TRANSPORTATION SCIENCE
Vol. 38, No. 4, November 2004, pp. 488–502
ISSN 0041-1655 | EISSN 1526-5447 | 04 | 3804 | 0488

DOI 10.1287/trsc.1030.0054
© 2004 INFORMS

A Decomposition Approach for the Inventory-Routing Problem

Ann Melissa Campbell
Management Sciences Department, Tippie College of Business, University of Iowa, Iowa City, Iowa 52242,
ann-campbell@uiowa.edu

Martin W. P. Savelsbergh
School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332-0205,
martin.savelsbergh@isye.gatech.edu

In this paper, we present a solution approach for the inventory-routing problem. The inventory-routing problem is a variation of the vehicle-routing problem that arises in situations where a vendor has the ability to make decisions about the timing and sizing of deliveries, as well as the routing, with the restriction that customers are not allowed to run out of product. We develop a two-phase approach based on decomposing the set of decisions: A delivery schedule is created first, followed by the construction of a set of delivery routes. The first phase utilizes integer programming, whereas the second phase employs routing and scheduling heuristics. Our focus is on creating a solution methodology appropriate for large-scale real-life instances. Computational experiments demonstrating the effectiveness of our approach are presented.

Key words: inventory routing; vehicle routing; insertion heuristics; clustering; integer programming
History: Received: October 2001; revision received: October 2002; accepted: October 2002.

1. Introduction

The inventory-routing problem (IRP) is of special interest because it integrates two components of supply chain management: inventory control and vehicle routing. These two issues have traditionally been dealt with separately, but their integration can have a dramatic impact on overall system performance.

The IRP arises where a vendor-managed resupply policy (VMR) is being used. Vendor-managed resupply, or vendor-managed inventory, refers to an agreement between a vendor and his customers in which the customers allow the vendor to choose the timing and size of their deliveries. In exchange for this freedom, the vendor agrees to ensure that the customers do not run out of product. In a more traditional relationship, in which customers call in their orders, large inefficiencies can occur as a result of the timing of the customers' orders. If the vendor negotiates a switch in policy though, this can become a distinct advantage because he can combine deliveries to make more efficient use of resources. It can also be to the advantage of the customers because they no longer have to dedicate resources to inventory management and often receive cost incentives to make the switch. Executing a vendor-managed inventory policy in an effective way, however, is not a simple task, particularly with a large number and variety of customers. Determining a distribution strategy that minimizes long-term distribution costs is the inventory-routing problem and is the problem addressed in this paper. Our focus is on designing and implementing a solution approach capable of solving practical instances of the IRP.

This paper is organized as follows. In §2, we will discuss in more detail the industry problem that motivated our interest in the IRP, and in §3 we will formally define the inventory-routing problem and briefly review the relevant literature. In §4, the two-phase approach we have developed for solving the IRP is introduced with the Phase I and Phase II methodology covered in greater depth in §§5 and 6, respectively. Section 7 reviews sample computational results, and the final section highlights general conclusions and plans for later research.

2. Industry Problem

Our research is motivated, in part, by work done with Praxair (www.praxair.com), an international industrial gases company. Praxair's production process involves taking air and separating it into its component parts, such as oxygen, hydrogen, nitrogen, and argon. The gases are transported in their liquid form in trucks from the plants to the customers. The trucks are product specific because they are engineered to safely carry a particular type of gas.

Praxair has negotiated a vendor-managed resupply policy with most of their customers and wants

to take better advantage of this relationship. However, even though customers let the vendor choose when to deliver, the delivery is often restricted to occur within customer-defined time windows. There are also substantial differences in how much time is required at a customer between the time a truck arrives and when delivery can begin due to customer-specific traits such as the location of the tank and bureaucratic procedures. Timing and sizing deliveries is further complicated by the existence of operating modes. Operating modes occur when customers do not consume product at the same rate 24 hours a day, 7 days a week. Instead, they have different usage rates associated with blocks of time during each day. Each customer can have different time blocks as well as different rates.

In terms of size, Praxair has about 60 production facilities and over 10,000 customers across North America. Some customers require a delivery once every three months, but others require multiple truckloads of product in a single day. These latter customers make delivery planning especially difficult.

Only a limited number of resources are available for distribution, which dramatically impacts the complexity of creating efficient delivery schedules. Both the number of trucks and the number of drivers is limited, and driver availability particularly complicates the planning activities. Driver constraints include restrictions on the number of hours each day that can be spent driving, and, at some production facilities, drivers have predetermined start times. These and other complexities often make finding just a feasible delivery schedule a nontrivial task.

This description of the IRP as it exists at Praxair is representative of the problem and its complexities at many other companies and in many other industries. Other industries with similar issues and complexities practicing vendor-managed inventory policies include

- the petrochemical industry;
- suppliers for supermarkets (Purpura 1997, Radice 1999), such as HEB Grocery (Ross 1998);
- department store chains, including Walmart (Mongelluzzo 1998);
- home products, such as Rubbermaid (Brumback 1995);
- the clothing industry, where VMR is encouraged by the American Apparel Manufacturers Association (Nannery 1994); and
- the automotive industry (parts distribution).

The number of these industries seems to be increasing, along with the need for approaches to the IRP that handle the additional constraints and influential complexities present in practical versions of the problem.

3. Inventory Routing

3.1. Problem Definition

To develop a strategy for managing a vendor-managed resupply policy, we start by examining the core of this problem that is the IRP.

The IRP is a variation of the well-studied vehicle-routing problem (VRP). The vehicle-routing problem is a daily problem. Customers place orders and the delivery company assigns the orders for that day to routes/trucks so as to minimize total distance traveled. In the IRP, however, the delivery company, not the customer, decides how much to deliver to which customers each day. There are no customer orders. Whereas vehicle-routing problems typically deal with a single day, inventory-routing problems have to deal with a longer horizon. All decisions made today must be made keeping in mind their impact on what has to be done in the future.

More specifically, the IRP is concerned with the repeated distribution of a single product from a single facility to a set of N customers over a given planning horizon of length T, possibly infinity. Customer i consumes the product at a given rate U_i (volume per day) and has the capability to maintain a local inventory of the product up to a maximum of C_i. The inventory at customer i is I_i^0 at Time 0. A fleet of M homogeneous vehicles, with capacity Q, is available for the distribution of the product. The objective is to minimize the average distribution costs during the planning period without causing stockouts at any of the customers. Three decisions have to be made:

- when to serve a customer,
- how much to deliver to a customer when served, and
- which delivery routes to use.

This basic model of the problem assumes that usage is deterministic and that there is an unlimited amount of the product available at the plant. Deliveries are not restricted by customer-specified time windows, and the usage rate of product, U_i, is assumed to be a constant rate through the day.

There are some variations in how the basic inventory-routing problem is defined, with most of the variation stemming from how inventory costs are handled. In the case where the customer is not part of the same company as the vendor, like at Praxair and many of our other examples, and the objective is to minimize distribution costs for the vendor only, it is not relevant to include inventory holding costs at the customers in modeling the problem. Therefore, these costs are not present in our definition of the problem.

3.2. Related Work

We will not review all of the related literature here. There are several extensive literature reviews, such as those included in Ball (1988), Campbell et al. (1998),

Dror et al. (1985), and Nori (1999). We will instead highlight certain concepts and trends to show how our approach fits into the literature.

Most of the research on the IRP is in one of three directions:

- single-day model using deterministic or stochastic demand;
- multiday model using deterministic or stochastic demand; or
- permanent routing, usually for long-term planning purposes.

