j$, $m - j$ with $m - n$ and $k - n$ with $k - l$ gives a new tour $i - j - k - l - m - n - i$ (see Fig. 3*b*).

The savings associated with the 3-arc interchange is

$$\mathrm{OS}(i, j, k, l, m, n) = \tau_{ij} + \tau_{kl} + \tau_{mn} - \tau_{il} - \tau_{kn} - \tau_{jm}.$$

Again if OS $(i, j, k, l, m, n) > 0$, the interchange will lead to an improvement in the routing.

Node Interchange. A second set of procedures involves moving one or more nodes to another position in a tour or to another tour. Similarly, it is also possible to remove a subset of nodes from one tour and place it in another or to exchange subsets of nodes between tours. In our implementation, we allow nodes to be eliminated from one tour and inserted into another as long as feasibility is maintained. Only one- and two-node interchanges are considered. We do not consider interchanges within a tour because of the similarity of such a procedure with arc interchanges.

One-node interchange. The idea is to delete one node from a particular route and insert it into another route if the result is feasible. The operation is shown in Figure 4. On the left side of the figure (Fig. 4*a*) there are two subtours $i - j$ and $k - s - l$. After deleting node s from the latter and inserting it into the former, two new subtours are created, $i - s - j$ and $l - k$ (see Fig. 4*b*).

The savings for the one-node interchange is determined from

$$\mathrm{OS}(i, j; k, s, l) = \tau_{ij} + \tau_{ks} + \tau_{sl} - \tau_{is} - \tau_{sj} - \tau_{kj}.$$

From this calculation, it is clear that a one-node interchange is equivalent to a three-arc exchange. Under certain circumstances, however, the former can lead to the elimination of a route.

Two-node interchange. In this procedure, we swap the positions of two nodes in two different routes. The graph in Figure 5 illustrates the interchange. The original two tours, $i - r - j$ and $k - s - l$, are

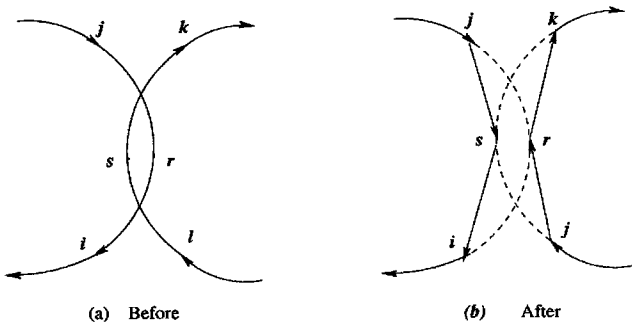

Fig. 5. Two-node interchange between two routes.

shown on the left side of the figure (Fig. 5*a*). After swapping the positions of node r and s, two new tours formed, $i - s - j$ and $k - r - l$ (see Fig. 5*b*).

The corresponding savings is

$$OS(i, s, j; k, r, l) = \tau_{jr} + \tau_{ri} + \tau_{ks} + \tau_{sl} - \tau_{js} - \tau_{si} - \tau_{kr} - \tau_{rl}.$$

Similarly, this result is equivalent to the four-arc exchange.

When routes do not contain satellite facilities, arc interchanges do not affect feasibility with respect to capacity, and hence, are accepted if they lead to a reduction in overall distance. When satellite facilities are included in a tour, the situation is more complicated. In this case, arc interchanges may alter the composition of those route segments that start and end at either a satellite facility or the depot. These new route segments must be checked for feasibility with respect to both capacity and time. Node interchanges between tours always require such checks. Infeasible arc and node interchanges are discarded.

We are now in a position to outline our post-processor.

Post-Processor Interchange Algorithm.

Step 1. For each route in turn, do 2-arc interchange within the route, checking each new route segment for feasibility with respect to time and capacity, if necessary. Ignore infeasible interchanges. Continue until 2-optimality has been achieved.

Step 2. Do 3-arc interchange within each route, making appropriate feasibility checks with respect to time and capacity, if necessary. Check if new routes are still 2-optimal. If not go back to Step 1; else, go to Step 3.

Step 3. Calculate the savings for all feasible 1- and 2-node interchanges between routes. Store those interchanges associated with positive savings. If none are positive, stop.

Step 4. Sort the savings in nonincreasing order.

