Algorithms for the Multi-item Multi-vehicles Dynamic Lot Sizing Problem

Shoshana Anily,[1] Michal Tzur[2]

[1] *Faculty of Management, Tel Aviv University, Tel Aviv, Israel. E-mail: anily@post.tau.ac.il*

[2] *Department of Industrial Engineering, Tel Aviv University, Tel Aviv, Israel. E-mail: tzur@eng.tau.ac.il*

Received 31 August 2004; revised 31 August 2005; accepted 30 September 2005
DOI 10.1002/nav.20129
Published online 28 December 2005 in Wiley InterScience (www.interscience.wiley.com).

Abstract: We consider a two-stage supply chain, in which multi-items are shipped from a manufacturing facility or a central warehouse to a downstream retailer that faces deterministic external demand for each of the items over a finite planning horizon. The items are shipped through identical capacitated vehicles, each incurring a fixed cost per trip. In addition, there exist item-dependent variable shipping costs and inventory holding costs at the retailer for items stored at the end of the period; these costs are constant over time. The sum of all costs must be minimized while satisfying the external demand without backlogging.

In this paper we develop a search algorithm to solve the problem optimally. Our search algorithm, although exponential in the worst case, is very efficient empirically due to new properties of the optimal solution that we found, which allow us to restrict the number of solutions examined. Second, we perform a computational study that compares the empirical running time of our search methods to other available exact solution methods to the problem. Finally, we characterize the conditions under which each of the solution methods is likely to be faster than the others and suggest efficient heuristic solutions that we recommend using when the problem is large in all dimensions. © 2005 Wiley Periodicals, Inc. Naval Research Logistics 53: 157–169, 2006.

Keywords: inventory/transportation; multi-item; search algorithm

1. INTRODUCTION AND LITERATURE REVIEW

In multi-stage supply chains, items must be shipped from one stage to the next, incurring transportation costs that are sometimes quite significant relative to the total item's cost. Typically, a vehicle that is used to transfer items from one stage to the next incurs a fixed cost, which is independent of the exact identity of the items. However, the existence of multiple items poses a challenge in determining their composition in each delivery, since different items typically face a different stream of external demands and incur different holding cost rates. Moreover, despite the economies of scale associated with dispatching full vehicles, it may sometimes be beneficial to dispatch a partially loaded vehicle.

We consider a two-stage supply chain, in which multiple items are shipped from a manufacturing facility or a central warehouse to a downstream retailer that faces deterministic external demand for each of the items over a finite planning horizon. The items are shipped through identical capacitated vehicles (any number of vehicles per period may be used), each incurring a fixed cost per trip. In addition, there exist item-dependent variable shipping costs as well as inventory holding costs at the retailer for items stored at the end of the period; these cost parameters are constant over time. The sum of all costs must be minimized while satisfying the external demand without backlogging.

We refer to this problem as the *MIMV* (*Multi-Item with Multiple Vehicles*) problem, as coined by Anily and Tzur [1]. Other related problems may involve only one item (SIMV) or a limit of one vehicle per period only (MISV). The special case of one item and one capacitated vehicle is known as the Capacitated Dynamic Lot Sizing (CDLS) Problem; see, e.g., Florian and Klein [3]. More recently, Kaminsky and Simchi-Levi [5] studied a variation of the CDLS, arising from a two-stage manufacturing model, and Van Hoesel et al. [10] studied a serial supply chain in which production, inventory, and transportation decisions are integrated in the presence of production capacities. In the production setting where the problem is applicable in a similar way, the content of a vehicle is referred to as a *batch.* A literature review on capacitated dynamic lot sizing problems, with and without batching considerations, can be

Correspondence to: S. Anily (anily@post.tau.ac.il); M. Tzur (tzur@eng.tau.ac.il)

found in the paper by Anily and Tzur [1]. Here we only mention the most relevant and recent works.

Most of the literature deals with the single-item case. The SIMV problem was investigated by Pochet and Wolsey [8], who considered time-varying setup, inventory holding, and variable production costs. The authors designed an $O(T^3)$ algorithm, where T is the number of periods, which is based on finding a shortest path in an appropriately defined network. Lee [6] addressed the SIMV problem in which there exists a separate setup cost per order and presented an $O(T^4)$ procedure for the problem.