The early work focused on single-day models, where the IRP is optimized in single-day slices. Minimizing distance over a horizon of one day was found to be very myopic, postponing all deliveries except those necessary today. It leaves too many deliveries for future days, eventually creating infeasibilities, and overlooks good opportunities today. Single-day approaches simplify the problem greatly though, which made them popular initially. Examples include Beltrami and Bodin (1974), Federgruen and Zipkin (1984), and Golden et al. (1984).

More recently, multiday models, like the one we are proposing, have become more prevalent. Though they are computationally more intensive, they tend to produce better-quality solutions. In multiday models, one of the key features that distinguish solution approaches is how long-term effects of short-term decisions are modeled and how customers are selected for a short-term version of the problem.

One of the more prominent bodies of work in solving the inventory-routing problems for a small number of days was created by Dror, Ball, and Golden (Dror et al. 1985, Dror and Ball 1987) and was extended to consider more of the long term by Jaillet, Bard, Huang, and Dror in Jaillet et al. (2002), Bard et al. (1998a, b). We will discuss this work in some detail because it has many similarities with ours, but also has some stark contrasts. Because the actual problem involves stochastic usage rather than constant usage, Dror et al. (1985) use a normal random variable with known parameters to represent customer usage. Given these parameters, they compute an optimal replenishment day for each customer (considering the customer in isolation). On this day, the customer will have a low inventory before delivery, but still has a positive probability that he will not run out of product before the delivery. In Jaillet et al. (2002), only those customers with optimal replenishment days within the next two weeks are considered. An allocation problem is solved, creating small changes for some customers in the day they will receive delivery from the optimal replenishment day to another day to prevent too large of a demand on any day in the two-week horizon. After the customers are allocated to days, the result is a set of daily vehicle-routing problems where the delivery quantity used is a quantity that approximates what will be necessary to "fill" each customer on that day. This approach is embedded within a rolling-horizon framework, in which only the first week of the schedule is used before the problem is resolved. The rolling-horizon framework allows the schedule for the first week (which is actually implemented) to be influenced by the future in the form of the second week.

Permanent routing, or periodic routing, involves creating a p-day schedule that can be repeated indefinitely. These approaches, as described in Christofides and Beasley (1984) and Gaudioso and Paletta (1992), are better for making strategic decisions such as determining fleet size rather than short-term planning. The true stochastic nature of the problem makes these approaches unsuccessful and inefficient in the short term.

Other relevant results include the evaluation by Gallego and Simchi-Levi (1990) of the long-run effectiveness of direct shipping (separate loads to each customer). They conclude that direct shipping is at least 94% effective over all inventory-routing strategies whenever minimal economic lot size is at least 71% of truck capacity. The effectiveness deteriorates as economic lot size gets smaller. Because our primary consideration is instances of the IRP where there is a large variety in customer capacity and consumption, it will not be efficient to consider direct shipping for all customers here.

In the last few years, several researchers have started to focus even more on the stochastic nature of product usage. In Kleywegt et al. (2002a, b) the stochastic inventory-routing problem (SIRP) is introduced and modeled as a Markov decision process and approximate dynamic programming approaches are developed. Although the initial results are interesting and promising, the solution approach is still computationally prohibitive for realistic instances of the IRP. Furthermore, in practice it is difficult to obtain the necessary probability distribution information to represent the problem correctly.

4. Problem Decomposition

Our approach combines a number of existing ideas and complements these with insights obtained from witnessing how IRPs are solved in practice.

To keep computation times within acceptable limits, we have adopted a fundamentally deterministic approach. However, recognizing the stochastic characteristics of the input data and the long-term nature of the problem, we embed our approach in a rolling-horizon framework. Another key aspect of our approach is that we want to carefully consider the long-term impact of short-term decisions. As a result, we want to solve the problem over a planning horizon that is as long as possible, subject to the condition

that we want to be reasonably sure that the available data accurately represent the expected product consumption during the planning period. The selection of an appropriate planning horizon may differ widely across products, facilities, and industries. The longer the planning horizon we can consider, the higher the quality and reliability of the schedule produced for the first few days.

In our rolling-horizon framework, we plan on using only the first j days of the k-day schedule being constructed. The value of j should be relatively small so as to be able to take advantage of new and updated information on inventory levels and usage rates. The value of j will typically be about two days for vendors to give drivers some advance notice of their schedules.

As in Jaillet et al. (2002), we decompose the solution process into two phases. We make decisions hierarchically: first, decisions for the longer k-day horizon, and second, decisions for the shorter j-day horizon (the portion of the plan that will actually be executed). The first phase will focus on assigning customer deliveries to days in the planning horizon; the second phase will focus on constructing delivery routes. How these two phases are structured, though, is fundamentally different from what is proposed in Jaillet et al. (2002). (An early version of our approach can be found in Campbell et al. 1998.)

Basing the assignment of deliveries to days based on information about the day that an individual customer runs out of product, as in Jaillet et al. (2002) does maximize the volume deliverable to a customer, but is not necessarily the best option in terms of long-term distribution costs because it does not recognize the synergies that may exist between customers. It may happen that two customers that are geographically close (and thus can be served on a single trip), will never be routed together because of their initial inventories and their usage rates (optimal replenishment days differ). If the problem is addressed differently, as we propose to do, they may be routed together, and this weakness corrected. We focus much more on proximity of customers in deciding the replenishment day. This is a fundamental change from how the IRP has been solved previously, and we think a fundamental improvement.

Phase I considers a coarse approximation of the problem, with daily decisions, over a k-day planning horizon. Based on the results of Phase I, Phase II considers a model with decision accuracy in terms of minutes rather than days for the first j days. The decisions made in Phase I are not set in stone when we move to Phase II, but are used as guiding suggestions. Details of Phase I are provided in §5 and details of Phase II are provided in §6.

5. Phase I: Planning

In Phase I, we use an integer program to determine a high-level base plan for the next k days indicating which customers to serve on each day and suggesting how much to deliver to them. In this section, we introduce the model, discuss how we modified it to make it solvable in a limited amount of time, and then discuss how it can be extended to include several complexities common in practice.

5.1. Model

Central to the Phase I integer program are two parameters: $LL_i^t = \max(0, tU_i - I_i^0 + l_iU_i)$, a lower bound on the total volume that has to be delivered to customer i by the end of day t; and $UL_i^t = tU_i + C_i - I_i^0 - l_iU_i$, an upper bound on the total volume that can be delivered to customer i by the end of day t. The definitions of LL_i^t and UL_i^t are self-explanatory, except possibly for the use of l_iU_i. The value l_i represents the fraction of a day it takes to travel to customer i, so l_iU_i is the product usage during this time. The idea is that the lower bound should be enough so that the customer will last until he can be reached the next day, and the upper bound should not be larger than the amount that can fit at the customer with time to return to the depot by the end of the day. Because the timing is actually set in Phase II, the use of the l_i terms is of importance primarily for the customers that are far from the depot. Let d_i^t represent the delivery volume to customer i on day t. To ensure that no stockout occurs at customer i and to ensure that we do not exceed the inventory capacity at customer i, we need to require

$$LL_i^t \le \sum_{1\le s\le t} d_i^s \le UL_i^t \quad \forall i\ \forall t. \tag{1}$$

To model resource constraints with some degree of accuracy and to have a meaningful objective function, we found it necessary to introduce the notion of "delivery routes" even in Phase I. However, it should be noted that when we refer to a "route" in the integer program, we will be referring only to a set of customers without enforcing a specific ordering among them; i.e., for the set of customers $\{A, B, C\}$ we do not distinguish between Routes A–B–C, A–C–B, C–A–B, B–A–C, B–C–A, and C–B–A. We estimate the distance required to visit the customers in the set to be the length of the optimal traveling-salesman tour through the customers.