Step 5. Take the first element on the list and execute the interchange. If the list is empty go to Step 1 and continue until no improvement is possible; otherwise, go to Step 6.

Step 6. Update the savings and go to Step 4.

The above procedure is dominated by the 3-arc interchange at Step 2, which has complexity $O(n^3)$. As a consequence, the exploration of each neighborhood (in search of a better solution) requires up to $O(n^3)$ iterations. Because the algorithm may return to Step 2 many times, its overall complexity is exponential in n, which is typical of most local search procedures. Given this result, it only makes sense to run the algorithm on a subset of the feasible solutions found by the heuristics.

To gauge the effectiveness of the three heuristics and post-processor, we solved the 14 asymmetric VRP instances included in Christofides et al. (1981) and compared the results with the tabu search algorithm of Gendreau et al. (1994). On average, our solutions were 3.9% above theirs. Our run times on a SUN Sparcstation 10, though, where only a few seconds, whereas TABUROUTE required several minutes on a Silicon Graphics workstation.

3.4 Improvement of the Two-Week Schedule

After the VRPSF is solved for each day of the planning horizon, all customers are routed. Assuming the corresponding tours are optimal for the given assignments of customers to days of the week, it still might be possible to reduce total distance by reassigning customers to different days in the planning horizon. The penalty for doing this will be measured by an increase in the annual operating costs. As discussed in Section 3.1, the incremental cost associated with a customer is a function of the day on which he is serviced, so swapping customers between days is likely to result in an increase in this value. (For the case where there are multiple optimal solutions to the original assignment problem (1)–(4), there is a chance that reassigning a customer to a different day and then reoptimizing might lead to a decrease in total distance and incremental cost.)

Now, taking the view that incremental costs are a long-term measure, it is problematic to compare them with the actual routing distances. We therefore keep track of distance and incremental cost and perform a parametric analysis allowing customers to be swapped up to d days in either direction. The objective is to minimize total distance subject to the daily balance constraints and the number of permissible days forward or backward a customer can be

moved. Steps 3 to 6 of the post-processor interchange algorithm have been adapted for the purpose of swapping customers between days.

With regard to the IRPSF, we embed this approach in a rolling horizon framework covering two weeks at a time. The base case involves assigning customers to one of 10 weekdays and then solving the VRPSF for each day. Total distance and incremental costs are computed. We then permit customers to be swapped one day forward or backward; e.g., customers routed on Tuesday can now be reassigned or exchanged with customers on Monday or Wednesday. This is termed a *1-day swap*. The results are recorded and the procedure is repeated for 2- to 5-day swaps. The corresponding points are plotted to reveal the tradeoff between total distance and incremental costs. The plots represent an approximation to the efficient frontier. A necessary condition for assuring that the true efficient frontier has been constructed requires an exact solution to the *d-day swap* VRPSF.

4. NUMERICAL RESULTS

GOLDEN ET AL. (1984) were the first to model the IRP and present a real-world application. The company they dealt with is one of the nation's largest distributors of liquid propane, with over 110,000 customers located in 66 districts. At the time, their annual operating expenses associated with direct distribution activities exceeded $6 million. In the industry, each district operates as an independent unit, typically serving 1500–7000 customers annually within a 25 miles radius of a central depot. In any 2-week period, 300 to 500 customers may require restocking.

In accordance with our recent experience with a similar company, we developed a dataset generator to reflect the characteristics of this operating environment. For purposes of testing, we generate a total of 3000 customers and select a subset of fixed size for the 2-week planning horizon. The decomposition scheme is then applied to route the customers and to develop the approximate efficient frontier.

4.1 Dataset Generation Procedure

We start by assuming that all customers in a district are uniformly located on a 100 × 100 grid. The depot is randomly placed in a 40 × 40 center square of the grid and the satellite facilities are located outside this square in four possible positions. Depending on the number, each is randomly placed in a 10 × 10 square at the various corners of the grid. A problem instance may have up to four satellite facilities, but the typical number is one or two for a propane distribution company.

The fleet is assumed to be homogeneous with each vehicle having a capacity of 2000 gallons. We define five different customer tank sizes: ⅓ truck capacity (5% of customers), ¼ truck capacity (5% of customers), ⅕ truck capacity (40% of customers), 1/10 truck capacity (40% of customers), and 1/20 truck capacity (10% of customers).

