

An Integrated Inventory-Routing System for Multi-item Joint Replenishment with Limited Vehicle Capacity

SOMBAT SINDHUCHARO^{1,†}, H. EDWIN ROMEIJN^{2,‡}, ELIF AKÇALI² and REIN BOONDISKULCHOK¹

¹*Department of Industrial Engineering Chulalongkorn University, Bangkok, Thailand 10330*

²*Department of Industrial and Systems Engineering University of Florida, Gainesville, FL 32611-6595, USA (e-mail: S7568@yahoo.com)*

(Received 12 December 2003; accepted in revised form 15 April 2004)

Abstract. In this paper, we develop a mathematical programming approach for coordinating inventory and transportation decisions in an inbound commodity collection system. In particular, we consider a system that consists of a set of geographically dispersed suppliers that manufacture one or more non-identical items, and a central warehouse that stocks these items. The warehouse faces a constant and deterministic demand for the items from outside retailers. The items are collected by a fleet of vehicles that are dispatched from the central warehouse. The vehicles are capacitated, and must also satisfy a frequency constraint. Adopting a policy in which each vehicle always collects the same set of items, we formulate the inventory-routing problem of minimizing the long-run average inventory and transportation costs as a set partitioning problem. We employ a column generation approach to determine a lower bound on the total costs, and develop a branch-and-price algorithm that finds the optimal assignment of items to vehicles. We also propose greedy constructive heuristics, and develop a very large-scale neighborhood (VLSN) search algorithm to find near-optimal solutions for the problem. Computational tests are performed on a set of randomly generated problem instances.

Key words: Inventory-routing, Multi-item inventory replenishment, Transportation, Economic order quantity, Column generation, Very large-scale neighborhood search

1. Introduction

Inventory control and transportation planning have traditionally been managed by different departments in an organization, each of which has its own set of goals. Consequently, inventory and transportation costs are typically minimized separately by each department. In general, however, there is a trade-off between the inventory and transportation costs in a logistics system, since decreasing one of these cost categories generally leads to an increase in the other one. Generally, the total inventory and

[†] The work of this author was supported by a scholarship of the Faculty of Engineering of Ubonratchathani University, Ubonratchathani, Thailand.

[‡] The work of this author was supported in part by the National Science Foundation under Grant No. DMI-0085682.

transportation costs in the system can be greatly reduced if inventory control and transportation planning are closely coordinated.

In a multi-item inventory system, it is practical to combine groups of items in a single replenishment order to accomplish substantial cost savings (see, e.g., [39]) due to the sharing of fixed replenishment costs. When these replenishment costs contain a transportation cost component, this cost sharing is often a consequence of the ability to share truck and loading equipment between the items. In addition, the design of a vehicle route for visiting a group of retailers (in the case of a distribution system) or a group of suppliers (in the case of a collection system) may have a significant effect on the magnitude of the replenishment costs. Hence, it is desirable to design an efficient joint replenishment strategy that coordinates inventory control and transportation planning. The interaction between transportation and inventory costs has been discussed extensively in the literature in the context of just-in-time (JIT) (see [46]), vendor managed inventory (VMI) (see [17, 43]), and supply chain coordination (see [13]).

In this paper, we consider an inbound commodity collection system that consists of a central warehouse and a set of geographically dispersed suppliers. Each supplier produces one or more non-identical items, each of which faces demand from outside retailers. The warehouse replenishes the retailers who, in turn, meet outside demand for the items. We assume that these retailers operate according to JIT principles in order to reduce inventory levels. As a result, the demand rate faced by the warehouse for each of its items is constant.

The warehouse uses a fleet of vehicles to collect the items from its suppliers. These vehicles have limited capacity, and they are also subject to frequency constraints that limit the number of trips that each truck can make per time unit. The frequency constraint may, for example, be due to the time required for vehicle maintenance or by the fact that material handling capacity is limited. The costs of the integrated inventory-routing system include inventory holding costs at the central warehouse, fixed ordering and vehicle dispatching costs, and vehicle routing costs. It is assumed that there is no shortage or delay at any supplier. Our objective is to develop mathematical programming models and algorithms to coordinate inventory and transportation decisions faced by the warehouse in the system described above.

It seems unlikely that it is possible to identify an optimal strategy for our problem. But more importantly, even if such an optimal strategy could be found efficiently, it may be too complex to be implementable in practice. Nevertheless, some progress has been made in this direction recently with the work of Adelman [1], who develops an approximate dynamic programming approach that finds high quality policies without imposing any a priori policy structure for inventory routing problems where only the routing

costs are taken into account. Due to the difficulty, as well as perhaps the undesirability from an implementability point of view, of finding truly optimal policies, it is common practice in inventory-routing problems to consider a given policy structure up front, and focus on finding optimal parameters for that policy.

We adopt a policy where the set of items is partitioned into disjoint groups and each group of items is assigned to a vehicle. The vehicle leaves the warehouse, visits the set of suppliers corresponding to the items in its group, and returns to the warehouse, where the items are unloaded and stored. We assume that no item can be assigned to more than one group, i.e. the orders cannot be split across multiple vehicles. However, it is not necessary for items produced by the same supplier to be in the same group, i.e. a supplier can be visited by multiple vehicles. Finally, the fact that the warehouse faces a constant demand for each item leads to a joint replenishment of all items in a group using an economic order quantity (EOQ) policy. In Figure 1, we illustrate an example in which there are five suppliers that the warehouse purchases items from and three retailers that are served by the warehouse. The warehouse has two vehicles, one of which collects items from suppliers S_1 , S_2 , and S_3 while the other one collects items from suppliers S_3 , S_4 , and S_5 . The warehouse faces demand for the items at the retailers R_1 , R_2 , and R_3 .

Our policy closely resembles to the class of Fixed Partition policies introduced by Bramel and Simchi-Levi [14] for an inventory-routing problem in which a single item is distributed among retailers. Although such policies are generally not optimal, they are important from a practical standpoint, as they are easy to implement. In particular, they allow for efficient inte-

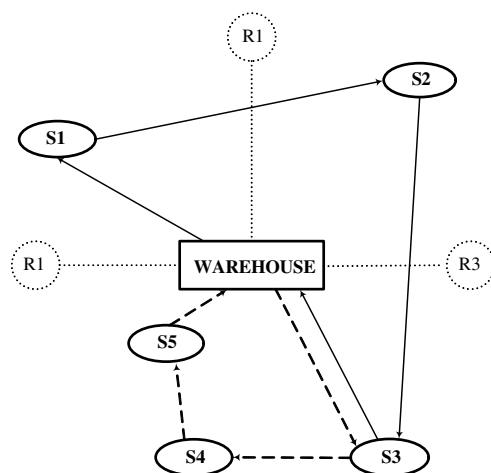


Figure 1. Inventory-routing system with multiple suppliers and a centralized warehouse.

gration of several business functions. Chan et al. [18] and Chan and Simchi-Levi [19] have shown that such policies can be highly effective, by deriving an asymptotic error bound on the obtained solution under different assumptions on the transportation cost structure.