The literature on problems that consider multi-items is quite sparse. Federgruen, Meissner, and Tzur [2] studied the MISV problem, where the capacity limits, as well as the cost parameters, vary over time. They developed and analyzed a class of so-called *progressive interval heuristics,* which under mild parameter conditions can be designed to be ε-optimal for any $\varepsilon > 0$, with a running time that is polynomially bounded in the size of the problem. Pryor, White, and Kapuscinski [9] presented a heuristic for the MIMV problem, which generalizes a heuristic algorithm due to Lippman [7] for the single-item case. They also presented a generalization of a search algorithm, which determines the optimal solution for the problem; no complexity bound is specified and no computational results are available with respect to their algorithm. Yano and Newman [11] analyzed a more general problem than the MIMV problem, in which the demand for the items dynamically becomes available. However, their algorithm doesn't necessarily generate the optimal solution for the special case of our problem, as demonstrated in Section 2. Anily and Tzur [1] developed for the MIMV problem an exact dynamic programming algorithm, which is based on properties of the optimal solution and is using an innovative way to partition the problem into sub-problems. The dynamic programming algorithm is polynomial for a fixed number of items, but exponential when the number of items is part of the input.

The complexity of the MISV and MIMV problems when the number of items is part of the input is still an open question, even when the capacities and cost parameters are constant over time. This is in contrast to the CDLS problem (single item with a single batch) that is polynomial for the constant capacity case (see [3]), but is known to be NP-hard for the time-varying capacities case; see [4]. Anily and Tzur [1] proved that the MIMV problem is at least as hard as the MISV problem, both under constant capacity and cost parameters, by showing how a solution to the former may be used to obtain a solution to the latter. Therefore, the results obtained in this paper for the MIMV problem are applicable for the MISV problem as well.

In this paper, we analyze the MIMV problem and provide the following contributions. First, we develop a search algorithm to solve the problem optimally. Our search algorithm, although exponential in the worst case, is very efficient empirically due to new properties of the optimal solution that we found, which allow us to reduce significantly the number of solutions examined. Second, we perform a computational study that compares the empirical running times of available exact solution methods to the MIMV problem, including our new search algorithm, the DP algorithm of Anily and Tzur [1], and the solution of the CPLEX algorithm to a straightforward MILP formulation of the problem. We characterize the conditions under which each of the above algorithms is likely to be faster than the others and suggest efficient heuristic methods that we recommend using when the problem is large in all dimensions.

The rest of the paper is organized as follows: in the next section we introduce the notation and review known results for the problem, which are needed later in the paper. In Section 2 we present our new search algorithm, which is an enumeration procedure, based on two simple extreme solutions and their properties. In addition, we propose that the two extreme solutions may be used as a basis for heuristics. Section 3 is devoted to a numerical study of our search algorithm, as well as its comparison to the recent DP algorithm of Anily and Tzur [1] and the CPLEX algorithm applied to a straightforward MILP formulation of the problem. We also examine the efficiency of the heuristic procedures. Finally, in Section 4 we draw conclusions.

Notation and Preliminaries

The MIMV problem is defined by the following parameters:

M = number of items;
T = number of periods in the planning horizon;
d_{it} = demand for item i in period t; we assume that all demands are integers.
p_i = cost of shipping a unit of item i;
h_i = cost of holding in inventory at the retailer a unit of item i at the end of each period; $h_i \geq 0$.
K = setup cost of dispatching a vehicle (or part of it);
C = capacity of a vehicle, i.e., the number of units that may be loaded in one vehicle.

We assume that all items have the same weight/volume; therefore, the identity of the items in the vehicle is not important for capacity considerations. All demands must be met on time (no backlogging allowed) either from shipment in the same period or from the period's initial inventory. Any number of vehicles may be used in each period; therefore, the problem is always feasible. The objective is to find an optimal shipping policy that minimizes the sum of dispatching, variable shipping, and holding costs.