The daily consumption rate of each customer is a normally distributed variable whose mean ranges from 1 and 10. The mean is specified as a function the tank size. The larger the tank size, the greater the consumption rate. For example, when the tank size is ⅓ of the truck capacity the mean rate is uniformly distributed in the interval [6, 10]. In all cases, the coefficient of variation (σ/μ) is uniformly distributed between 0 and 1. Each customer is randomly assigned an initial inventory from a uniform distribution between zero and their tank size. Their optimal service frequency d^* is determined by the procedures described in (Jaillet et al., 1997). Of the 3000 customers in the data set, n are selected by checking each day in the 2-week planning period for the likelihood of stockout. If the probability that a customer will run out of commodity by the end of the day is greater than 90%, he is included in the pool. If we want a problem instance that contains, say 400 customers, we randomly select 400 entries from the pool.

The unloading time at a customer is assumed to be 10 minutes, which includes restocking the tank and minor paperwork. Although the actual time depends on the amount pumped, field experience indicates that the variation from one customer to another is minimal. The loading time at a satellite facility is taken as 30 minutes, which is the time to refill the vehicle. Similarly, this value depends on the residual amount of propane on the truck but is nearly constant in practice. The average speed of a vehicle is 30 miles/hour and each route is limited to 8 hours/day. To avoid an imbalance of customer demand on any day of the week, recall that we solve a generalized assignment problem (1)–(4) that minimizes total incremental cost. On each day of the planning horizon then, we end up with approximately equal amounts of commodity to deliver.

4.2 Comparison of Results for Different Heuristics

To evaluate the proposed procedures, we have solved problems containing up to 5 satellite facilities and 500 customers. Using the dataset generation scheme described above, we obtained a pool of 838 customers for the current 2-week period. We then

TABLE I

Comparison of IRPSF Heuristics

No.	Dimension			CW		GRASP		SWEEP	
	n	s	df Δ	Distance	Cost	Distance	Cost	Distance	Cost
1	300	3	0	6573	87	6783	87	6879	87
			1	5587	342	5587	339	5459	177
			2	5271	347	5256	320	5113	277
			3	4942	722	5189	512	4960	871
			4	4847	1588	4992	750	4841	1089
			5	4818	1654	4852	1959	4828	1177
2	350	3	0	7245	191	7447	191	7756	191
			1	6597	266	6578	351	6504	470
			2	6016	898	6179	702	5974	950
			3	5635	1010	5997	925	5784	944
			4	5635	1010	5997	925	5705	1422
			5	5463	1829	5878	1459	5495	1718
3	400	3	0	7986	82	8353	82	8467	82
			1	7317	186	7448	319	7093	391
			2	6819	671	6966	509	6722	673
			3	6505	938	6678	991	6474	1127
			4	6405	1576	6453	1930	6307	1657
			5	6327	1612	6350	1577	6146	2177
4	450	3	0	8665	182	9274	182	9449	182
			1	7616	366	8261	379	8177	480
			2	7335	360	7707	914	7791	510
			3	7192	897	7572	1233	7503	1241
			4	7044	1298	7540	1525	7503	1241
			5	6691	2320	7221	3131	72760	2866
5	500	3	0	9424	133	10054	133	10228	133
			1	8406	292	8894	265	8539	406
			2	7702	446	8584	537	8271	870
			3	7512	1067	8194	790	7892	1402
			4	7512	1067	8141	1429	7703	2270
			5	7276	2136	7753	2159	7681	2821

randomly selected 300, 350, 400, 450, and 500 customers from the 838 to form five data sets (available from the authors).

All codes were written in C and run on a SUN Sparcstation 10. Table I presents the results for the three heuristics—revised CW algorithm, GRASP, and modified SWEEP—for the case of 1 central depot and 3 satellite facilities. Column 1 gives the problem set number and column 2 states the dimensions of the problem; n is the number of customers and s is the number of satellite facilities. In the third column, *df* stands for the degrees of freedom of swapping customers among days. When $\Delta = 0$, customers can only be moved within the current day; when $\Delta = 1$, customers can be moved up to one day in either direction, and so on. Note that it is not permissible to assign a customer to either Saturday or Sunday.