Despite restricting ourselves to the policy described above, finding the optimal policy parameters is still a daunting task. In particular, our problem is to partition the items into groups, design collection routes for the vehicles, and specify the replenishment quantities for the items simultaneously while minimizing the total vehicle dispatching, vehicle routing, fixed ordering, and inventory holding costs, while observing vehicle capacity and frequency constraints. We formulate this problem as a set partitioning problem, and develop a column generation approach for this set-partitioning formulation, which can be used to determine lower bounds on the optimal costs. This lower bound can be used to evaluate the performance of heuristic approaches to the problem. In addition, exact solutions for small problem instances can be obtained by embedding this column generation procedure in a branch-and-price algorithm. To obtain near-optimal solutions for larger problem instances, we propose several constructive heuristics. In order to improve the solutions found by these heuristics, we develop a very large-scale neighborhood (VLSN) search algorithm (see, e.g., [3, 7]) that efficiently finds a local optimum solution whose quality can be expected to be high due to the vast size of the neighborhood used.

In the remainder of this section, we will review the literature related to our problem. We note that the literature on the integrated solution of inventory and transportation problems is very rich. Our discussion will focus only on the single- and multiple-item models that are most closely related to our research. The first model that dealt with the integration of inventory and routing problems in a single model was developed by Federgruen and Zipkin [26]. They study the allocation of a single item from a central depot to many retailers using a fleet of capacitated vehicles. The problem is formulated as a non-linear integer program and interchange heuristics are modified to solve the problem. Other early references in single-item systems include Burns et al. [16], who analyze direct shipping and peddling strategies for an infinite-horizon one-warehouse multi-retailer distribution system, and Gallego and Simchi-Levi [29], who study a direct shipping strategy.

Anily and Federgruen [8] also consider a single-item distribution system with one depot and a set of retailers. They assume that the demand rate of each retailer is an integer multiple of some base demand rate, and study a specific class of replenishment strategies in which a set of (possibly overlapping) regions are defined, where each retailer may belong to several regions. Vehicles are assigned to regions, and when a vehicle visits a retailer in a particular region, it must visit all retailers in that region. In [8],

they analyze the case where the inventory is held only at the retailers. Later, they extend the work to consider the case where inventory can be held at the central warehouse as well (see [9]), and to more general inventory holding cost functions (see [10]). Other relevant papers that study inventory-routing problems with a single item are Dror et al. [23], Dror and Ball [24], Chien et al. [21], and Chaovalitwongse et al. [20].

The literature on multi-item inventory-routing systems is relatively scarce. Works by Viswanathan and Mathur [45] and Qu et al. [35] are the notable papers in the area. Viswanathan and Mathur [45] consider the integration of vehicle routing and inventory decisions for a single warehouse, multi-retailer, multi-product distribution system with deterministic demands. They propose a stationary nested joint replenishment policy heuristic for the problem where replenishment intervals are limited to be power of two multiples of a base planning period and are computed using a modification of the standard EOQ formula. Qu et al. [35] consider multi-item joint replenishment in an inbound material-collection system with a central warehouse using uncapacitated vehicles, where geographically dispersed suppliers face stochastic demands. They propose a modified periodic policy with an order-up-to level in which each replenishment period is an integer multiple of a base period. A heuristic solution method is proposed that decomposes the problem into inventory and vehicle routing models. For a review of multi-item inventory-routing systems, we refer the reader to Buffa and Munn [15], Ben-Khedher and Yano [12], and Bertazzi and Speranza [13].

The outline of this paper is as follows. In Section 2, we model our integrated inventory and transportation problem under the policy described above as a set partitioning problem. Based on this formulation, we develop a column generation and a branch-and-price algorithm in Section 3, paying particular attention to the pricing subproblem, which is a very challenging optimization problem in its own right. Greedy construction heuristics and the VLSN algorithms are given in Section 4. In Section 5, we report on computational results. Finally, concluding remarks and future research directions are provided in Section 6.

2. Model Formulation

We denote the set of items stored by the central warehouse by \mathcal{S} . Item j ($j \in \mathcal{S}$) faces a deterministic demand rate D_j . The items are collected from the suppliers using a fleet of m vehicles. The total system costs consist of the holding costs associated with each item, which are incurred at a constant rate of h_j per unit per year for item j ($j \in \mathcal{S}$), as well as fixed costs. These fixed costs include fixed ordering costs and fixed vehicle dispatching costs, as well as the total vehicle routing costs associated with a trip, which

is the cost of a Traveling Salesman Problem (TSP) where the cities are the warehouse and the suppliers of the items collected in the trip. For convenience, the fixed ordering and dispatching costs are combined in a single term K per vehicle per trip. Using a policy where each item is assigned to a single group that is replenished repeatedly using a given vehicle, our inventory-routing problem is then to determine the subsets of items that are replenished with a single vehicle, as well as the corresponding replenishment quantities, the replenishment interval and the optimal vehicle routes, that minimize the average total inventory and transportation cost per unit time.

We will first determine the average total inventory and transportation costs per unit time for a *given* set of items $S \subseteq \mathcal{S}$ assigned to a vehicle, and under the simplifying assumption that the vehicle is uncapacitated and does not face a frequency constraint. We assume that the fixed transportation cost associated with this set is of the form

$$L(S) = K + \text{TSP}(S)$$

where $\text{TSP}(S)$ denotes the cost of the optimal TSP route for visiting the suppliers corresponding to the items in S , and K represents any other fixed costs associated with using the vehicle. If we denote the time between replenishments of the items in S by $T(S)$, then the corresponding optimal replenishment quantities are given by $Q_j = D_j T(S)$ for all items $j \in S$. The total costs per unit time incurred for replenishing the items in S as a function of the replenishment interval $T(S)$ is equal to

$$\frac{L(S)}{T(S)} + \frac{1}{2} \sum_{j \in S} h_j D_j T(S).$$

It is convenient to define the aggregate demand and weighted average unit holding costs for subset S as follows:

$$D(S) = \sum_{j \in S} D_j$$

$$h(S) = \frac{\sum_{j \in S} h_j D_j}{D(S)}.$$

The cost function can then be rewritten as

$$\frac{L(S)}{T(S)} + \frac{1}{2} h(S) D(S) T(S)$$

which is a standard EOQ-type cost function and thus immediately yields that the optimal replenishment time for set S is equal to

$$T^*(S) = \sqrt{\frac{2L(S)}{h(S)D(S)}}.$$

Alternatively, we could formulate the cost function in terms of the aggregate replenishment quantity

$$Q(S) = \sum_{j \in S} Q_j.$$

Note that the individual item replenishment quantities have to satisfy

$$\frac{Q_j}{Q(S)} = \frac{D_j}{D(S)} \quad \text{for } j \in S$$

or

$$Q_j = \frac{D_j}{D(S)} Q(S) \quad \text{for } j \in S.$$

Since $Q_j = D_j T(S)$ for all $j \in S$, we clearly also have $Q(S) = D(S)T(S)$. The total costs per unit time incurred for replenishing the items in S can now equivalently be written as

$$L(S) \frac{D(S)}{Q(S)} + \frac{1}{2} h(S) Q(S).$$

This is again a standard EOQ-cost function, and leads to the optimal aggregate replenishment quantity for subset S

$$Q^*(S) = \sqrt{\frac{2D(S)L(S)}{h(S)}}.$$

Using either approach, we obtain that the optimal replenishment quantities for the individual items are equal to

$$Q_j^* = D_j T^*(S) = \frac{D_j}{D(S)} Q^*(S) = D_j \sqrt{\frac{2L(S)}{D(S)h(S)}} \quad \text{for } j \in S.$$

The corresponding optimal costs are equal to

$$c(S) = \sqrt{2D(S)L(S)h(S)}.$$

The integrated inventory-routing problem can now be formulated as a set partitioning problem:

$$\min \sum_{i=1}^m c(S^{(i)})$$

subject to

$$\bigcup_{i=1}^m S^{(i)} = \mathcal{S}$$

$$S^{(i)} \cap S^{(k)} = \emptyset \quad \text{for all } i, k = 1, \dots, m; i \neq k.$$