In the following we present a straightforward MILP formulation for the MIMV problem. In Section 3 we compare the solution obtained by applying CPLEX to this formulation to alternative algorithms for the MIMV problem.

Let

$X_i(t)$ = shipping quantity of item i in period t, $1 \le i \le M$ and $1 \le t \le T$;
$Y(t)$ = number of vehicles dispatched in period t, $1 \le t \le T$;
$I_i(t)$ = inventory of item i at the end of period t, $1 \le i \le M$ and $0 \le t \le T$.

We assume without loss of generality $I_i(0) = 0$ and $I_i(T) = 0$ for $1 \le i \le M$.

We also define

$X(t)$ = total shipping quantity of all items in period t, i.e., $X(t) = \sum_{i=1}^{M} X_i(t)$, $1 \le t \le T$;
$I(t)$ = total inventory of all items at the end of period t, i.e., $I(t) = \sum_{i=1}^{M} I_i(t)$, $1 \le t \le T$.

$D_i(t_1, t_2) = \sum_{\tau=t_1}^{t_2-1} d_{i\tau}$ and $D(t_1, t_2) = \sum_{i=1}^{M} D_i(t_1, t_2)$ be the total demand in periods $t_1, \ldots, t_2 - 1$ where $D_i(t, t) = 0$ $\forall i, t$.

Then, a MILP formulation of the MIMV problem, denoted (P1), is

$$\text{(P1)} \quad \text{Min} \sum_{t=1}^{T} \left(\sum_{i=1}^{M} (p_i X_i(t) + h_i I_i(t)) + KY(t) \right)$$

$$s.t.$$

$$I_i(t) = I_i(t-1) + X_i(t) - d_{it} \qquad 1 \le t \le T, 1 \le i \le M$$
$$X(t) = \sum_{i=1}^{M} X_i(t) \qquad 1 \le t \le T$$
$$X(t) \le CY(t) \qquad 1 \le t \le T$$
$$I_i(0) = 0,\ I_i(T) = 0 \qquad 1 \le i \le M$$
$$I_i(t) \ge 0,\ X_i(t) \ge 0 \qquad 1 \le t \le T, 1 \le i \le M$$
$$Y(t) \text{ integer} \qquad 1 \le t \le T.$$

Since the variable production costs are constant over time and the total number of units shipped is given by the total demand, the total variable shipment cost of a feasible solution is constant and therefore ignored in our future analysis. It remains to consider the setup cost for dispatching vehicles and the holding costs. In the rest of this section we state results that were proved by Anily and Tzur [1] and are needed in our future analysis.

We use the term "a full vehicle" to refer to a vehicle that is loaded with C units and the term "a partial vehicle" to refer to a non-empty vehicle loaded with fewer than C units. Therefore, a shipment of $X(t)$ units in any period t consists of $\lfloor X(t)/C \rfloor$ full vehicles. If $X(t) \bmod C > 0$ then the shipment consists, in addition, of a partial vehicle of quantity $X(t) \bmod C$.

We assume that the items are numbered in a non-descending order of their holding cost rates, i.e., $0 < h_1 < h_2 < \cdots < h_M$. This is without loss of generality since different items with identical holding cost rates can be combined into a single item; they are identical for all computational purposes.

LEMMA 1: For any optimal shipping quantities $X(1), \ldots, X(T)$ and associated optimal inventory quantities $I_i(t)$, $1 \le i \le M$, and $1 \le t \le T$, the following hold:

(i) $I(t-1)(X(t) \bmod C) = 0 \qquad 1 \le t \le T$

(ii) $$X(t) - \sum_i (d_{it} - I_i(t-1))^+ < \begin{cases} X(t) \bmod C & \text{if } X(t) \bmod C > 0 \\ C & \text{if } X(t) \bmod C = 0 \end{cases} \qquad 1 \le t \le T$$

(iii) $X(t) \le \lceil D(t, t+1)/C \rceil C \qquad 1 \le t \le T$

(iv) $I_M(t) < C \qquad 1 \le t \le T.$

PROOF: See Lemma 1 by Anily and Tzur [1].