Columns 4 to 6 display the computational results. Dist denotes the total routing distance over the 2-week (10-day) planning horizon for all n customers and cost denotes the accompanying incremental cost. The first row in the table is typical of the overall results. Here it is seen that CW outperforms the other two heuristics by a slight margin. Also, GRASP does better than SWEEP, but, on balance, the score between these two evens out. Because $\Delta = 0$, no swapping among days is allowed, so the incremental costs are all identical; i.e., cost = \$87.

As we go down the rows from $\Delta = 0$ to $\Delta = 5$, the tradeoff between routing distance and incremental cost becomes more apparent. For CW, total distance drops from 6573 miles to 4818 miles (27%), while costs go up from \$87 to \$1654, almost a factor of 20. Ideally, we would like both distance and cost to be as low as possible. Figure 6 plots the approximate efficient frontier for these two parameters for each of the three heuristics. This view visually demonstrates the superiority of CW. The break points on the curves represent the data in Table I for Problem No. 5. Note that the third and fourth data points are the same for CW. The largest incremental reduction in distance is achieved by allowing 1-day swaps. As the degrees of freedom increase, incremental costs rise as expected and distance continues to shrink but at a decreasing rate. Diminishing returns set in

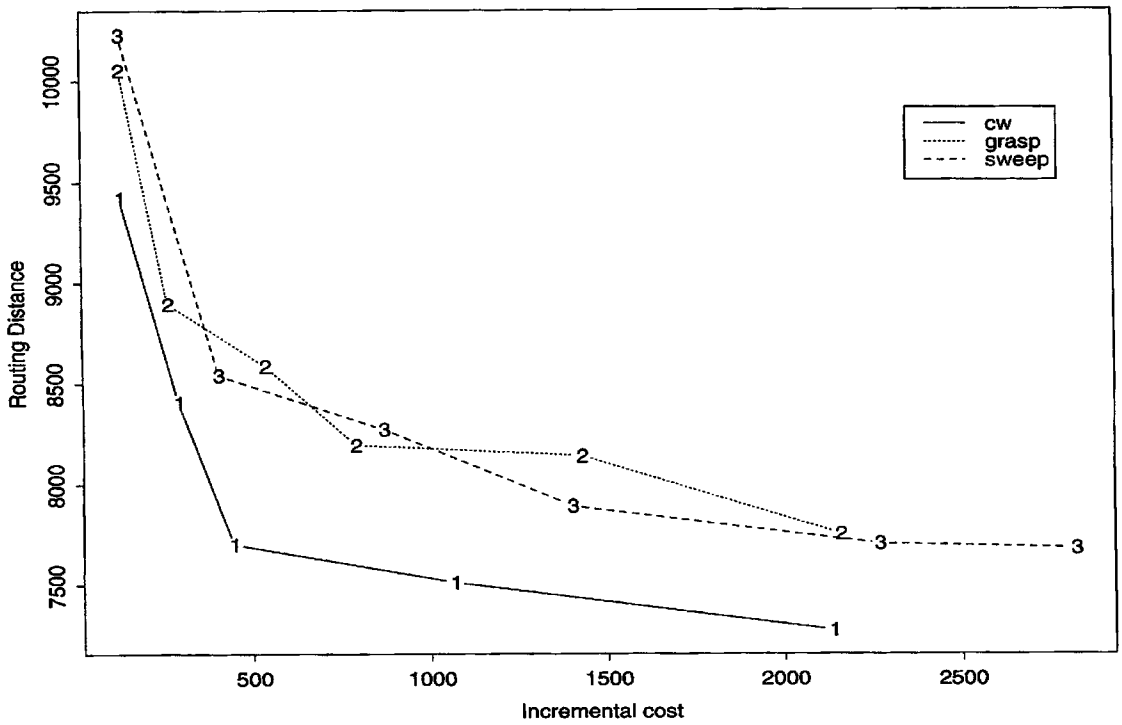

Fig. 6. Approximate efficient frontier for 500 customers, 3 satellite facilities.

quickly. This can be explained qualitatively as follows. The constraint region expands in proportion to the degrees of freedom with respect to feasible transitions but remains fixed with respect to daily balance, route time, and vehicle capacity. Permitting 1- and 2-day swaps leads to relatively large reductions in distance due to the greater number of feasible transitions. As the degrees of freedom increase, though, it is increasingly more difficult to achieve reductions without coming up against the limits of the other constraints. This results in part from the uniform distribution of customers over the feasible region. Allowing more swap days does not proportionately increase the number of attractive exchanges available for all routes on all days within the balance, time, and capacity bounds. Whereas it may be possible to improve one or more routes on any one day in the planning horizon, it is not possible to improve all routes on all days in proportion to the degrees of freedom.