In the capacitated case where each vehicle has limited capacity C , a similar partitioning problem can be obtained but with

$$Q^*(S) = \min \left\{ \sqrt{\frac{2D(S)L(S)}{h(S)}}, C \right\}$$

and corresponding optimal costs equal to

$$c(S) = \begin{cases} \sqrt{2D(S)L(S)h(S)} & \text{if } \sqrt{\frac{2D(S)L(S)}{h(S)}} \leq C \\ L(S)\frac{D(S)}{C} + \frac{1}{2}h(S)C & \text{if } C \leq \sqrt{\frac{2D(S)L(S)}{h(S)}} \end{cases}$$

In addition, the vehicles may face a frequency constraint. This means that the number of trips per time unit is bounded from above or, equivalently, the replenishment interval is bounded from below. If F is the maximum number of trips allowed for each vehicle, then $1/F$ is the minimum length of the replenishment interval. Clearly, the set $S \subseteq \mathcal{S}$ is a *feasible* subset of items only if

$$D(S) \leq CF.$$

With both the vehicle capacity and frequency constraints included in the model, the aggregate replenishment quantity for the items in a feasible subset S can be determined from

$$Q^*(S) = \begin{cases} \frac{D(S)}{F} & \text{if } \sqrt{\frac{2D(S)L(S)}{h(S)}} \leq \frac{D(S)}{F} \\ \sqrt{\frac{2D(S)L(S)}{h(S)}} & \text{if } \frac{D(S)}{F} \leq \sqrt{\frac{2D(S)L(S)}{h(S)}} \leq C \\ C & \text{if } C \leq \sqrt{\frac{2D(S)L(S)}{h(S)}} \end{cases}$$

or

$$Q^*(S) = \max \left\{ \frac{D(S)}{F}, \min \left\{ \sqrt{\frac{2D(S)L(S)}{h(S)}}, C \right\} \right\}$$

and the corresponding optimal costs can be obtained from

$$c(S) = \begin{cases} L(S)F + \frac{1}{2}h(S)\frac{D(S)}{F} & \text{if } \sqrt{\frac{2D(S)L(S)}{h(S)}} \leq \frac{D(S)}{F} \\ \sqrt{2D(S)L(S)h(S)} & \text{if } \frac{D(S)}{F} \leq \sqrt{\frac{2D(S)L(S)}{h(S)}} \leq C \\ L(S)\frac{D(S)}{C} + \frac{1}{2}h(S)C & \text{if } C \leq \sqrt{\frac{2D(S)L(S)}{h(S)}}. \end{cases} \quad (1)$$

In this case, the set partitioning problem becomes

$$\min \sum_{i=1}^m c(S^{(i)})$$

subject to

$$S^{(i)} \subseteq \mathcal{S}$$

$$\bigcup_{i=1}^m S^{(i)} = \mathcal{S}$$

$$D(S^{(i)}) \leq CF \quad \text{for all } i = 1, \dots, m$$

$$S^{(i)} \cap S^{(k)} = \emptyset \quad \text{for all } i, k = 1, \dots, m; i \neq k.$$

3. An Exact Solution Approach Using Branch-and-Price

In this section, we will develop a branch-and-price algorithm that can be used to solve our inventory-routing problem to optimality. This algorithm is based on a column generation approach to the set partitioning formulation of the problem. After formulating this problem as an integer programming problem, we solve its linear programming (LP) relaxation via column generation. In this approach, we iteratively solve the problem with only a limited number of candidate subsets for the vehicles. In each iteration, we solve a subproblem, called the pricing problem, that either verifies that the current solution is optimal for the entire problem, or identifies one or more subsets that should be added to the limited model. We incorporate this solution method for solving the LP-relaxation of the set partitioning problem in a branch-and-bound algorithm if the optimal solution of the LP-relaxation is fractional. Applications of this methodology have been applied to other set partitioning problems, such as the generalized assignment problem (see [37]), the multi-period single-sourcing problem (see [28]), a continuous-time version of that model (see [31]), a joint location-inventory model (see [38]), and the crew scheduling problem (see [11]).

3.1. A COLUMN GENERATION APPROACH TO THE SET PARTITIONING FORMULATION

We will first formulate the set partitioning problem as an integer programming problem. Without loss of generality, we will assume that there are n items, and $\mathcal{S} = \{1, \dots, n\}$. Then, let N_i denote the number of feasible candidate subsets of items that can be assigned to vehicle i . Each of these subsets is represented by a binary vector

$$\alpha_i^\ell = (\alpha_{i1}^\ell, \dots, \alpha_{in}^\ell)^\top,$$

where $\alpha_{ij}^\ell = 1$ if item j is in candidate subset ℓ for vehicle i , and $\alpha_{ij}^\ell = 0$ otherwise. Letting $c_i(\cdot)$ denote the cost function for vehicle i (as derived in Section 2 for a generic vehicle), we obtain that the cost of subset ℓ of vehicle i is $c_i(\alpha_i^\ell)$. Note that we use a binary incidence vector of a subset of \mathcal{S} rather than the subset itself as the argument of c_i . When it is convenient, we will also do so for all set functions introduced in Section 2. Finally, we introduce a binary variable y_i^ℓ that takes on the value 1 if we choose subset ℓ for vehicle i , and 0 otherwise. The set partitioning problem can then be reformulated as

$$\min \sum_{i=1}^m \sum_{\ell=1}^{N_i} c_i(\alpha_i^\ell) y_i^\ell$$

subject to (P)

$$\sum_{i=1}^m \sum_{\ell=1}^{N_i} \alpha_{ij}^{\ell} y_i^{\ell} = 1 \quad j = 1, \dots, n \quad (2)$$

$$\sum_{\ell=1}^{N_i} y_i^{\ell} = 1 \quad i = 1, \dots, m \quad (3)$$

$$y_i^{\ell} \in \{0, 1\} \quad \ell = 1, \dots, N_i, \quad i = 1, \dots, m.$$

The first n constraints ensure that each item is collected by exactly one vehicle, while the next m constraints state that only one feasible subset of items can be assigned to each vehicle. It is clear that the number of variables in this problem grows extremely rapidly in the number of items considered, which would make even solving the LP-relaxation of (P) a daunting task. However, since we may expect that most variables will have a value of zero in the optimal solution, we apply a column generation approach to solving LP(P), the LP-relaxation of (P).

In this approach, we start by considering only a small number of subsets (columns) for each vehicle. These can, for example, be obtained using a heuristic. After obtaining the solution to the master problem, we solve a pricing problem in order to either identify columns that would provide a better objective value if they would be added to the problem, or conclude that the current solution is optimal. This process is then repeated iteratively until the optimal solution is obtained. To check for optimality of an intermediate solution we consider the dual problem (D) of LP(P). Letting μ_j denote the dual variables associated with constraints (2) ($j = 1, \dots, n$) and δ_i the dual variables associated with constraints (3) ($i = 1, \dots, m$), and noting that we replace the binary constraints by non-negativity constraints in the LP-relaxation of (P), we obtain

$$\begin{aligned} \max \quad & \sum_{j=1}^n \mu_j + \sum_{i=1}^m \delta_i \\ & \text{subject to (D)} \\ & \sum_{j=1}^n \alpha_{ij}^{\ell} \mu_j + \delta_i \leq c_i(\alpha_i^{\ell}) \quad \ell = 1, \dots, N_i, \quad i = 1, \dots, m \\ & \mu_j \text{ free} \quad j = 1, \dots, n \\ & \delta_i \text{ free} \quad i = 1, \dots, m. \end{aligned} \quad (4)$$

Now suppose that we have the optimal primal and, in particular, dual solution to a restricted version of LP(P) and (D) in which only a subset of the columns has been taken into account. Extending the primal optimal solution with the implicit zero values of all omitted variables, we find that

if the corresponding dual solution is feasible for the entire dual problem (D), then the current solution is optimal.