LEMMA 2: Given a vector $(X(1), \ldots, X(T))$ of aggregate (over items) shipping quantities in each period, the *Scheduling Algorithm,* described in Appendix A, computes the best detailed schedule for each period, that is, determines the shipment quantities of each item in each period. Its complexity is $O(TM)$ given that the items are ordered in a non-descending order of their inventory holding costs.

PROOF: See the discussion following Lemma 2 in [1].

An optimal policy may be obtained by solving a shortest path problem on a network with nodes $1, 2, \ldots, T+1$. In the proposed network there exist arcs connecting pairs of nodes t_1 and t_2 where $1 \le t_1 < t_2 \le T + 1$; An arc connecting node t_1 to node t_2 represents the best schedule between period t_1 and period $t_2 - 1$ given $I(t_1 - 1) = I(t_2 - 1) = 0$ and $I(\tau) > 0$ for $t_1 - 1 < \tau < t_2 - 1$. Its cost consists of all setup and inventory holding costs incurred in periods $t_1, t_1 + 1, \ldots, t_2 - 1$. We know from Lemma 1(i) that the total setup cost of arc (t_1, t_2) is $K\lceil D(t_1, t_2)/C \rceil$ since except for period t_1 in which a partial vehicle may be dispatched, the shipping in all other periods should be in full vehicles.

It remains to determine the total holding costs on the arcs, which turns out to be the hard part, whose complexity is still unknown; see the Introduction. Given the optimal holding (and therefore total) cost on each arc, the optimal solution can be obtained by applying a shortest path algorithm that requires $O(T^2)$ time. Our search algorithm, described in the next section, focuses on finding the optimal holding cost of

an arc (t_1, t_2), which satisfies $I(t_1 - 1) = I(t_2 - 1) = 0$ and $I(\tau) > 0$ for $t_1 - 1 < \tau < t_2 - 1$. It uses the result of the next lemma, which identifies arcs in the network that violate properties of the optimal solution and thus can be removed.

LEMMA 3: Suppose that the optimal shipping quantities $X^*(1), \ldots, X^*(T)$ imply that arc (t_1, t_2) is used in the optimal solution, that is, the associated inventory levels satisfy $I(t_1 - 1) = I(t_2 - 1) = 0$ and $I(\tau) > 0$ for $t_1 - 1 < \tau < t_2 - 1$. Then,

(a) $X^*(t_1) \bmod C = D(t_1, t_2) \bmod C$
(b) $D(t_1, t_1 + 1) \le X^*(t_1) \le \lceil D(t_1, t_1 + 1)/C \rceil C$. Thus, potential arcs in the network that don't satisfy parts (a) and (b) of the lemma may be removed. Equivalently, we can state that arcs (t_1, t_2) that don't satisfy parts (c) and (d) below may be removed from the network:
(c) If $D(t_1, t_2) \bmod C \neq 0$ then $D(t_1, t_2) \bmod C \ge D(t_1, t_1 + 1) \bmod C$.
(d) If $D(t_1, t_1 + 1) \bmod C = 0$ then $t_2 = t_1 + 1$.

PROOF: See Lemma 5 and its corollaries in [1].

2. THE SEARCH ALGORITHM

In this section we develop a search algorithm to find an optimal shipping schedule and associated holding cost of an arc in the network defined in the previous section. Although the search algorithm is non-polynomial in general, we show in the next section that it is capable of solving quite large problems. This is achieved due to properties of the optimal solution that we identify (in addition to those mentioned in the previous section), which allow us to consider only a small fraction of all possible solutions. In Section 3 we present a numerical study that analyzes the factors that most affect the running time of the search algorithm and compares it to two other methods.

The search algorithm is based on the observation that finding the holding cost of an arc (t_1, t_2) in the network is determined by finding a vector $(X(t_1), \ldots, X(t_2 - 1))$ of aggregate (over all items) shipping quantities in each period. We refer to such a vector as a *schedule.* Given this vector, the best *detailed schedule* (i.e., by items) for each period can be obtained by applying the Scheduling Algorithm, discussed in the previous section. Therefore, it remains to find the best vector of aggregate shipping quantities for an arc in the network.