Now, let R be the set of delivery routes, let T_r denote the duration of route r (as a fraction of a day), and let c_r be the cost of executing route r (which we assume depends on the duration of the route r). Furthermore, let x_r^t be a 0-1 variable indicating whether route r is used on day t ($x_r^t = 1$) or not ($x_r^t = 0$) and let d_{ir}^t be a continuous variable representing the amount

of product delivered to customer i on route r on day t (replacing the use of the variable d_i^t).

The total volume that can be delivered on a single day is limited by a combination of capacity and time constraints. Because vehicles are allowed to make multiple trips per day, we cannot simply limit the total volume delivered on a given day to be the sum of the vehicle capacities. The resource constraints can be modeled by (recalling that our fleet has M vehicles of capacity Q)

$$\sum_{i:\, i\in r} d_{ir}^t \le Q x_r^t \quad \forall r \in R \ \forall t \tag{2}$$

and

$$\sum_{r:\, r\in R} T_r x_r^t \le M \quad \forall t. \tag{3}$$

These constraints ensure that we do not exceed the vehicle capacity on any of the selected routes and that the time required to execute the selected routes does not exceed the time available.

The basic integer programming model is thus given by

$$\min \sum_t \sum_r c_r x_r^t, \tag{4}$$

$$LL_i^t \le \sum_{1\le s\le t} \sum_{i:\, i\in r} d_{ir}^s \le UL_i^t \quad \forall i \ \forall t, \tag{5}$$

$$\sum_{i:\, i\in r} d_{ir}^t \le Q x_r^t \quad \forall r \in R \quad \forall t, \tag{6}$$

$$\sum_{r:\, r\in R} T_r x_r^t \le M \quad \forall t, \tag{7}$$

$$x_r^t \in (0, 1). \tag{8}$$

5.2. Phase I: Solving the IP

The Phase I integer program as presented above is not very practical for two reasons: the huge number of possible delivery routes and, although to a lesser extent, the length of the planning horizon. To make the integer program computationally tractable we consider a small (but good) set of delivery routes and aggregate periods toward the end of the planning horizon.

5.2.1. Clusters. Our approach to reduce the number of routes is based on allowing customers to be on a route together only if they are in the same *cluster*. A cluster is a group of customers that can be served cost effectively by a single vehicle for a long period of time.

The following approach is used to identify a good set of disjoint clusters covering all customers:

(1) generate a large set of possible clusters;

(2) estimate the cost of serving each cluster; and

(3) solve a set-partitioning problem to select clusters.

Observe that the selection of clusters only has to be done once as a preprocessing step before the actual planning starts. It does not have to be rerun before every execution of the Phase I integer program. In practice it makes sense to recluster when new customers have been added or there have been significant changes to customer usage patterns.

- *Cluster Generation.* The number of clusters that can be generated may be huge. Therefore, we have developed several heuristic rules, mainly based on usage considerations, to limit the number of possible clusters. For example, five customers that all need a full-truckload delivery per day will not be combined to form a cluster, because we want a cluster to be deliverable by a single vehicle. Likewise, two customers whose combined inventory capacity is significantly less than the size of a truck will not be a cluster either because we want the routes created by the cluster to have a chance to make full-truckload deliveries. Other criteria include basic distance requirements such as making sure the distance for the TSP tour through all the customers in the cluster is below some threshold to ensure that customers in a cluster are geographically close. For feasibility reasons, we always generate clusters containing each customer by itself.
- *Estimating Cluster Cost.* The cost used for a cluster is the distribution cost for serving the customers in the cluster for a month, or another period of time sufficient for all customers to receive a delivery and get an idea of the long-term costs of this particular combination of customers. The cost of serving a cluster should not only depend on the geographic locations of the customers in the cluster, but also on whether the customers in the cluster have compatible inventory capacities and usage rates. Therefore, to evaluate the cost of a cluster, we need a model that considers all of these factors.

Because we generate a large number of clusters, we also need a costing procedure that is fast, but able to provide an accurate estimate of the cost of serving the cluster. We decided after evaluating several models to use the following simple integer program that has the key features represented.

Let c_r denote the cost of an optimal route r through a subset of customers in a cluster. Let y_{ir} be a variable indicating the total volume delivered to customer i on route r in the planning period, and let z_r be a variable indicating the number of times route r is executed in the planning period. Again r is a subset of customers, and R is the set of all routes for the cluster. We will use T for the duration of the cluster-costing model. This leads to the following:

$$\min \sum_{r:\, r\in R} c_r z_r, \tag{9}$$

subject to

$$\sum_{i:\, i\in r} y_{ir} \le \min\left(Q, \sum_{i:\, i\in r} C_i\right) z_r \quad \forall r \in R, \tag{10}$$

$$y_{ir} \le \min(Q, C_i) z_r \quad \forall i \in r\ \forall r \in R, \tag{11}$$

$$\sum_{r:\, i\in r} y_{ir} = TU_i \quad \forall i, \tag{12}$$

$$z_r \text{ integer}, \qquad y_{ir} \ge 0. \tag{13}$$

This model ensures (1) that the total volume delivered on route r in the planning period is less than or equal to the minimum of the vehicle capacity and the total storage capacity times the number of times route r was executed, (2) that we do not deliver more to a customer than the minimum of the vehicle capacity and its inventory capacity times the number of times route r was executed, and (3) that the total volume delivered to a customer in the planning period is equal to its total usage during the planning period of length T.

• *Set Partitioning.* We want to choose a group of clusters such that each customer is in only one cluster and the total cost is minimized. This is a classic set-partitioning problem in which the clusters are the sets.

5.2.2. Reducing the Customer Set. For practical instances of the Phase I integer program, even with the use of clusters, the number of routes can still be huge. This happens primarily when several low-usage customers are located near other customers in the same cluster. These low-usage customers lead to many routes with approximately equal cost and structure. For example, if *A*, *B*, and *C* are low users, near each other, and near *D*, and *D* is far from the depot, routes *D*, *A*–*D*, *B*–*D*, *C*–*D*, *A*–*B*–*D*, *A*–*C*–*D*, *B*–*C*–*D*, and *A*–*B*–*C*–*D* all look reasonable and have almost equal cost. If neither *A* nor *B* needs a delivery any time soon, though, we really only need to consider *D* and *C*–*D*. Eliminating nonurgent low-usage customers from consideration can cause a huge reduction in the number of routes in the Phase I integer program. Therefore, we will carefully restrict the set of customers to be considered. Unlike other proposed approaches in the literature, we will not cut down the set of customers by only considering those requiring an impending delivery. We broaden the set under consideration to capitalize on good routing combinations that reduce long-term costs. We will continue to generate routes only among customers in the same cluster, but we will use the following guidelines to decide which of these customers should be considered in each IP.

• *Critical and Impending Customers.* First, we will classify the customers into two sets: critical and not critical. *Critical* customers will be those that have a large impact on the efficiency of the schedule, including those customers with high demand or very distant customers. We will always want to consider critical customers when making our plan. They consume a large percentage of resources and thus have a significant impact on the schedule.

Second, we can classify the remaining customers into two sets: impending and not impending. *Impending* customers are those, given their current inventory level and usage rates, that require a delivery in the next several days. The idea is that it will not really be beneficial to consider all of the customers that do not need a delivery soon. We will clearly always want to include impending customers in the Phase I integer program.