3.2. THE PRICING PROBLEM

The pricing problem aims to find, for each vehicle i , the feasible subset (represented by a binary vector α_i), for which the corresponding constraint in (D) is most violated. Denoting the decision variable representing a feasible subset for vehicle i by z , and the optimal dual solution to the restricted version of LP(P) by $(\hat{\mu}, \hat{\delta})$, the pricing problem for vehicle i can be formulated as follows:

$$\begin{aligned} \max \quad & \sum_{j=1}^n \hat{\mu}_j z_j - c_i(z) \\ & \text{subject to (PP}_i\text{)} \\ & \sum_{j=1}^n D_j z_j \leq C_i F_i \\ & z_j \in \{0, 1\} \quad \text{for } j = 1, \dots, n, \end{aligned}$$

where C_i denotes the capacity of vehicle i , and F_i denotes its maximum frequency. If the optimal solution value of (PP _{i}) is no more than $-\hat{\delta}_i$, then all dual constraints for vehicle i are satisfied. If the optimal solution value of (PP _{i}) exceeds $-\hat{\delta}_i$, the corresponding optimal solution yields a subset for vehicle i that may improve the solution if added to the limited set partitioning problem.

In the remainder of this section, we will develop a branch-and-bound algorithm that can be used to solve the pricing problem (PP _{i}) to optimality. For notational convenience, we will omit the index i indicating the vehicle, and consider a general pricing problem (PP). Similar to often used branch-and-bound strategies for the knapsack problem (see, e.g., [34]), we will branch on the binary variables z . Therefore, each node of the branch-and-bound tree is characterized by a partition of the items in \mathcal{S} into the following three sets: J^0 , J^1 , and J :

$$\begin{aligned} J^0 &= \{j \in \mathcal{S} : z_j \text{ has been fixed to } 0\} \\ J^1 &= \{j \in \mathcal{S} : z_j \text{ has been fixed to } 1\} \\ J &= \{j \in \mathcal{S} : z_j \text{ has not been fixed}\}. \end{aligned}$$

Note that we can set $z_j = 0$ for all items j such that $\hat{\mu}_j < 0$ without loss of optimality, which may significantly reduce the size of the problem. So, we will assume that these items are always included in the set J^0 . We can now find an upper bound on the objective function value of (PP) in a node of the tree as follows. Note that we can bound the function c given in Equation 1 by noting that the fixed costs are given by

$$L(z) = K + \text{TSP}(z)$$

so that we can bound these from below by

$$L(z) \geq K + \text{TSP}(J^1)$$

to obtain

$$c(z) \geq \underline{c}(z)$$

where

$$\underline{c}(z) = \begin{cases} (K + \text{TSP}(J^1))F + \frac{1}{2}h(z)\frac{D(z)}{F} & \text{if } \sqrt{\frac{2D(z)(K+\text{TSP}(J^1))}{h(z)}} \leq \frac{D(z)}{F} \\ \sqrt{2D(z)(K + \text{TSP}(J^1))h(z)} & \text{if } \frac{D(z)}{F} \leq \sqrt{\frac{2D(z)(K+\text{TSP}(J^1))}{h(z)}} \leq C \\ (K + \text{TSP}(J^1))\frac{D(z)}{C} + \frac{1}{2}h(z)C & \text{if } C \leq \sqrt{\frac{2D(z)(K+\text{TSP}(J^1))}{h(z)}}. \end{cases}$$

If the number of items in J is small, we may efficiently solve the problem

$$\max \sum_{j=1}^n \hat{\mu}_j z_j - \underline{c}(z)$$

subject to $(\overline{\text{PP}}^1)$

$$\sum_{j=1}^n D_j z_j \leq CF$$

$$z_j = 0 \quad \text{for } j \in J^0$$

$$z_j = 1 \quad \text{for } j \in J^1$$

$$z_j \in \{0, 1\} \quad \text{for } j \in J$$

by complete enumeration, which will clearly provide a valid upper bound in the current node of the branch-and-bound tree. However, in general we will need to find an upper bound that can be computed more efficiently.

Clearly, we can find an alternative lower bound on the costs by, in addition to using the lower bound on $L(z)$, ignoring the capacity and frequency constraints, which yields

$$c(z) \geq \sqrt{2D(z)(K + \text{TSP}(J^1))h(z)}.$$

Recalling the definition of the aggregate demand and inventory holding cost functions, we may rewrite the lower bound as

$$c(z) \geq \sqrt{2(K + \text{TSP}(J^1)) \sum_{j=1}^n h_j D_j z_j}.$$

An upper bound on the solution value of (PP) given the sets J^0 and J^1 can now be determined by solving the following optimization problem

$$\max \sum_{j=1}^n \hat{\mu}_j z_j - \sqrt{2(K + \text{TSP}(J^1)) \sum_{j=1}^n h_j D_j z_j}$$

subject to $(\overline{\text{PP}}^2)$

$$\begin{aligned} z_j &= 0 && \text{for } j \in J^0 \\ z_j &= 1 && \text{for } j \in J^1 \\ z_j &\in \{0, 1\} && \text{for } j \in J \end{aligned}$$

where we have also ignored the capacity and cardinality constraints. This problem can now be solved efficiently using a result from Huang et al. [31]. This result says that, if the $|J|$ relevant items are renumbered and sorted in non-increasing order of the ratio

$$\frac{\hat{\mu}_j}{h_j D_j} \tag{5}$$

the optimal solution will be of the form

$$z_j^* = \begin{cases} 1 & \text{for } j = 1, \dots, k \\ 0 & \text{for } j = k + 1, \dots, |J| \end{cases}$$

for some $k = 0, \dots, |J|$.

Since our relaxation yields an integral solution, we cannot use this solution to guide the branching. We will instead use a strategy that has been successfully applied to many knapsack and related problems. This is a depth-first-search strategy that first explores the subtree in which the variable corresponding to the most promising item is set to one. For our problem, this means that we first consider the subproblem in which the unassigned item with the largest positive ratio (5) is added to J^1 .

Since any feasible solution to the pricing problem for vehicle i with a value that exceeds $-\hat{\delta}_i$ provides a subset of items that is attractive, it is not strictly necessary to solve the pricing problem to optimality, especially in the early stages of the column generation procedure. We therefore usually implement the branch-and-bound procedure for (PP) heuristically by finding an approximate bound in each node. That is, we find a value that will often, but not necessarily, be an upper bound to the objective function value in the current node of the tree. This approximation is based on considering solutions by sequentially adding items according to the ranking scheme given by the ratio (5). Observing the capacity and cardinality constraints, we then choose the solution that maximizes the objective function of (\overline{PP}^1) . Since this procedure does not necessarily find the optimal solution to (\overline{PP}^1) , the corresponding bound is not exact, and therefore the solution to the pricing problem obtained is not necessarily optimal.