Below we present two specific schedules, which will be used as a basis for constructing candidates for the best vector of aggregate shipping quantities. The first one, constructed by Algorithm_DS, described below, is called a *delaying schedule* (DS) since it delays the shipping of vehicles as much as possible. The second schedule, constructed by Algorithm_AS, described below, is called an *advancing schedule* (AS) since it ships vehicles as early as possible.

The schedules DS and AS ship in each period the minimum number of units that is required in order to cover the remaining demand in the period (given the shipments and item allocation in previous periods), while satisfying Lemmas 1 and 3. The difference between the two schedules can be viewed as a difference in the priority rules used for allocation of units shipped for future periods: schedule DS first gives priority to items required in early periods, and within a period it gives priority to the least expensive items. Schedule AS first gives priority to the least expensive items, and for a given item it gives priority to early periods. The actual allocation of the items, that is, their shipment quantities in each period, as well as the total cost incurred in periods $t_1, \ldots, t_2 - 1$ for schedules DS and AS, are obtained by applying the Scheduling Algorithm on these schedules for periods $t_1, \ldots, t_2 - 1$.

It may seem at first that schedule DS may be optimal, as it delays shipments as much as possible. Indeed, for the SIMV problem (i.e., $M = 1$) the DS schedule is optimal for all feasible arcs. Yano and Newman [11] proposed such a schedule for solving the MIMV problem. However, the following example by Pryor, White, and Kapuscinski [9] demonstrates that this schedule is not necessarily optimal for all arcs in the network. Consider an arc of four periods and two items with $C = 10$, $h_1 = 1$, $h_2 = 100$, demand for item 1: 0, 0, 6, 6 in periods 1–4, respectively, and demand for item 2: 4, 4, 0, 0 in periods 1–4, respectively. According to schedule DS, which delays shipment as much as possible, we ship 10 units in periods 1 and 3 and none in periods 2 and 4. More specifically, we ship 8 units of item 2 and 2 units of item 1 in period 1 and 10 units of item 1 in period 3. However, the optimal solution for this instance is to ship 10 units in each of the first two periods (4 units of item 2 and 6 units of item 1 in each period).

We use the same example to demonstrate the mechanism of schedule AS. In period 1 we ship a total of 10 units according to Lemma 3 and need to determine how to allocate these units between the two items. We first allocate 4 units to item 2 since those units are demanded in period 1. Then, as priority is given to the least expensive item, the remaining 6 units are of item 2. In period 2 we ship again a total of 10 units, since no item 2 units are available in inventory. The allocation is again 4 units of item 2 (demanded in period 2) and 6 units of item 1 (the remaining demand). For this example, schedule AS achieves the optimal solution.

Indeed, an optimal schedule must balance between shipping the least expensive items and shipping the items that

are required early. Assuming zero inventory at the beginning of periods t_1 and t_2, we denote by $XB(t_1), \ldots, XB(t_2 - 1)$ the schedule constructed by Algorithm_DS for periods $t_1, \ldots, t_2 - 1$ (referred to as schedule B), and by $XA(t_1), \ldots, XA(t_2 - 1)$ the schedule constructed by Algorithm_AS for periods $t_1, \ldots, t_2 - 1$ (referred to as schedule A). We denote the corresponding ending inventories at periods $t_1, \ldots, t_2 - 1$ by $IB(t_1), \ldots, IB(t_2 - 1)$, and by $IA(t_1), \ldots, IA(t_2 - 1)$, respectively.

A formal description of the algorithms is given in Appendix B. Lemma 4 below presents important properties of schedules DS and AS on an arc (t_1, t_2), which are the basis for a central result stated in Theorem 1. These results are used in enumerating all candidates for optimal schedules on a specific arc.