• *Balance Customers.* Reducing the customer set just to critical and impending customers has a dramatic effect on the number of variables. The Phase I integer programs become much easier to solve, as we anticipated, but the solutions are not always quite what we want. For example, consider the situation such as the one described earlier, where there is a customer *D* far from the depot, who is critical. The only customers near *D* are *A*, *B*, and *C*. If neither *A* nor *B* nor *C* are critical or impending and *D* cannot receive a full truckload, there will not be a good choice for how to visit *D* among the routes in the Phase I integer program. There is no opportunity for *balancing* the load. We are not taking advantage of all of the good routing opportunities we wanted to find. There is a middle ground, though, between including none and all of the customers. We will iteratively add customers to the model based on the set of critical and impending customers. We will create balancing opportunities for all customers that are currently considered in the model. Thus, we can classify the remaining customers into two sets: balance and not balance. *Balance* customers will be those customers that do not need a delivery imminently, but are near and in the same cluster as customers that are critical or impending. We include the h nearest neighbors within the same cluster for every critical or impending customer in the IP that can receive a delivery of a minimum size in the next few days. Including these customers with these attributes improves the chance for the Phase I integer program to plan good truckload routes. Note that these h nearest neighbors often may include customers already in the model, so they will not be added. The full number, h, will be added when a customer is located far from other impending or critical customers. These are ones where adding a good balancing opportunity is key.

Summarizing, we do not consider noncritical, nonimpending, and nonbalance customers in the Phase I integer program. These customers do not consume large amounts of resources, do not need deliveries in

the near future, and are not conveniently located relative to other customers needing deliveries. Including these customers may slightly change the resulting base plan, but at a prohibitively large cost in terms of computation time.

5.2.3. Aggregation/Relaxation. Given that our two-phase solution approach will be embedded in a rolling-horizon framework, the emphasis should be on the quality and detail of the decisions concerning the first few days of the plan. This provides us with an excellent opportunity to reduce the size of the integer program by aggregating days toward the end of the planning period.

For the first l days, with $j < l < k$, we will have route-selection variables for each day, but for the days after that, we will have route-selection variables covering periods of several days. Instead of making a decision on whether to execute each route on Days 8 to 14 individually, for example, we now decide how many times each of the routes will be executed during that whole week instead. Several aggregation schemes were tested. We found that considering weeks rather than days toward the end of the planning horizon still does a good job of preserving the costs associated with the effect on the future and yields a significant reduction in CPU time. Therefore, the daily variables associated with these later days are replaced by weekly variables. Upper and lower bounds can be altered accordingly as well.

The solution for the later weeks is not without value, though. We can still use the solution to get an estimate of expected workload in the weeks ahead, which can help in scheduling vehicle maintenance, driver vacations, and other planning needs.

A further simplification is obtained by relaxing the integrality restrictions on the variables representing the weekly decisions. Therefore, the only binary variables appearing in the integer program will be those representing route selections for the first l days. This makes the planning IP much easier, but still reflects the longer term in the short-term decisions.

5.2.4. Integer Programming Techniques. Even after aggregation and relaxation, the Phase I integer programs can still be difficult to solve to optimality. We make use of branching priorities to find feasible solutions more quickly (by preferring branching on variables early in the planning horizon), and we use the best solution found within a fixed amount of time.

5.3. Extensions

In any hierarchical approach, it is important to strike a proper balance between what aspects of the problem are handled in each of the different phases and at what level of detail. In this section, we discuss extensions to the basic integer program to make Phase I better reflect practical instances of the IRP. Because we are making daily decisions, we can only incorporate problem characteristics that can be modeled at that level of granularity, possibly in aggregate form. Note that we also have to ensure that the integer program remains computationally tractable. The following problem characteristics have been incorporated:

- *Fixed and Variable Stop Times. Fixed stop time* is the time required for a delivery at a customer. It is either considered to be customer dependent (f_i) or the same for all stops (f). *Variable stop time* represents the portion of the delivery time that varies due to the size of the delivery. We use p to represent the delivery rate, or pump rate, such that pQ represents the time to deliver a full truckload of product. We can add the fixed time for all stops plus the variable time for a full truck delivery to the travel time for a route to approximate the new completion time. We redefine T_r' as follows:

$$T_r' = \sum_{i:\, i \in R} f_i + T_r + pQ. \tag{14}$$

- *Operating Modes.* As discussed earlier, operating modes are the blocks of time associated with different usage rates throughout the day. When we know the usage rates over the day, we can use them to compute the total consumption over the day. This alters the computation of the upper and lower bounds.
- *Time Windows.* Time windows for delivery at the customers affects the IP by modifying the lower bound. The lower bound must be sufficient to keep the customer from running out of product before the product can be delivered at the earliest part of the time window. Time windows also can impact the generation of the routes. Routes will only be created with customers that have overlapping time windows to insure deliveries can be made within the same general time frame.
- *Driver Availability.* In practice, the number of drivers working on a given day can vary and can have a dramatic impact on the problem. When the number of drivers are known, we can use this to modify the number of work hours available on a given day beyond the restriction imposed by the number of vehicles.
- *Driver Restrictions.* Because drivers can only be on the road limited hours per day, we only generate routes that can be completed within such a time limit.
- *Order-Only Customers.* Very few companies that implement a VMR program have *all* customers willing to make this change, except for where customers are part of the same company. This can be a result of unwillingness to relinquish control over deliveries, but it can also be because not all customers use product at a regular rate. With this in mind, any successful IRP algorithm must be flexible enough to account for these customers as well as anticipatable customers.

We add the order-only customers for which upcoming orders are already known to clusters and add balance customers for these as well, so routes for order-only customers can be included in efficient full-truckload routes. Orders are usually not known more than a week ahead of time, but we can approximate what percent of resources are typically consumed by these orders. This percentage can be used to appropriately reduce the available vehicle resources for the weeks ahead.

- *Multiple Deliveries Per Day.* Because some customers may require multiple deliveries per day, the described IP may not generate feasible solutions for these customers. For those it does, it may choose a solutions such as route A, A–B, and A–C with nonzero delivery quantities only to A on the last two routes. Keeping in mind the work of Gallego and Simchi-Levi (1990), we replicate the binary variables representing the single customer route the number of times necessary to serve these customers strictly through direct delivery. We still include, though, one copy of each binary variable representing multicustomer routes including these customers. Because these customers are large consumers, direct shipments are appropriate for them in most cases. Multicustomer routes are available for the customer too, however, if there are situations where using a balancing opportunity is more efficient. To help branch and bound find a feasible solution, we require that the binary variable representing the first replication of the single-customer route must be used before the second and so forth.

6. Phase II: Scheduling

The high-level base plan produced in Phase I does not specify departure times and delivery sequences for the different vehicles. We still need to construct vehicle routes and schedules. In this section, we discuss the heuristics used for this and show how they can be modified to include additional complexities.

Because the delivery quantities specified in the base plan may not fit before a specific time of the day or may need to be received before a certain time to prevent a stockout, the deliveries have self-imposed time windows. Therefore, to convert the base plan into an actual daily delivery schedule, we can solve a sequence of vehicle-routing problems with time windows (VRPTW).

Such an approach, which is common in the literature, does not capitalize on the flexibility inherent in the inventory-routing problem. The delivery quantities and times specified by the solution to the Phase I integer program are good from a long-term perspective; they may not be as good from a short-term perspective. Therefore, we treat the delivery sizes and times specified by the solution to the integer programs as suggestions. We want to follow these suggestions as closely as possible, because this helps to achieve our long-term goals, but often some changes are needed. Because some constraints common in practice cannot be enforced at the daily level in the Phase I integer program, the routes chosen may either be infeasible when creating the detailed schedule for the jth day or not as efficient as other potential schedules. Also, in constructing the delivery schedule we may be able to deliver more to the customers than the quantity specified by the Phase I integer program. Because this creates fuller truckload routes (and thus more revenue to the vendor) and does not lead to increased long-term costs, we want to take advantage of these opportunities where possible. To be more precise, we construct vehicle routes and schedules for j consecutive days, where *we force the total volume delivered to a customer over the j days to be greater than or equal to the total delivery volume specified by the solution to the integer program for these days, but we do not enforce specific delivery quantities to occur on specific days*. In this way, we should stay close to the delivery quantities suggested by the Phase I integer program, which are good from a long-term perspective, but we introduce some flexibility in the daily routing and scheduling, which is good from a short-term perspective.