3.3. BRANCHING

We now return to our main problem (P). If the optimal solution of LP(P) obtained using the column generation approach is not integral, we need to use branch-and-price. It remains to discuss the corresponding branching strategy. As has been mentioned by several authors (e.g., [28]), branching

on the subset selection variables y_i^ℓ in the set partitioning formulation is problematic, since excluding a subset from consideration would require finding the 2nd best solution to the pricing problem. However, we can transform the subset selection variables to *assignment variables* that indicate the fraction of an item that is included in a subset:

$$x_{ij} = \sum_{\ell=1}^{N_i} \alpha_{ij}^\ell y_i^\ell.$$

Clearly, x is integral if y is integral. In a node of the branch-and-price tree, we can now branch on fractional assignment variables. This corresponds to requiring or disallowing an item to be replenished by vehicle i .

4. Heuristics

In Section 3 we have developed a branch-and-price algorithm to solve the inventory-routing problem described in Section 2. Clearly, we can only expect to be able to find an exact solution in reasonable time for relatively small problem instances. The computational effort is likely to increase rapidly when the number of items, suppliers, and/or the vehicle capacities increase. In this section we will therefore focus on heuristic approaches to the problem. In particular, we will describe two constructive heuristics that can be used to find an initial feasible solution by constructing routes for the vehicles either sequentially or simultaneously. In addition, we will develop a neighborhood search algorithm that can be used to improve a solution found by the constructive heuristics.

4.1. DISTANCE RATIO (DR) HEURISTIC

Our first heuristic constructs routes sequentially for one vehicle at a time. The idea of the heuristic is to add items to a vehicle whose supplier is (a) located far away from the warehouse, but (b) close to at least one supplier that is already visited in the route. In that case, it is attractive to add the item to the vehicle under consideration rather than supplying this item with another vehicle. Items are added until no more items can be added without violating the capacity and/or frequency constraints. Initially, when the route for a vehicle is empty, this criterion says that we should choose the item that is located furthest away from the warehouse.

When a group of items is assigned to a vehicle and no more items can be added, we estimate the cost associated with the vehicle by solving its associated TSP heuristically. We first find a TSP tour using the Arbitrary Insertion (AI) heuristic (see [36]). To improve the vehicle tour, the 2-opt exchange heuristic studied by Croes [22], Lin [32], and Lin and Kernighan [33] is utilized.

In the remainder, let $d_{j_1 j_2}$ denote the distance (cost) from the supplier of item j_1 to the supplier of item j_2 , for all $j_1, j_2 \in \mathcal{S}$. Similarly, let d_{0j} and d_{j0} denote the distance from the warehouse to the supplier of item j and from the supplier of item j to the warehouse.

DR heuristic

Step 0. Initialize an empty route for the next vehicle.

Step 1. For each ungrouped item that can be added to the vehicle without violating its capacity constraint, say j , determine its *distance-ratio* as the minimum value of $d_{j'j}/d_{0j}$ over all items j' served by the current vehicle. If the current vehicle does not contain any items, let the distance-ratio be $1/d_{0j}$. If no such items exist, go to Step 3.

Step 2. Find the item with the smallest distance-ratio, assign it to the vehicle, and return to Step 1.

Step 3. If all items have been assigned to a vehicle, go to Step 5. Otherwise, if there are available vehicles left, return to Step 0.

Step 4. For each ungrouped item, determine the vehicle to which it can be added with minimal capacity violation. Find the item requiring the smallest violation, assign it to the vehicle, and return to Step 3.

Step 5. Find a TSP tour for all vehicles using AI and 2-opt.

As an alternative, we have explored the possibility of choosing the first item in the vehicle to be the unassigned item that has the smallest replenishment interval when replenished individually. In this case, Steps 0 and 1 in the algorithm are replaced by

Step 0. Initialize an empty route for the next vehicle, and find the ungrouped item with the smallest individual replenishment interval that can be added to this vehicle without violating its capacity constraint. Add that item to the vehicle.

Step 1. For each ungrouped item that can be added to the vehicle without violating its capacity constraint, say j , determine its *distance-ratio* as the minimum value of $d_{j'j}/d_{0j}$ over all items j' served by the current vehicle. If no such items exist, go to Step 3.

4.2. ARBITRARY ITEM INSERTION (AII) HEURISTIC

This heuristic is based on the Arbitrary Insertion (AI) heuristic for the TSP (see [36]). In this heuristic, we start with an empty route for each vehicle. On each iteration, given a partial route for each vehicle, we find the total insertion costs for each unassigned item and each possible *slot* (i.e., a (depot, supplier), (supplier, supplier), or (supplier, depot) pairs) in each

partial route. The insertion costs estimate the additional total costs to be incurred if an item is inserted in a given slot in a route. This estimate is obtained by first finding the traditional TSP insertion costs associated with inserting the item in a route, and next computing the total costs incurred by the vehicle with this additional item. More formally, consider a vehicle, say i , and a given pair of items, say j_1 and j_2 , that are visited consecutively in the current route for that vehicle. Moreover, let $\widehat{L}(i)$ denote the length of the current costs associated with the route. Then, ignoring for simplicity the capacity and frequency constraints, the corresponding insertion costs are:

$$\sqrt{2D(S_i \cup \{j\})(\widehat{L}(i) + d_{ij} + d_{jj_2} - d_{ij_2})h(S_i \cup \{j\})} - \sqrt{2D(S_i)\widehat{L}(i)h(S_i)}.$$

Analogously, we can derive the insertion costs in the presence of capacity and frequency constraints.

AII heuristic

Step 0. Initialize an empty route for each vehicle.

Step 1. Randomly select an unassigned item for insertion, and determine its insertion costs corresponding to each slot in each vehicle's partial route, for each vehicle to which the item can feasibly be assigned. If no such vehicle exists, we consider the insertion slots in the vehicle route for which the capacity constraint will be least violated.

Step 2. Find the minimum insertion cost for this item, and insert the item in the corresponding slot.

Step 3. If all items have not been assigned, return to Step 1. Otherwise, improve, if possible, the TSP tour for each vehicle by applying a 2-opt exchange heuristic and stop.

4.3. VERY LARGE-SCALE NEIGHBORHOOD SEARCH (VLSN)

Neighborhood search algorithms, in which an initial solution is iteratively replaced by an improved solution until no further improvements can be found or some termination criterion is satisfied, are often the most effective approaches available for solving partitioning problems. For the traditional Vehicle Routing Problem, 1- and 2-exchange heuristics have been developed and applied with some success (see, e.g., [44]). However, these methods search for an improved solution in a relatively small neighborhood of the current solution. Much better results may be expected if we are able to search larger neighborhoods. Rather than extending the 1- and 2-exchange heuristics to our inventory-routing problem, we will develop a very large-scale neighborhood (VLSN) search algorithm. Using this technique, very

large neighborhoods can be explored implicitly through the solution of a subproblem, rather than explicitly by enumeration, as is common practice with small neighborhood search methods. This technique has relatively recently been developed and applied with much success to several hard combinatorial optimization problems. For example, the technique has been applied to vehicle routing problems (see [25, 30, 42]), minimum makespan machine scheduling (see [27]) and other scheduling problems (see [42]), the capacitated minimum spanning tree problem (see [5, 6]), and several single-sourcing problems (see [4, 31]). Surveys of VLSN can be found in Ahuja et al. [3, 7].