LEMMA 4:

(i) Schedules B and A are both feasible for arc (t_1, t_2).
(ii) The ending inventory in each of the periods $t_1, t_1 + 1, \ldots, t_2 - 2$ under schedule B (representing DS) is less than C, i.e., $IB(t) < C$ for all $t_1 \le t \le t_2 - 2$.
(iii) Any optimal production schedule $Z(t_1), Z(t_1 + 1), \ldots, Z(t_2 - 1)$ for arc (t_1, t_2) satisfies $Z(t_1) = XB(t_1) = XA(t_1)$, and $\sum_{\tau=t_1}^{t_2-1} Z(\tau) = \sum_{\tau=t_1}^{t_2-1} XB(\tau) = \sum_{\tau=t_1}^{t_2-1} XA(\tau)$. Moreover, if we let $IZ(t)$ be the inventory level at the end of period t for schedule Z, then $IB(t) \bmod C = IA(t) \bmod C = IZ(t) \bmod C$ for all $t_1 \le t \le t_2 - 2$.

PROOF:

(i) Define for each period and each item the *item's net demand* as the demand minus the available inventory at the beginning of the period of that item, if positive, and zero otherwise. Define also the *period's net demand* as the sum of all items' net demand in that period. Then, schedules B and A are feasible by construction, since in every period the amount shipped according to these schedules is at least as large as the period's net demand.
(ii) For period t_1, the claim is satisfied by Lemma 3(b). For other periods it is satisfied by construction of schedule B, since the units in inventory are first allocated to satisfy the next period's demand, and the shipment quantity in each period equals the minimum number of full vehicles that satisfy the net demand of the period.
(iii) The equation $XB(t_1) = XA(t_1)$ is true by construction, and according to Lemma 3(a) any optimal schedule Z for arc (t_1, t_2) must satisfy $Z(t_1) = XB(t_1) = XA(t_1)$. Then, the total shipping quantity in periods $t_1, \ldots, t_2 - 1$ for schedules B, A, and Z must be equal to the total demand in these periods, namely, $D(t_1, t_2)$; therefore, $\sum_{\tau=t_1}^{t_2-1} Z(\tau) = \sum_{\tau=t_1}^{t_2-1} XB(\tau) = \sum_{\tau=t_1}^{t_2-1} XA(\tau)$. Moreover, both B and A (by construction) and Z (by Lemma 1(i) and 1(ii)) ship in every period t, $t_1 < t \le t_2 - 1$ only *full* vehicles (the minimum number of full vehicles that covers the net demand in the period). Thus, $IB(t)$, $IA(t)$, and $IZ(t)$ have the same residual modulo C for $t_1 \le t \le t_2 - 2$. □

We are now ready to state in Theorem 1 a central result that provides bounds on the aggregate quantities (over all items) shipped by an optimal solution in each period.

THEOREM 1: Any optimal schedule $Z(t_1), Z(t_1 + 1), \ldots, Z(t_2 - 1)$ for arc (t_1, t_2) satisfies $\sum_{\tau=t_1}^{t} XB(\tau) \le \sum_{\tau=t_1}^{t} Z(\tau) \le \sum_{\tau=t_1}^{t} XA(\tau)$ for every t, $t_1 \le t \le t_2 - 1$.

PROOF: Note first that by Lemma 4(iii), $Z(t_1) = XB(t_1) = XA(t_1)$, and $\sum_{\tau=t_1}^{t_2-1} Z(\tau) = \sum_{\tau=t_1}^{t_2-1} XB(\tau) = \sum_{\tau=t_1}^{t_2-1} XA(\tau)$; therefore, we must prove the theorem for $t_1 + 1 \le t \le t_2 - 2$. Let t be in that interval, that is: $t_1 + 1 \le t \le t_2 - 2$. Note that since $IB(t) < C$ (Lemma 4(ii)) and $IB(t)$ and $IZ(t)$ have the same residual modulo C (Lemma 4(iii)) then either $IB(t) < IZ(t)$ or $IB(t) = IZ(t)$. Combining that with the fact that the consumed demand under both schedules is identical proves that $\sum_{\tau=t_1}^{t} XB(\tau) \le \sum_{\tau=t_1}^{t} Z(\tau)$.