The next issue examined relates to the computational burden posed by each heuristic. Table II presents the CPU time in seconds required for each phase of the analysis. For Problem No. 1, for instance, CW took a total of 89.41 seconds to construct solutions for all 10 days and 0.30 seconds for post-processing. Thus, an average of 8.97 seconds was needed to solve each VRPSF.

Comparing the three heuristics for the particular parameter settings used, we see that CW takes about 3 times longer than GRASP and anywhere from 1.25 to 3 times longer than SWEEP to construct and improve the 1-day solutions. As expected, the d-day swap computations are about the same in each case because they are independent of the VRP heuristics. The real computational burden is in the tradeoff analysis, where we generate the approximate efficient frontier by performing these swaps. For the 500-customer problems, CPU times may approach 2 hours.

The final issue investigated concerned the value of the satellite facilities. Figure 7 plots total distance versus number of satellite facilities for the 500-customer case. The revised CW algorithm was used to obtain the solutions. Not surprisingly, the results show that the routing distance is a nonincreasing function of the number of satellite facilities. Here, as in almost all other instances, there was a proportional advantage in having more satellite facilities.

5. CONCLUSIONS

THE INVENTORY ROUTING problem with its many aspects represents a challenge to researchers and practitioners alike. In this paper, we have presented a methodology that allows us to decompose the problem over the planning horizon and then solve daily rather than multi-day VRPs. In so doing, we devel-

TABLE II

Comparison of CPU Time (sec) for IRPSF Heuristics

Heuristic	Phase	Problem 1 n = 300	Problem 2 n = 350	Problem 3 n = 400	Problem 4 n = 450	Problem 5 n = 500
CW	Construction	89.41	112.16	163.99	199.86	277.11
	Post-processing	0.30	1.07	1.39	1.82	2.12
	1-day swap	96.29	85.75	123.36	308.73	306.32
	2-day swap	201.40	283.98	406.87	589.31	760.49
	3-day swap	313.29	610.91	672.67	974.38	1247.71
	4-day swap	446.58	523.08	920.46	1224.31	1279.28
	5-day swap	562.88	911.48	933.50	1847.16	1893.44
	total CPU (sec)	1710.16	2628.43	3222.20	5145.57	5766.49
GRASP	Construction	29.62	40.20	51.70	69.45	75.95
	Post-processing	0.26	0.34	1.39	0.90	1.25
	1-day swap	89.48	111.14	146.22	203.05	281.68
	2-day swap	224.62	253.08	436.88	583.07	538.84
	3-day swap	290.58	413.55	705.81	918.81	1092.13
	4-day swap	395.38	386.85	1001.58	1120.96	1368.11
	5-day swap	495.50	550.65	1104.25	1725.25	2051.88
total CPU (sec)		1525.46	1755.84	3447.86	4621.52	5409.87
SWEEP	Construction	42.17	44.26	85.97	162.50	156.99
	Post-processing	0.29	1.02	1.72	2.12	3.10
	1-day swap	117.29	135.88	212.86	281.63	380.72
	2-day swap	219.38	408.19	504.39	557.74	805.88
	3-day swap	358.89	526.69	754.62	927.09	1464.28
	4-day swap	512.56	705.19	1045.16	1225.36	1761.25
	5-day swap	470.62	990.50	1279.92	1886.95	2336.49
	total CPU (sec)	1721.17	2811.60	3884.66	5043.42	6908.72

oped and tested three heuristics for the VRP with satellite facilities. In general, the computations showed that the randomized CW algorithm outperformed the GRASP and Modified Sweep algorithm, but that all three procedures are computationally sufficient for real-time decision making. In reality,