The VLSN algorithms that we propose can be viewed as extensions of 1- and 2-exchange heuristics for VRPs. In the first algorithm, which we call *Supplier-VLSN* (S-VLSN), we consider a neighborhood of solutions that can be reached by moving groups of items with a common supplier that are currently replenished by one vehicle to another vehicle. In particular, we consider simultaneous moves of this form, where each of a subset of the vehicles exchanges one group of items by another. A solution is called a neighbor of a given solution if it can be reached through a set of moves of the following form: for some sequence of distinct vehicles i_1, \dots, i_k , a group of items is moved from vehicle i_1 to vehicle i_2 , while simultaneously a group of items is moved from vehicle i_2 to vehicle i_3, \dots , a group of items is moved from vehicle i_{k-1} to vehicle i_k , and a group of items is moved from vehicle i_k to vehicle i_1 . This type of exchange is called a cyclic exchange. An even larger neighborhood is obtained when we also consider sets of moves that do not include the last one, that is, one vehicle “loses” a group of items without “gaining” one, while another vehicle “gains” a group of items without “losing” one. Those type of exchanges are called path exchanges. The second algorithm, which we call *Item-VLSN* (I-VLSN), is similar to S-VLSN, with the distinction that we now do not move groups of items with a common supplier, but single items only.

Efficient methods for identifying an improving neighbor without explicit enumeration and evaluation of all neighbors in the neighborhood are based on a characterization of the neighborhood through a so-called improvement graph (see [3, 7]), which captures all information needed to evaluate any exchange. The improvement graph for cyclic exchange can be constructed by creating a node corresponding to each item (or group of items) that is a candidate for exchange. Then, an arc is created from one node to another if it is possible to move the item(s) corresponding to the first node to the vehicle that currently replenishes the item(s) corresponding to the second node, while removing these latter items from their vehicle. The arc costs in the improving graph are defined to be the change in costs due to the move incurred by the “receiving” vehicle. To allow also path exchanges, the improvement graph is extended by a node for each vehicle

as well as a dummy node. Then, an arc with appropriate cost is created from an item-node to a vehicle-node if it is possible to move the item(s) corresponding to the item-node to the vehicle corresponding to the vehicle-node without removing any items from that vehicle. Similarly, an arc with appropriate cost is created from the dummy node to each item-node, modeling the fact that an item may leave a vehicle without being replaced by one. Finally, zero-cost arcs are created from the vehicle-nodes to the dummy node.

Each neighbor is now represented by a so-called subset-disjoint cycle in the improvement graph, that is, a cycle whose nodes correspond to distinct vehicles. (Note that a cycle that does not contain the dummy node corresponds to a cyclic exchange, and a cycle that does contain the dummy node corresponds to a path exchange.) Furthermore, the cost change from the current solution to the neighbor is equal to the total cost of the corresponding cycle in the improvement graph. As shown by Thompson and Orlin [41], and Thomson and Psaraftis [42], the problem of finding an improving neighbor therefore reduces to the problem of finding a negative-cost subset-disjoint cycle in the improvement graph. However, the problem of determining whether there exists a subset-disjoint cycle in the improvement graph is NP-complete, and the problem of finding a negative cost subset-disjoint cycle is NP-hard (see [40, 41, 42]). We will employ heuristics for this problem that have been developed by Ahuja et al. [2, 5, 6], and appear to be highly effective in practice.

Both constructive heuristics that we have described earlier in this section for finding an initial solution may find a solution that violates the capacity constraint of one or more vehicles. If this is the case, we will use the VLSN algorithm described in this section to first solve a Phase I problem with the temporary objective function of minimizing the total constraint violations, i.e., the goal of finding a feasible solution. Next, we will return to the original problem, and employ the VLSN algorithm improve that feasible solution.

5. Computational Results

In order to test the performance of our algorithms, we have performed computational experiments using a randomly generated set of test instances. In this section, we will first discuss the performance of the column generation and exact branch-and-price algorithm on a number of small instances. As was expected, the branch-and-price method is very time-consuming. However, these tests can be used to assess the tightness of the lower bound obtained using the column generation solution to the LP-relaxation of the set partitioning formulation. In the remaining experiments

we compare the performance of our constructive and improvement heuristics, and assess the quality of the solutions obtained by comparing the heuristic costs to the column generation lower bound on the optimal costs.

5.1. GENERATION OF THE TEST INSTANCES AND IMPLEMENTATION

For our tests, we have generated random instances as follows. The demand rate (demand per unit time) for each item is generated from the uniform distribution on $[100, 300]$, and the inventory holding cost rate (cost per unit per unit time) for each item is generated from the uniform distribution on $[1, 15]$. The items are randomly assigned to one of 10 suppliers, while ensuring that each of these manufactures at least one item. The locations of the warehouse and suppliers are generated uniformly in the square $[0, 20]^2 \subset \mathbb{R}^2$, and Euclidean distances are used to measure transportation costs, with unit cost per unit distance traveled.

We have defined a base case where each vehicle has a capacity of $C = 150$ units, the fixed transportation cost is set to $K = 50$, and the maximum number of trips allowed per time unit by each vehicle is $F = 10$. We identify the size of an instance by the number of items, n , and the number of vehicles, m . In our experiments, we have considered four different sizes: 15 items and 3 vehicles, 30 items and 5 vehicles, 40 items and 6 vehicles, and 50 items and 8 vehicles, where the number of vehicles has been chosen to ensure that in most instances a solution with this number of vehicles indeed exists given the capacity constraints. In addition, we have performed a sensitivity analysis of the computational results for changes in the capacity, frequency, and fixed cost parameters, where we have adjusted the number of available vehicles accordingly.

We have implemented all our algorithms and heuristics using the C++ programming language on a PC with a 1.80 GHz Intel Pentium 4 CPU and 240 MB of RAM. To obtain solutions to the LP-relaxation of the set partitioning problem in our column generation procedure, we used the CPLEX 8.1 solver.

5.2. EXPERIMENTS

5.2.1. *Experiment 1: Performance of Branch-and-Price and Quality of the Lower Bounding Procedure*

The goal of our first experiment is to test the computational efficiency of the branch-and-price algorithm and assess the quality of the lower bound obtained from the column generation procedure. For this experiment, we have used the branch-and-price algorithm to obtain optimal solutions for all 15-item instances, as well as the corresponding lower bounds. The results of this test on the 10 instances are given in Table 1.

Table 1. Optimal cost vs. Lower bound ($n = 15, m = 3$)

Problem	LB	Optimal	% deviation
1	2778.1	2778.1	0.00
2	2645.8	2645.8	0.00
3	2545.2	2598.6	2.10
4	2669.7	2761.2	3.43
5	2557.6	2726.0	6.58
6	2563.9	2699.3	5.28
7	2511.3	2526.7	0.61
8	2378.3	2426.2	2.02
9	2556.2	2577.2	0.82
10	2710.0	2825.5	4.26
Average			2.51
Max			6.58
Time (sec)	13.5	13571.3	

We conclude that the branch-and-price algorithm is very time-consuming, even for these small problem instances. However, the column generation procedure is able to find a reasonably tight lower bound to the optimal costs efficiently, with an average gap between the optimal cost and the lower bound of approximately 2.5%, and an average computational time of 13.5 CPU seconds.

5.2.2. Experiment 2: Performance of the Constructive Heuristics and Improvement Algorithms

In our second set of experiments, our goal is to assess the quality of the solutions obtained by our constructive heuristics and improvement algorithms. We first solve each instance using the constructive heuristics. Next, we improve these initial solutions using the I-VLSN and S-VLSN improvement algorithms. We then compare the objective function value found by the heuristic to the lower bound for each instance using the column generation approach. In addition, for the instances with $n = 15$ items we also compare the heuristic costs to the optimal costs. The results of these experiments are given in Tables 2 and 3.