It remains to show that $\sum_{\tau=t_1}^{t} Z(\tau) \le \sum_{\tau=t_1}^{t} XA(\tau)$ for any t, $t_1 + 1 \le t \le t_2 - 2$. Assume by contradiction that there exists some t, $t_1 + 1 \le t \le t_2 - 2$, for which $\sum_{\tau=t_1}^{t-1} Z(\tau) \le \sum_{\tau=t_1}^{t-1} XA(\tau)$ and $\sum_{\tau=t_1}^{t} Z(\tau) > \sum_{\tau=t_1}^{t} XA(\tau)$. Our assumption implies that $IZ(t - 1) \le IA(t - 1)$ and $IZ(t) > IA(t)$, and therefore $Z(t) > XA(t)$.

Denote by $IZ^1(t - 1)$ and $IZ^2(t - 1)$ the units out of $IZ(t - 1)$ that are allocated to period t and to periods later than t, respectively. Define similarly $IA^1(t - 1)$ and $IA^2(t - 1)$ as the units out of $IA(t - 1)$ that are allocated to period t and to periods later than t, respectively. Assuming as above $IZ(t - 1) \le IA(t - 1)$, we distinguish between two cases:

(a) $IZ^2(t - 1) \le IA^2(t - 1)$. In this case, since under schedule Z no more than $C - 1$ units are shipped in period t to inventory and $IZ(t) \bmod C = IA(t) \bmod C$, it is impossible to obtain $IZ(t) > IA(t)$, a contradiction to the assumption.
(b) $IZ^2(t - 1) > IA^2(t - 1)$. This implies that $IZ^1(t - 1) < IA^1(t - 1)$. Let KA and KZ be the most expensive item held in inventory at the end of period $t - 1$ under schedules A and Z, respectively. Then, $IZ^1(t - 1) < IA^1(t - 1)$ implies $KZ \le KA$. Now, according to the definition of schedule A the entire

demand (up to period t_2) of items $1, \ldots, KA - 1$ is held in inventory; therefore, $IZ_i(t-1) \le IA_i(t-1)$ for $i = 1, \ldots, KA - 1$, where $IZ_i(t-1)$ and $IA_i(t-1)$ are the amount of inventory of item i carried at the end of period $t-1$ under schedule Z and A, respectively. In addition, according to the definition of KZ, no units of items $KZ + 1, \ldots, M$ are held in inventory at the end of period $t-1$; therefore, $IZ_i(t-1) \le IA_i(t-1)$ for all items $i = KZ + 1, \ldots, M$. Finally, if $KZ = KA$ then $IZ_{KZ}(t-1) < IA_{KA}(t-1)$ since $IZ^1(t-1) < IA^1(t-1)$ and by the optimality of Z it is preferable to hold in inventory less expensive items. Combining the last three statements with $KZ \le KA$ we obtain: $IZ_i(t-1) \le IA_i(t-1)$ for all $i = 1, \ldots, M$.

Using Lemma 1(ii) on schedule Z and the above relationships, we obtain $IZ(t) < \sum_{i=1}^{M} (IZ_i(t-1) - d_{it})^+ + C \le \sum_{i=1}^{M} (IA_i(t-1) - d_{it})^+ + C \le IA(t) + C$. Therefore, $IZ(t) < IA(t) + C$ and since $IZ(t) \bmod C = IA(t) \bmod C$, we have $IZ(t) \le IA(t)$, contradicting our assumption. □

Theorem 1 forms the basis for enumerating and evaluating all possible schedules for an arc in the network. The next two lemmas, which are based on Theorem 1, are used to enhance the entire algorithm for the MIMV problem. Lemma 5 is useful in reducing the number of arcs that must be considered in the shortest path network, while Lemma 6 is useful in reducing the number of schedules that must be evaluated for a given arc.

LEMMA 5: If Algorithm_AS on arc (t_1, t_2), $1 \le t_1 < t_2 \le T + 1$, produces a schedule A for which period t, $t_1 \le t < t_2 - 1$, is the first period after $t_1 - 1$ with $IA(t) = 0$ then arc (t_1, t_2), as well as all arcs (t_1, τ) with $\tau > t$ and $D(t_1, t_2) \bmod C = D(t_1, \tau) \bmod C$ can be eliminated from the network.