6.1. Insertion Heuristics

We have developed and implemented an insertion heuristic for the j-day routing and scheduling problem that takes advantage of the inherent flexibility of the problem. Insertion heuristics are construction algorithms that build a feasible delivery schedule by inserting one as of yet unrouted customer into one of the current (partial) routes at every iteration. Insertion heuristics have proven their value in many routing and scheduling contexts; insertion heuristics are fast, produce decent solutions, are easy to implement, and can be extended easily to handle various practical complexities. For an elaborate treatment of insertion heuristics for routing and scheduling problems, including a discussion on how to handle variable demands, see Campbell and Savelsbergh (2004b). To keep the complexity of the algorithm a low-order polynomial, it is important to maintain values for the earliest and latest start times for each inserted delivery, as well as the minimum and maximum delivery quantities. In this section, we focus on those aspects of the insertion heuristic developed for the j-day problem that cannot be found in the above cited paper.

The selection of the insertion to be performed next is based on a score assigned to each insertion. A lower score indicates a more desirable insertion. The score is comprised of several components. The weighting of the different components can be adjusted to encourage a certain type of preferred delivery schedules.

• The first, and most commonly used, component is the increase in travel time resulting from the insertion. As the costs associated with a delivery route are typically assumed to be proportional to its length and duration, it is natural to include this "extra-mileage" cost. If we are considering adding a new route containing customer i, this is simply

$$2 \cdot tt_{i,\,depot}, \tag{15}$$

where $tt_{a,\,b}$ represents the travel time between any two locations a and b. Otherwise, if we consider inserting i between $p(i)$ and $s(i)$,

$$tt_{p(i),\,i} + tt_{i,\,s(i)} - tt_{p(i),\,s(i)}. \tag{16}$$

• The second component is the change in estimated waiting time resulting from the insertion. The waiting time on a route is estimated by starting the route at its latest possible start time, delivering the maximum quantity possible at each customer, and proceeding through the schedule, waiting only when necessary, until the end of the route. By computing the estimated waiting time on the route with and without the customer to be inserted, we can determine the change in estimated waiting time. Note that there may be some waiting on a route that, given the delivery windows, is unavoidable, so-called *forced waiting time*. When customers are near each other in terms of distance but far apart in terms of delivery windows, they do not make a good combination. A large increase in estimated waiting time is an indication of this situation.

• The third component is a charge for making routes "inflexible." In the final j-day delivery schedule, we want to have full-truckload or nearly full-truckload routes. Therefore, we want to discourage the construction of routes with a small difference between the earliest and latest possible start time and a large difference between the truck capacity and the maximum volume deliverable given the current set of customers, because it is unlikely that such routes can be extended to nearly full-truckload routes. An inflexibility charge is incurred only if an insertion results in a route with a difference between earliest and latest starting time that is less than x minutes and with a maximum volume deliverable that is less than y% of truck capacity. The charge is inversely dependent on y.

• The fourth component is a charge for inserting a customer on a route that is already capable of delivering a full truckload of product. This charge will serve to encourage larger deliveries, which postpones return visits to customers. A customer will still be inserted if resources are tight or there are no other satisfactory insertion points. Often there are many feasible solutions with similar distance costs. We want to encourage the selection of routes that make the best use of the available vehicle capacity.

At each iteration, we check the feasibility of inserting each delivery on all existing routes for the j days and of creating a (new) route containing only this delivery. We then evaluate the scores associated with every feasible insertion. For each delivery, we retain the feasible insertions with the lowest and second lowest score. Note that because we always consider creating a (new) route containing a delivery by itself, there exists at least one feasible insertion if there are sufficient resources.

If we start by inserting the deliveries with the lowest score, it may happen that nonurgent deliveries are inserted first and make it impossible to insert deliveries to customers that are running out of product soon. To prevent this from happening, we distinguish between high- and low-priority deliveries and always insert high-priority deliveries before low-priority deliveries. Deliveries that must take place during the j-day planning horizon (because, otherwise a customer would experience a stockout) are considered high-priority deliveries; the remaining deliveries are considered low-priority deliveries. In this way, if we do not have the resources to carry out all the deliveries suggested by the solution to the integer program solved in Phase I, only low-priority deliveries will be postponed.

We use the following selection rule (first for high-priority, then low-priority deliveries).

(1) If there are deliveries that cannot be inserted into any existing route, then among those deliveries select the one with the most expensive route for itself.

(2) If all deliveries can be inserted into at least one existing route, then select the one with the largest difference between the lowest and second lowest score.

The first part of the rule captures the idea that if there are deliveries that cannot be inserted into the current set of routes, we know that we have to create at least one more route, so we might as well do it now. Consequently, we are likely to always have several partial routes at one time. This creates more choices at each iteration, hopefully leading to better routing decisions. Realize that "most expensive route for itself" means that the cheapest route containing that customer alone is the most expensive over all of the other uninserted customers that cannot be inserted into an existing route.

The second part of the rule captures the idea of trying to insert a delivery well, and before all of its good alternatives become infeasible. If the second cheapest insertion point is very expensive or nonexistent, we need to insert this customer soon. This type of selection is sometimes referred to as "minimizing maximum regret."

The insertion heuristic is converted into a greedy randomized adaptive search procedure (GRASP) in a way similar to what was proposed by Kontoravdis and Bard (1995). A GRASP combines a greedy heuristic with randomization. Whenever the heuristic selects the next delivery to be inserted, it will pick randomly from among the q best choices, where q is prespecified. This allows the algorithm to make choices that do not seem to be the best at the time, but may provide better opportunities later. Within the GRASP framework, the insertion heuristic is executed many times and the best set of delivery routes constructed is ultimately selected. (Note that it is also possible to generate multiple sets of delivery routes by adjusting the weights on the different components of the score.)

Once a set of delivery routes has been selected, there may still exist some flexibility in the individual routes in terms of their earliest and latest start times and the minimum and maximum delivery quantities. In the basic case described here, it is optimal to deliver the maximum quantity possible at the latest start time. In Campbell and Savelsbergh (2004a), we propose an algorithm for selecting the times and quantities to maximize total delivery volume in a variety of situations. For example, if we include a variable stop time at each customer based on the size of the delivery, as we discussed in extensions for Phase I, how to maximize delivery quantity on a route is much less obvious. An example of the impact of addressing this issue optimally is included in the computational results.

6.2. Extending the Methodology

6.2.1. Multiple Deliveries. Handling customers with more than one delivery over the planning period is one of the more difficult extensions. For the customers who do not require more than one delivery per day, the integer program will choose no more than j deliveries in the first j days. If the number of deliveries in the integer program for a customer i over the j days is nod_i, then we will generate nod_i ordered deliveries for our insertion algorithm. The algorithm is modified to insert the multiple deliveries for a customer in the order dictated by the integer program; i.e., if the integer program picks a delivery of 500 for Day 1 and 100 for Day 3, the 500 will be inserted first, at a time before the customer runs out of product.

6.2.2. Shifts. Another extension that requires fairly extensive modification to the algorithm is the idea of grouping routes into shifts of lengths that can be executed feasibly by drivers. The Department of Transportation places limits on how much time drivers can work, including a 16-hour limit on how long a driver can be on duty (shift time limit), a 10-hour limit on the amount of time spent driving (drive time limit), and a limit of 70 total hours on duty in any 8 consecutive days. Therefore, the tendency in practice is to stay well below the 16-hour limit on a daily basis to keep more drivers available for duty. Because of the flexibility offered by insertion heuristics, we can create shifts for drivers literally by inserting deliveries into routes and routes into shifts.