We report the average as well as the maximum cost deviations from the lower bound of the solutions obtained by both constructive heuristics and both VLSN improvement algorithms. The results show that the DR heuristic outperforms the AII heuristic both with respect to the average and the maximum error over all 10 instances. In addition, its performance improves with increasing problem size, while this effect is absent from the AII heuristic. With respect to the VLSN improvement algorithms, I-VLSN outperformed S-VLSN for all cases, except for the largest problem with $n = 50$ and the starting solution found by AII. The I-VLSN algorithm based on

Table 2. Error bounds of solutions obtained using the constructive heuristics and improvement algorithms

Size			Error bound (%)					
n	m		DR	I-VLSN	S-VLSN	AII	I-VLSN	S-VLSN
15	3	Avg	8.81	3.28	4.91	9.98	3.57	5.45
		Max	14.11	6.92	9.68	16.94	6.57	9.33
30	5	Avg	8.25	2.84	5.63	12.18	3.89	4.94
		Max	11.32	6.73	7.46	16.71	6.51	6.46
40	6	Avg	5.53	2.69	3.38	13.00	3.64	4.30
		Max	8.21	3.20	6.08	16.19	4.58	6.16
50	8	Avg	5.43	2.37	3.70	13.31	4.04	3.76
		Max	8.48	3.31	5.64	17.74	7.44	5.00

Table 3. Error with respect to optimal solution of solutions obtained using the constructive heuristics and improvement algorithms

Size			Error bound (%)					
n	m		DR	I-VLSN	S-VLSN	AII	I-VLSN	S-VLSN
15	3	Avg	6.20	0.76	2.37	7.31	1.05	2.89
		Max	14.11	5.26	9.04	15.99	6.04	9.04

the DR solution on average provided the solution with smallest error, with a maximum average error of 3.28%. Moreover, as for the DR heuristic, the effectiveness of I-VLSN increases when the number of items increases.

Finally, in Table 4 we provide the computation time needed for finding the lower bound as well as for the heuristics. We immediately see that the heuristics are extremely efficient, finding a solution in usually less than a second. On the other hand, the time needed for computing the lower bound is very time-consuming, increasing dramatically as the problem size increases. However, as we have seen in Tables 2 and 3, the solution quality of the heuristics is very good, and improves as the problem size increases.

Table 4. Average computation times for the column generation method (LB) and the heuristics

Size		Average computation time (sec)						
n	m	LB	DR	I-VLSN	S-VLSN	AII	I-VLSN	S-VLSN
15	3	13.5	0.000	0.017	0.006	0.000	0.027	0.011
30	5	849.5	0.000	0.136	0.008	0.000	0.235	0.038
40	6	14207.6	0.002	0.217	0.011	0.002	0.525	0.081
50	8	48672.0	0.003	0.433	0.016	0.006	1.378	0.133

Therefore, the need for computing the lower bound in practice decreases as the problem size increases.

5.2.3. Experiment 3: Sensitivity Analysis

In order to investigate the impact of changing some of the problem parameters on the computational performance of our heuristics and algorithms, we have conducted a third set of experiments. Using the 40-item instances as a base case, we have varied the vehicle capacity C from 100 to 200, the maximum number of trips per vehicle F from 8 to 12, and the fixed transportation cost K from 0 to 100. In each case, the number of vehicles is adjusted to suit the capacity constraints. Based on the results in Table 2, we have chosen to perform this sensitivity analysis for the best constructive heuristic, DR, and the best overall method, DR followed by I-VLSN.

The results in Tables 5–7 show that both methods perform better on average for problems with smaller vehicle capacity, smaller maximum number of trips per time unit, or larger fixed transportation cost. The error bound also seems to be more stable across instances when the vehicle capacity or the maximum number of trips is smaller.

6. Concluding Remarks and Future Research

In this paper, we have developed a mathematical formulation model that integrates the inventory replenishment and transportation decisions for an inbound commodity collection system with one warehouse, multiple suppliers, and multiple items. We consider the problem in a deterministic setting, and assume that the items are replenished according to an economic order quantity policy. In order to find the optimal operating parameters for this assumed policy, we develop a set partitioning formulation for the problem and propose a column generation approach that can be used to obtain a lower bound on the objective function value. In order to solve the small size instances to optimality, we also develop a branch-and-price algorithm. Since the branch-and-price algorithm is not scalable, i.e., the solution time requirement increases very quickly as the size of the instance increases, we develop constructive as well as improvement heuristics that efficiently find near-optimal solutions for the problems.

Our computational analysis indicates that our constructive heuristics used in conjunction with one of the proposed VLSN algorithms can find near-optimal solutions very efficiently. The sensitivity analysis has shown that this behavior is robust under changes in various key problem parameters. For smaller instances or with some more time investment, the column generation algorithm may be used to provide a bound on the deviation of the cost of the heuristic solution from the optimal cost.

Table 5. Error bounds when the vehicle capacity is varied

C	m		Error bound (%)	
			DR	I-VLSN
100	9	Avg	5.34	1.91
		Max	6.66	3.05
150	6	Avg	5.53	2.69
		Max	8.21	3.20
200	5	Avg	7.45	2.89
		Max	11.25	4.59

Table 6. Error bounds when the maximum number of trips allowed is varied

F	m		Error bound (%)	
			DR	I-VLSN
8	8	Avg	5.45	2.24
		Max	6.63	2.67
10	6	Avg	5.53	2.69
		Max	8.21	3.20
12	5	Avg	5.12	2.93
		Max	10.00	4.50

Table 7. Error bounds when the fixed transportation cost is varied

K		Error bound (%)	
		DR	I-VLSN
0	Avg	4.34	3.49
	Max	7.07	5.96
20	Avg	5.26	3.29
	Max	7.93	5.95
50	Avg	5.53	2.69
	Max	8.21	3.20
100	Avg	4.06	2.04
	Max	6.79	2.95

In our current analysis, we assume that the retailers operate according to JIT principle. This allows us to use economic order quantities for replenishing the inventory for the items. An interesting extension to our work would consider a system where the retailers face stochastic demand. In such a system, we could assume that each retailer uses a standard periodic review order-up-to policy. In this case, the amount that needs to be delivered to a retailer is a random variable, which significantly complicates the

problem in the presence of capacity constraints for the vehicles. Alternatively, we could consider a periodic review, fixed order quantity policy. Although the behavior of such a policy is not known even for single-item single-retailer systems, it has the advantage that the quantities delivered to the retailers are deterministic.