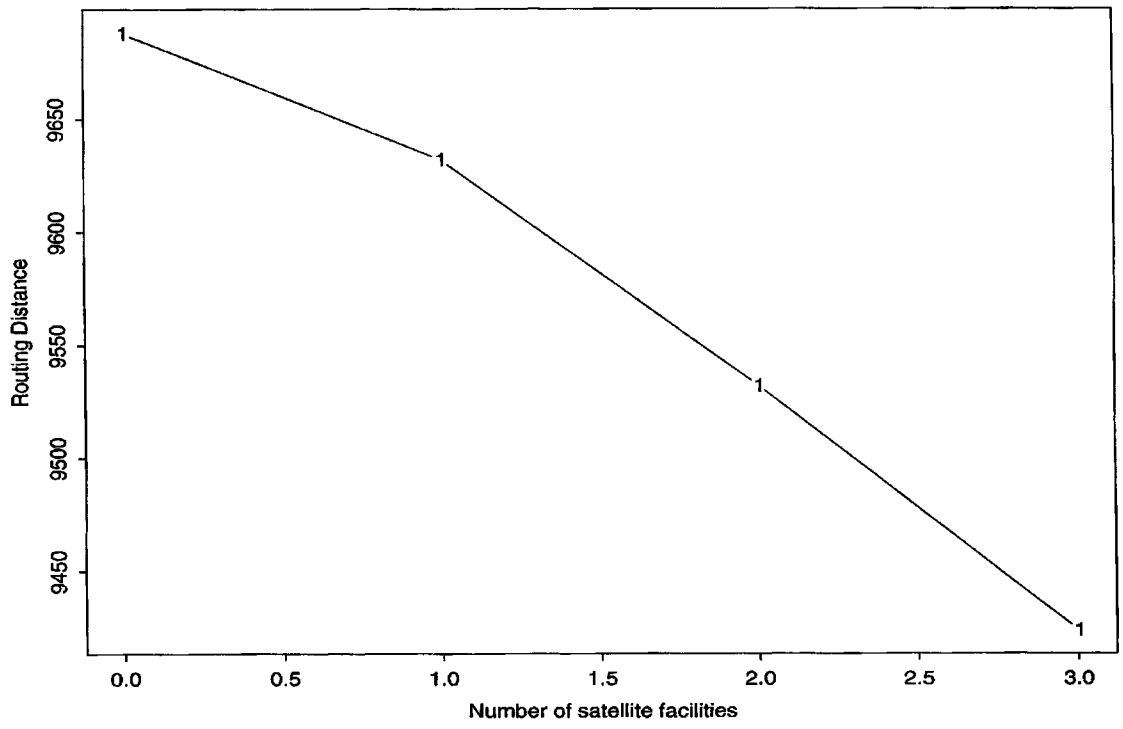

Fig. 7. Value of satellite facilities.

the routing component of the IRP has to be re-solved dynamically from day to day to account for revised forecasts, actual stockouts or other disturbances to the system. Obtaining high-quality solutions quickly is what drives practical applications.

REFERENCES

S. ANILY AND A. FEDERGRUEN, "One Warehouse Multiple Retailer Systems with Vehicle Routing Costs," *Management Sci.* **36(1),** 92–114 (1990).

S. ANILY AND A. FEDERGRUEN, "Two-Echelon Distribution Systems with Vehicle Routing Costs and Central Inventories," *Opns. Res.* **41,** 37–47 (1993).

E. BELTRAMI AND L. BODIN, "Networks and Vehicle Routing for Municipal Waste Collection," *Networks* **4,** 65–94, (1974).

T. W. CHIEN, A. BALAKRISHNAN, and R. T. WONG, "An Integrated Inventory Allocation and Vehicle Routing Problem," *Transp. Sci.* **23,** 67–76 (1989).

N. CHRISTOFIDES AND J. E. BEASLEY, "The Period Routing Problem," *Networks* **14,** 237–256 (1984).

N. CHRISTOFIDES, A. MINGOZZI, AND P. TOTH, "Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations," *Math. Program.* **20,** 255–282 (1981).

G. CLARKE AND J. W. WRIGHT, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Opns. Res.* **12,** 568–581 (1964).

M. DESROCHERS, J. DESROSIERS, AND M. M. SOLOMON, "A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows," *Opns. Res.* **40,** 342–355 (1992).