6.2.3. Fixed-Start-Time Drivers. Fixed-start-time drivers are drivers who work on specified days of the week, and on those days, they go on duty at a certain time. Most plants that use fixed-start-time drivers have variable-start-time drivers as well (ones who start work at any time of the day).

When creating shifts, the algorithm can assign shifts to a driver based on the time windows of the first customer inserted on the shift. If the first customer inserted on a shift forces the shift to start between 9 am and 11 am, for example, then this shift will be assigned to the driver whose availability starts closest to 9 am. If no fixed-start-time drivers are available to make this delivery, then we assign this shift to a variable-start-time driver.

7. Computational Experiments

For all our computational experiments we will be using two datasets representing real-life problem instances (although relatively small ones). Because of confidentiality agreements between Praxair and its customers, there will be little specific information given about the customers, and all results will be scaled.

7.1. Datasets

Before discussing our computational experiments, we will examine some basic characteristics of the two datasets. The datasets represent two different plants. Plant 1 serves approximately 100 customers, and Plant 2 serves approximately 50 customers. In Figures 1 and 2, we present, for each plant, the geographical distribution of its customers. In each of the graphs, the customers are represented by small diamonds and the plant by a large square. We see that Plant 1 is south of almost all of its customers and that more customers are west of the plant than east of it. Furthermore, there are many customers located near each other on the west side. The customer usage rates, tank sizes, and vehicle capacities are such that there should be predominantly multistop routes. The customers served by Plant 2 are distributed quite differently, and the vehicle capacities, customer usage rates, and tank sizes are such that there could be many single-stop routes.

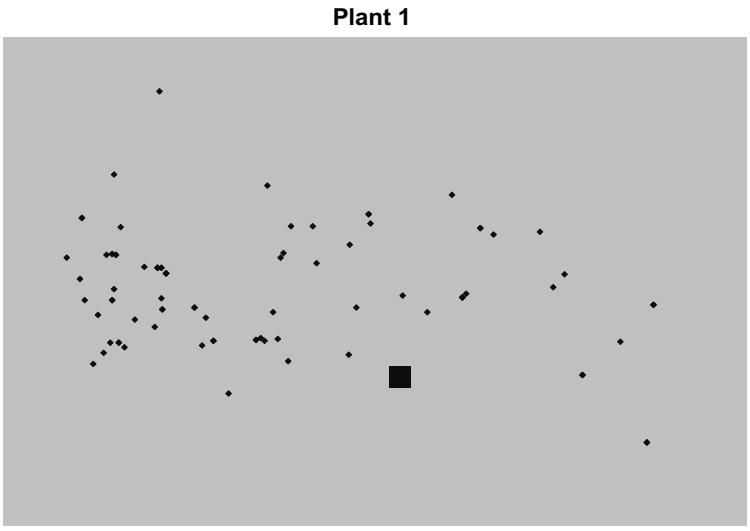

Figure 1 Geographical Distribution of Customers (Plant 1)

7.2. Solving the IP

In §5, we discussed several efforts to reduce the size of the Phase I integer program. Our first set of experiments demonstrates the effectiveness of these reductions. To demonstrate the impact of each of the various reductions, we will solve the Phase I integer program for Plant 1 once with all reductions in place, and subsequently with each of the reductions deactivated. All integer programs were solved on a Pentium II 366 MHz processor using XPRESS release 11 with default settings and a 10-minute time limit. The results are presented in Table 1, where the columns represent

- the case we are studying (*Case*),
- the number of rows in the IP (*Rows*),
- the number of columns in the IP (*Columns*),
- the total number of nonzero variables (*Nonzeros*),
- the time at which the first integral solution is found (*Time First IP*),
- the number of integral solutions found (# *IP Sols*), and
- the minimum of the solution time (proof of optimality) and the time limit (*Solution Time*).

We see that without aggregation we are unable to find a single feasible solution in 10 minutes. Furthermore, we see that clustering significantly reduces the size of the problem. The reduction in size due to the removal of nondriving, nonimpending, and nonbalance customers is only about 10%, but the solution process completed about six times faster. Finally, using specialized branching yields a speedup of a factor of almost two.

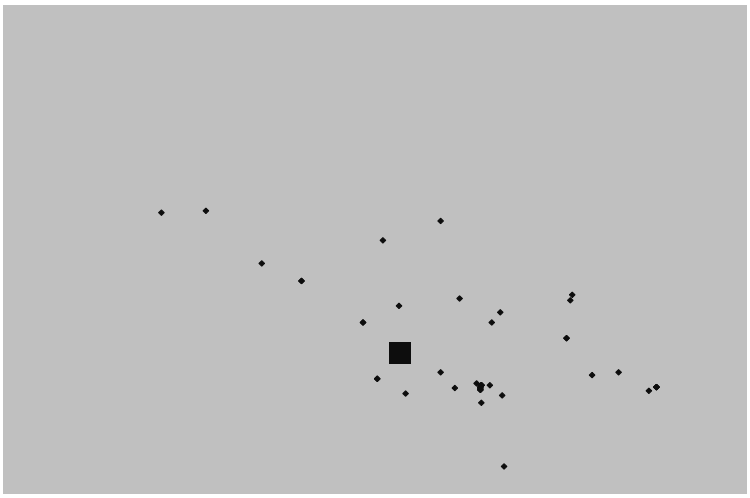

Figure 2 Geographical Distribution of Customers (Plant 2)

Table 1 Experiments with Various Modifications to the IP

Case	Rows	Columns	Nonzeros	Time first IP	# IP sols	Solution time
All modifications	4,280	6,457	84,282	76	1	76
No special branching	4,280	6,457	84,282	76	2	127
No customer set reduction	4,710	7,318	95,553	111	6	465
No clusters	10,718	17,424	226,891	285	3	600
No aggregation	30,860	27,690	991,032	n/a	0	600

7.3. Considering the Future

The planning horizon used in the Phase I integer program should be based on the time period for which we believe customer usage data to be reliable. Furthermore, as indicated above, the integer program has variables representing daily decisions for the first j days of the planning horizon and variables representing weekly decisions for the remaining $k' = k/7$ weeks of the planning horizon. Even if the customer usage data has historically been reliable for a period of about four weeks, for example, it is not clear that there is no value in considering six weeks. Similarly, it is not clear that considering only one week beyond the k days would lead to an inferior solution. In fact, if the solution is only slightly worse, but much faster to obtain, it might even be preferred to consider a smaller planning horizon. It is also not obvious how many days should be represented at a daily level, i.e., how to choose the value of j. With all of this in mind, we performed a variety of tests, experimenting with different values of j and different number(s) of weeks beyond j to see how the solutions for our two datasets are impacted.

In our computational experiments, we simulate the use of a rolling-horizon approach covering a month. In each iteration of the rolling-horizon framework, we solve a Phase I integer program, we run the Phase II routing and scheduling heuristic with the information from the solution to the integer program for the first two days, we implement the resulting routes, and we move the clock forward two days in time.

Before presenting and discussing this set of computational results, we elaborate on the difficulty of comparing solutions. A solution to the inventory-routing problem for a given planning period specifies which vehicles are visiting which customers on each day of the planning period, in what order the deliveries are being made, and how much is delivered to each customer. However, even with all this information it is still nontrivial to evaluate the quality of the solution. This is an issue faced by everyone who uses a vendor-managed inventory policy in practice. For example, if we consider a planning period of four weeks, as we will do in our computational experiments, it is not obvious how to compare two solutions and claim that one is better than the other because the IRP is really an infinite-horizon problem. If the total distance traveled in one solution is less than in the other solution, this represents lower distribution costs. However, if in the solution with a higher total distance traveled only full-truckload deliveries are made, how can we say this solution is worse? It utilizes the trucks extremely well and may end in a state that is a much better starting point for the deliveries that have to be made in the following weeks.