References

1. Adelman, D. (2003), Price-directed replenishment of subsets: methodology and its application to inventory routing, *Manufacturing and Service Operations Management* 5, 348–371.
2. Ahuja, R.K., Boland, N. and Dumitrescu, I. (2001), Exact and heuristic algorithms for the subset disjoint minimum cost cycle problem. Working paper, Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida.
3. Ahuja, R.K., Ergun, O., Orlin, J.B. and Punnen, A. (2002), A survey of very large scale neighborhood search techniques, *Discrete Applied Mathematics* 123, 75–102.
4. Ahuja, R.K., Huang, W., Romeijn, H.E. and Romero Morales, D. (2002), A heuristic approach to the multi-period single-sourcing problem with production and inventory capacities and perishability constraints. Research Report 2002-2, Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida.
5. Ahuja, R.K., Orlin, J.B. and Sharma, D. (2001), A composite very large-scale neighborhood structure for the capacitated minimum spanning tree problem, *Operations Research Letters* 31, 185–194.
6. Ahuja, R.K., Orlin, J.B. and Sharma, D. (2001), Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem, *Mathematical Programming* 91(1), 71–97.
7. Ahuja, R.K., Orlin, J.B. and Sharma, D. (2000), Very large scale neighborhood search, *International Transactions in Operations Research* 7, 301–317.
8. Anily, S. and Federgruen, A. (1990), One warehouse multiple retailer systems with vehicle routing costs, *Management Science* 36(1), 92–114.
9. Anily, S. and Federgruen, A. (1993), Two-echelon distribution systems with vehicle routing costs and central inventories, *Operations Research* 41(1), 37–47.
10. Anily, S. (1994), The general multi-retailer EOQ problem with vehicle routing costs, *European Journal of Operational Research* 79, 451–473.
11. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P. and Vance, P. (1998), Branch-and-Price: column generation for solving huge integer programs. *Operations Research* 46(3), 316–329.
12. Ben-Khedher, N. and Yano, C.A. (1994), The multi-item joint replenishment problem with transportation and container effects, *Transportation Science* 28(1), 37–54.
13. Bertazzi, L. and Speranza, M.G. (1999), Minimizing logistic costs in multistage supply chains, *Naval Research Logistics* 46, 399–417.
14. Bramel, J. and Simchi-Levi, D. (1997), *The Logic of Logistics: Theory, Algorithms, and Applications for Logistics Management*. Springer-Verlag, New York.
15. Buffa, F.P. and Munn, J.R. (1990), Multi-item grouping algorithm yielding near-optimal logistics cost, *Decision Sciences* 21(1), 14–34.
16. Burns, L.D., Hall, R.W., Blumenfeld, D.E. and Daganzo, C.F. (1985), Distribution strategies that minimize transportation and inventory costs, *Operations Research* 33(3), 469–490.
17. Çetinkaya, S. and Lee, C.-Y. (2000), Stock replenishment and shipment scheduling for vendor managed inventory systems, *Management Science* 46(2), 217–232.

18. Chan, L.M.A., Federgruen, A. and Simchi-Levi, D. (1998), Probabilistic analyses and practical algorithms for inventory-routing models, *Operations Research* 46(1), 96–106.
19. Chan, L.M.A. and Simchi-Levi, D. (1998), Probabilistic analyses and algorithms for three-level distribution systems, *Management Science* 44(11), 1562–1576.
20. Chaovalitwongse, P., Romeijn, H.E. and Pardalos, P.M. (2002), A scenario-based heuristic for a capacitated transportation-inventory problem with stochastic demands. In: Kontoghiorghes, E., B. Rustem and S. Siokos (eds.), *Computational Methods in Decision-Making, Economics and Finance*. Kluwer Academic Publishers, Dordrecht, The Netherlands.
21. Chien, T.W., Balakrishnan, A. and Wong, R.T. (1989), An integrated inventory allocation and vehicle routing problem, *Transportation Science* 23(2), 67–76.
22. Croes, G.A. (1958), A method for solving traveling-salesman problems, *Operations Research* 6, 791–812.
23. Dror, M., Ball, M. and Golden, B. (1985/1986), A computational comparison of algorithms for the inventory routing problem, *Annals of Operations Research* 4, 3–23.
24. Dror, M. and Ball, M. (1987), Inventory/routing: reduction from an annual to a short-period problem, *Naval Research Logistics* 34, 891–905.
25. Fahrion, R. and Wrede, M. (1990), On a principle of chain exchange for vehicle routing problems (I-VRP), *Journal of the Operational Research Society* 41, 821–827.
26. Federgruen, A. and Zipkin, P. (1984), A combined vehicle routing and inventory allocation problem, *Operations Research* 32(5), 1019–1037.
27. Frangioni, A., Necciari, E. and Scutella, M.G. (2000), Multi-exchange algorithms for the minimum makespan machine scheduling problem. Technical report, Dipartimento di Informatica, University of Pisa, Pisa, Italy.
28. Freling, R., Romeijn, H.E., Morales, D. and Wagelmans, A.P.M. (2003), A branch-and-price algorithm for the multi-period single-sourcing problem, *Operations Research* 51, 922–939.
29. Gallego, G. and Simchi-Levi, D. (1990), On the effectiveness of direct shipping strategy for the one-warehouse multi-retailer R -systems, *Management Science* 36(2), 240–243.
30. Gendreau, M., Guertin, F., Potvin, J.-Y. and Seguin, R. (1998), Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. CRT Research Report No. CRT-98-10, Centre for Research on Transportation, University of Montréal, Montréal, Canada.
31. Huang, W., Romeijn, H.E. and Geunes, J. (2004), The continuous-time single-sourcing problem with capacity expansion opportunities. Technical Report, Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida.
32. Lin, S. (1965), Computer solutions of the traveling salesman problem, *Bell Systems Technical Journal* 44, 2245–2269.
33. Lin, S. and Kernighan, B.W. (1973), An efficient heuristic algorithm for the traveling-salesman problem, *Operations Research* 21, 498–516.
34. Martello, S. and Toth, P. (1990), *Knapsack Problems, Algorithms and Computer Implementations*. John Wiley & Sons, New York.
35. Qu, W.W., Bookbinder, J.H. and Iyogun, P. (1999), An integrated inventory-transportation system with modified periodic policy for multiple products, *European Journal of Operational Research* 115, 254–269.
36. Rosenkrantz, D.J., Stearns, R.E. and Lewis, P.M. (1977), An analysis of several heuristics of traveling salesman problem, *SIAM Journal of Computing* 6, 563–581.
37. Savelsbergh, M.W.P. (1997), A branch-and-price algorithm for the generalized assignment problem, *Operations Research* 6, 831–841.
38. Shen, Z.J., Coullard, C. and Daskin, M.S. (2003), A joint location-inventory model, *Transportation Science* 37(1), 40–55.

39. Silver, E.A., Pyke, D.F. and Peterson, R. (1998), *Inventory Management and Production Planning and Scheduling*. John Wiley & Sons, New York.
40. Thompson, P.M. (1988), Local Search Algorithms for Vehicle Routing and Other Combinatorial Problems. Ph.D. Thesis, Operations Research Center, MIT, Cambridge, Massachusetts.
41. Thompson, P.M. and Orlin, J.B. (1989), *The Theory of Cyclic Transfers*. Working Paper No. OR 200-89, MIT, Cambridge, Massachusetts.
42. Thompson, P.M. and Psaraftis, H.N. (1993), Cyclic transfer algorithms for multi-vehicle routing and scheduling problems, *Operations Research* 41, 935–946.
43. Toptal, A., Çetinkaya, S. and Lee, C.-Y. (2003), The buyer-vendor coordination problem: modeling inbound and outbound cargo capacity and costs, *IIE Transactions on Logistics and Scheduling* 35(11), 987–1002.
44. Toth, P. and Vigo, D. (2002), *The Vehicle Routing Problem*, Society for Industrial and Applied Mathematics, Philadelphia, USA.
45. Viswanathan, S. and Mathur, K. (1997), Integrating routing and inventory decisions in one-warehouse multi-retailer multi-product distribution systems, *Management Science* 43(3), 294–312.
46. Yano, C. and Gerchak, Y. (1989), Transportation contracts and safety stocks for just-in-time deliveries, *Manufacturing and Service Operations Management* 2, 314–330.