M. DROR AND M. BALL, "Inventory/Routing: Reduction from an Annual to a Short-Period Problem," *Naval Res. Logist.* **34,** 891–905 (1987).

M. DROR, M. BALL, AND B. GOLDEN, "A Computational Comparison of Algorithms for the Inventory Routing Problem," *Ann. Opns. Res.* **4,** 3–23 (1985).

M. DROR, G. LAPORTE, AND P. TRUDEAU, "Vehicle Routing with Stochastic Demands: Properties and Solution Frameworks," *Transp. Sci.* **23(3),** 166–176 (1989).

M. DROR, AND L. LEVY, "A Vehicle Routing Improvement Algorithm Comparison of a 'Greedy' and a Matching Implementation for Inventory Routing" *Comput. Opns. Res.* **13(1),** 33–45 (1986).

M. DROR, AND P. TRUDEAU, "Stochastic Vehicle Routing with Modified Savings Algorithm," *Eur. J. Oper. Res.* **23,** 228–235 (1986).

A. FEDERGRUEN, AND P. ZIPKIN, "A Combined Vehicle Routing and Inventory Allocation Problem," *Opns. Res.* **32(5),** 1019–1037 (1984).

M. L. FISHER, A. J. GREENFIELD, R. JAIKUMAR, AND P. KEDIA, "Real Time Scheduling of a Bulk Delivery Fleet: Practical Application of Lagrangian Relaxation," Working paper 82-10-11, Department of Decision Science, The Wharton School, University of Pennsylvania, Philadelphia, 1983.

M. GENDREAU, A. HERTZ, AND G. LAPORTE, "The Tabu Search Heuristic for the Vehicle Routing Problem," *Management Sci.* **40(10),** 1276–1290 (1994).

B. GILLETT AND L. MILLER, "A Heuristic Algorithm for the Vehicle Dispatch Problem," *Opns. Res.* **22,** 340–349 (1974).

B. GOLDEN, A. ASSAD, AND R. DAHL, "Analysis of a Large Scale Vehicle Routing Problem with an Inventory Component," *Large Scale Syst.* **7,** 181–190 (1984).

M. HAIMOVICH AND A. H. G. RINNOOY KAN, "Bounds and Heuristics for Capacitated Routing Problems," *Math. Opns. Res.* **10(4),** 527–542 (1985).

L. HUANG, "The Inventory Routing with Satellite Facilities Problem," PhD. dissertation, College of Business Administration, University of Texas, Austin, 1997.

P. JAILLET, L. HUANG, J. F. BARD, AND M. DROR, "A Rolling Horizon Framework for the Inventory Routing Problem," Working paper, Department of Management Science and Information systems, University of Texas, Austin, 1997.

G. KONTORAVDIS AND J. F. BARD, "A Greedy Random Adaptive Search Procedure for the Vehicle Routing Problem with Time Windows," *ORSA J. Comput.* **7(1),** 10–23 (1995).

G. LAPORTE, "The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms," *Eur. J. Oper. Res.* **59,** 345–358 (1992).

G. LAPORTE, H. MERCURE, AND Y. NOBERT, "Exact Algorithms for the Asymmetrical Capacitated Vehicle Routing Problem," *Networks* **16,** 33–46 (1986).

G. LAPORTE, AND Y. NOBERT, "Exact Algorithms for the Vehicle Routing Problem," *Ann. Discrete Math.* **31,** 147–184 (1987).

R. C. LARSON, "Transporting Sludge to the 106-Mile Site: An Inventory/Routing Model for Fleet Sizing and Logistics System Design," *Transp. Sci.* **22(3),** 186–198 (1988).

L. LASDON, *Optimization Theory for Large System*, McMillan, New York, 1970.

S. LIN, "Computer Solutions of the TSP," *Bell Syst. Tech. J.* **44,** 2245–2269 (1965).

I. R. WEBB AND R. C. LARSON, "Period and Phase of Customer Replenishment: A New Approach to the Strategic Inventory/Routing Problem," *Eur. J. Oper. Res.* **85,** 132–148 (1995).

A. WREN, *Computers in Transport Planning and Operation*, Ian Allan, London, 1971.

(Received: November 1996; revisions received: September 1997; accepted: November 1997)