Therefore, we look at several statistics to evaluate the quality of a solution for a planning period. The statistics we will consider, as we mentioned earlier, will be scaled to conceal the true values, except for average hours on shift. Even though the results are scaled, we can still use these values to make comparisons among different solutions. The values presented include the following.

- Volume (*Volume*): total volume delivered over the planning horizon.
- Mileage (*Mileage*): total miles driven over the planning horizon.
- Volume per mile (*V/M*): ratio of total volume and total mileage. Care has to be taken when using and interpreting this performance measure. It may not be a good measure to compare performance of different plants as the customer characteristics—such as location, usage rate, and tank capacity—can have a big impact on volume per mile. It is, however, a good measure to compare different planning methods for a single plant. If one method yields a higher volume per mile than another method for the same area and over the same time horizon, it indicates that the first method delivers more product with the same amount of resource usage.
- Truck utilization (*Avg. Util.*): the average percent of truck volume delivered on a route.
- Shift length (*Avg. Shift*): the average driver shift length.

To represent the different variants we considered, we use the following notation. A variant is represented by a string consisting of the number of days + "*d*" + number of weeks considered aggregately + "*w*."

The results of the experiments for Plant 1 are presented in Table 2 and for Plant 2 in Table 3.

These statistics show that it is not straightforward to pick the best day/horizon because different statistics suggest different settings. Only after experimentation and familiarity with a dataset will the best settings emerge. Computational requirements as well as

Table 2 Experiments with Different Numbers of Days and Weeks (Plant 1)

Case	Volume	Avg. util.	Mileage	*V/M*	Avg. shift
2*d*0*w*	18,703.94	98.83	16,026.95	1.16	8.98
2*d*4*w*	20,710.66	95.47	18,207.41	1.13	8.69
3*d*0*w*	42,544.48	98.89	37,336.66	1.14	8.54
3*d*1*w*	20,482.35	98.82	17,502.81	1.17	9.07
3*d*4*w*	19,545.74	96.62	17,167.25	1.15	8.88
3*d*6*w*	43,284.98	97.34	38,890.91	1.11	8.74
7*d*4*w*	42,136.38	97.99	37,390.66	1.12	8.64

solution quality will be the final determinants. For the remaining experiments, we will use setting $3d1w$.

7.4. Algorithm Comparison

Next, we compare our approach to a greedy algorithm that captures many of the ideas presented in the literature. Most existing algorithms do not consider many of the complications that we do, so we can only compare the simplest form of our approach. The greedy algorithm assigns customers to days based on when customers will run out of product and the resource availability. More specifically, the algorithm selects customers that will be close to run out by the time a truck can arrive on a given day and sets the delivery amount for those customers equal to available capacity, i.e., tank capacity minus inventory. The remaining vehicle capacity is then distributed among customers who will run out the next day. The deliveries are sequenced (as with the IP) using an insertion-based heuristic. The deliveries, however, are restricted to being routed on the day for which the delivery was created and are restricted to the specified size (as is the case in most existing algorithms for the IRP). The results for Plant 1 are presented in Table 4 and for Plant 2 in Table 5.

We see for Plant 1 that even with a larger total volume delivered, the greedy method has a substantially lower volume per mile, lower average utilization, and longer average shift length. Our IP-based approach is able to include deliveries to customers near the imminent customers, which consume the remaining truck capacity and improve the utilization numbers. For Plant 2, the volume per mile is identical, but again

Table 3 Experiments with Different Numbers of Days and Weeks (Plant 2)

Case	Volume	Avg. util.	Mileage	*V/M*	Avg. shift
2*d*0*w*	24,761.70	99.14	7,027.08	3.60	8.22
2*d*4*w*	30,937.16	98.14	8,705.28	3.61	7.98
3*d*0*w*	30,755.17	97.92	8,344.15	3.82	7.66
3*d*1*w*	27,881.57	97.68	7,573.03	3.80	8.98
3*d*4*w*	31,864.67	98.17	8,842.11	3.76	8.73
3*d*6*w*	31,324.59	96.43	8,966.76	3.70	8.66
7*d*4*w*	31,443.81	97.80	9,033.81	3.69	7.53

Table 4 IP vs. Greedy—Results for One Month (Plant 1)

Case	Volume	Avg. util.	Mileage	*V/M*	Avg. shift
IP	20,482.35	98.82	17,502.81	1.17	9.07
Greedy	41,449.50	94.79	37,119.57	1.11	9.70

we see lower average utilization and longer average shift lengths.

These results indicate that IP-based approaches, even in their simplest form, may provide a valuable component of an overall control strategy for vendor-managed inventory resupply.

7.5. Flexibility

The strength of our hierarchical approach is a Phase I that captures the long-term aspects of the problem as well as several of the practical complexities but is still computationally tractable and a Phase II that capitalizes on the inherent flexibility in a VMR relationship. To evaluate the importance of capitalizing on the inherent flexibility in a VMR relationship, we have taken the greedy algorithm used in the previous section and extended it in various ways to exploit this flexibility. More specifically, we created orders according to the scheme outlined above, but we allowed the routing and scheduling heuristic to change the day at which the delivery occurs (as long as this is feasible), to change the delivery quantity slightly above and below the specified quantity, and to add balance customers (as long as this is profitable).

We see that even though the volume per mile changes only slightly in Tables 6 and 7, the average utilization goes up and the average shift length goes down, both desirable characteristics in practice.

7.6. Using GRASP

In §5, we discussed randomization as a powerful tool to improve the performance of insertion heuristics. We employ randomization at several points in our routing and scheduling heuristic. Our next set of computational experiments demonstrates the impact of randomization on the solution quality.

As a first experiment, we run the routing and scheduling heuristic multiple times with the same input data and look at the variance in solution quality. We report the resulting total mileage statistic for different numbers of repetitions in Figures 3 and 4. In these graphs, the x-axis represents the solution quality relative to a lower bound. We can see

Table 5 IP vs. Greedy—Results for One Month (Plant 2)

Case	Volume	Avg. util.	Mileage	*V/M*	Avg. shift
IP	27,881.57	97.68	7,573.03	3.80	8.98
Greedy	28,812.72	91.80	7,658.64	3.80	11.58

Table 6 Added Flexibility (Plant 1)

Case	Volume	Avg. util.	Mileage	*V*/*M*	Avg. shift
Greedy	41,449.50	94.79	37,119.57	1.11	9.70
With flexibility	41,345.51	95.10	37,113.03	1.11	8.43

Table 7 Added Flexibility (Plant 2)

Case	Volume	Avg. util.	Mileage	*V*/*M*	Avg. shift
Greedy	28,812.72	91.80	7,658.64	3.80	11.58
With flexibility	29,138.81	94.25	7,594.72	3.85	7.22

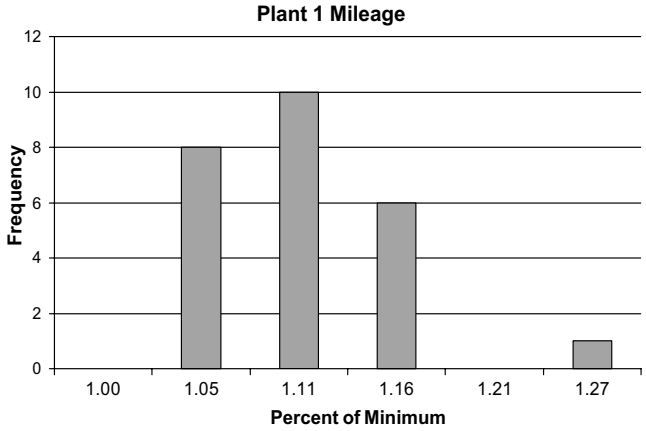

Figure 3 Frequency with Varying Total Mileage (Plant 1)

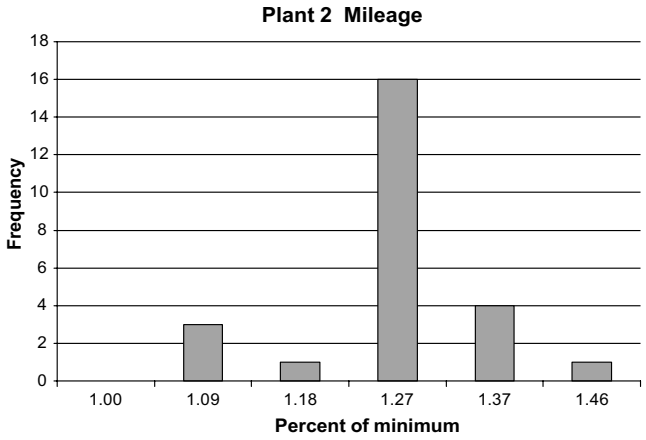

Figure 4 Frequency with Varying Total Mileage (Plant 2)

Table 8 Experiments with Different Numbers of Runs over Long Term (Plant 1)

Case	Volume	Avg. util.	Mileage	*V*/*M*	Avg. shift
2 runs	41,322.75	93.67	37,814.62	1.09	8.33
5 runs	41,297.26	93.86	37,575.31	1.10	8.42
25 runs	41,345.51	95.10	37,113.03	1.11	8.43

Table 9 Experiments with Different Numbers of Runs over Long Term (Plant 2)

Case	Volume	Avg. util.	Mileage	*V*/*M*	Avg. shift
2 runs	29,106.52	90.58	7,931.25	3.72	6.50
5 runs	29,133.55	92.61	7,747.53	3.79	7.23
25 runs	29,138.81	94.25	7,594.72	3.85	7.22

that 25 runs creates a wide variety of distance values, indicating that GRASP may help us find better solutions.

In itself, this information is not sufficient to draw any conclusions about the impact over an entire planning horizon. Therefore, in our next set of experiments, we investigate the impact of always choosing the solution with the highest volume per mile for each two-day period within a month. For this test, we used the enhanced greedy heuristic discussed above (the one that allows for more flexibility during routing and scheduling). The results for Plant 1 are presented in Table 8 and for Plant 2 in Table 9.

Even though the improvements are not large, for both Plants 1 and 2, we observe a steady improvement of the statistics as the number of iterations increases for the routing and scheduling heuristic.

8. Summary

We have presented the inventory-routing problem and an optimization-based approach for its solution. Computational experiments indicate the potential value of optimization-based approaches for complex routing and scheduling problems. Problems of this magnitude and complexity require careful decomposition, and many future research opportunities exist in this area.

8.1. Future Research

A customer may be located equidistant from two plants, so it may be more efficient to deliver to this customer from either one of the plants at different times, depending on the demand of other nearby customers. Examining how to best deliver to a set of customers considering a number of plants is known as the multidepot inventory-routing problem. The research community has not addressed this problem, even though this is a real concern for many companies practicing vendor-managed inventory. The problem of making all of the delivery and routing decisions for one plant is hard enough, but removing the customer-plant assignments for a company with a large number of plants makes the problem literally multiply in size. When the customer-plant assignments are "freed," however, plant inventory

management becomes an issue, because we have to make sure we are not planning too many trips from any of the plants. The benefits from automated multidepot routing are obvious and could result in a substantial improvement in distribution costs for many industries. We would like to extend our decomposition approach to handle this even larger and more complex problem.

Acknowledgments

The authors would like to thank Laurie Dutton of Praxair for her involvement in and support of this research effort.

References

Ball, M. 1988. Allocation/routing: Models and algorithms. B. Golden, A. Assad, eds. *Vehicle Routing: Methods and Studies*. Elsevier Science, Amsterdam, The Netherlands.

Bard, J., L. Huang, M. Dror, P. Jaillet. 1998a. A branch and cut algorithm for the VRP with satellite facilities. *IIE Trans. Oper. Engrg.* **30** 821–834.

Bard, J., L. Huang, P. Jaillet, M. Dror. 1998b. A decomposition approach to the inventory routing problem with satellite facilities. *Transportation Sci.* **32** 189–203.

Beltrami, F., L. Bodin. 1974. Networks and vehicle routing for municipal waste collection. *Networks* **4** 65–94.

Brumback, N. 1995. Rubbermaid's new system scores. *Home Furnishing Network* **69** 11.

Campbell, A., M. Savelsbergh. 2004a. Delivery volume optimization. *Transportation Sci.* **38**(2) 210–223.

Campbell, A., M. Savelsbergh. 2004b. Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation Sci.* **38**(3) 369–378.

Campbell, A., L. Clarke, A. Kleywegt, M. Savelsbergh. 1998. Inventory routing. T. Crainic, G. Laporte, eds. *Fleet Management and Logistics*. Kluwer Academic Publishers, Boston, MA.

Christofides, N., J. Beasley. 1984. The period routing problem. *Networks* **14** 237–256.

Dror, M., M. Ball. 1987. Inventory/routing: Reduction from an annual to a short period problem. *Naval Res. Logist. Quart.* **34** 891–905.

Dror, M., M. Ball, B. Golden. 1985. Computational comparison of algorithms for the inventory routing problem. *Ann. Oper. Res.* **4** 3–23.

Federgruen, A., P. Zipkin. 1984. A combined vehicle routing and inventory allocation problem. *Oper. Res.* **32** 1019–1036.

Gallego, G., D. Simchi-Levi. 1990. On the effectiveness of direct shipping strategy for the one-warehouse multi-retailer *R*-systems. *Management Sci.* **36** 240–243.

Gaudioso, M., G. Paletta. 1992. A heuristic for the periodic vehicle routing problem. *Transportation Sci.* **26** 86–92.

Golden, B., A. Assad, R. Dahl. 1984. Analysis of a large scale vehicle routing problem with an inventory component. *Large Scale Systems* **7** 181–190.

Jaillet, P., L. Huang, J. Bard, M. Dror. 2002. Delivery cost approximations for inventory routing problems in a rolling horizon framework. *Transportation Sci.* **36** 292–300.

Kleywegt, A., V. Nori, M. Savelsbergh. 2002a. Dynamic programming approximations for a stochastic inventory routing problem. Technical Report TLI-02-06, Department of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Kleywegt, A., V. Nori, M. Savelsbergh. 2002b. The stochastic inventory routing problem with direct deliveries. *Transportation Sci.* **36** 94–118.

Kontoravdis, G., J. Bard. 1995. A GRASP for the vehicle routing problem with time windows. *ORSA J. Comput.* **7** 10–23.

Mongelluzzo, B. 1998. Shippers let vendors manage the stock: Wal-Mart's suppliers share in databases. *J. Commerce* **417** 12A.

Nannery, M. 1994. AAMA promotes vendor-managed inventory; manufacturers say they can replenish better than stores. *Daily News Record* **24** 8.

Nori, V. 1999. Algorithms for dynamic and stochastic logistics problems. Ph.D. dissertation, Department of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Purpura, L. 1997. Vendor-run inventory: Are its benefits exaggerated? *Supermarket News* **47** 59–60.

Radice, C. 1999. Two heads are better than one. *Chain Store Executive with Shopping Center Age* **74** 60–61.

Ross, J. 1998. HEB project leads expansion of vendor managed inventory programs. *Stores* **80** 46–47.