PROOF: Suppose that Algorithm_AS on arc (t_1, t_2), $1 \le t_1 < t_2 \le T + 1$ produces a schedule A for which t, $t_1 \le t < t_2 - 1$, is the first period after $t_1 - 1$ with $IA(t) = 0$. We first claim that applying Algorithm_AS on any arc (t_1, τ) with $\tau > t$ and $D(t_1, t_2) \bmod C = D(t_1, \tau) \bmod C$ will produce a schedule A' that is identical to schedule A on periods $t_1, t_1 + 1, \ldots, t$. This follows from the observation that schedule A does not ship during periods $t_1, t_1 + 1, \ldots, t$ any unit for consumption in periods later than period t, combined with the definition of the algorithm and the fact that $D(t_1, t_2) \bmod C = D(t_1, \tau) \bmod C$.

Then, by Theorem 1, any optimal schedule Z on arc (t_1, t_2) or arc (t_1, τ) as defined above, must have $IZ(t) = 0$, which means that there is no optimal solution for these arcs for which the inventory levels at the end of all periods $t_1, \ldots, t_2 - 2$ or $t_1, \ldots, \tau - 2$ respectively, are all strictly positive. □

Recall that according to Lemma 1(iii), any candidate for an optimal solution $Z(t_1), Z(t_1 + 1), \ldots, Z(t_2 - 1)$ on arc (t_1, t_2), must satisfy $Z(t) \le \lceil D(t, t+1)/C \rceil C$ for $t_1 \le t \le t_2 - 1$. We show in Lemma 6 how to obtain tighter bounds on the optimal shipping quantity in each period. Toward that, recall the following.

1. Any optimal schedule $Z(t_1), Z(t_1 + 1), \ldots, Z(t_2 - 1)$ for arc (t_1, t_2) satisfies $Z(t_1) = XB(t_1) = XA(t_1)$, and in all other periods t, $t_1 < t \le t_2 - 1$, $Z(t) \bmod C = 0$ (Lemma 4(iii) and Lemma 1(i)).
2. The ending inventory levels $IB(t_1), \ldots, IB(t_2 - 1)$ (obtained under schedule DS) are the minimum ending inventory levels that can be obtained under an optimal policy (Theorem 1).
3. The ending inventory levels $IA(t_1), \ldots, IA(t_2 - 1)$ (obtained under schedule AS) are the maximum ending inventory levels that can be obtained under an optimal policy (Theorem 1).

LEMMA 6:

(i) An upper bound $U(t+1)$ on the optimal shipping quantity in period $t + 1$ is obtained by allocating $IB(t)$ (the minimum possible inventory level at the end of period t) to items, from the least to the most expensive, where the allocation for item i is at most $D_i(t + 1, t_2)$. Given such allocation, the upper bound $U(t+1)$ on the optimal shipping quantity in period $t + 1$ is obtained by rounding up the number of vehicles required for the remaining demand in period $t + 1$.

(ii) A lower bound $L(t+1)$ on the optimal shipping quantity in period $t + 1$ is obtained by allocating $\min\{IA(t), D(t + 1, t + 2)\}$ (where $IA(t)$ is the maximum possible inventory level at the end of period t) to period $t + 1$. Given such allocation, the lower bound $L(t + 1)$ on the optimal shipping quantity in period $t + 1$ is obtained by rounding up the number of vehicles required for the remaining demand in period $t + 1$.

PROOF: Note that in (i) we are applying the mechanism of Algorithm_AS, which advances shipment as much as possible, to the minimum quantity $IB(t)$. Therefore, the demand that remains uncovered by the allocated units is the maximum possible remaining demand in period $t + 1$. Similarly, in (ii) we are applying in period $t + 1$ the mechanism of Algorithm_DS, which delays shipment as much as possible, to the maximum quantity $IA(t)$. There-

fore, the demand that remains uncovered by the allocated units is the minimum possible remaining demand in period $t + 1$. In both cases, we then ship the minimum number of full vehicles that covers the remaining demand of period $t + 1$. □

The upper and lower bounds $U(t + 1)$ and $L(t + 1)$, respectively, for $t = t_1 + 1, \ldots, t_2 - 1$, described in Lemma 6, are derived rigorously in Algorithm BOUNDS in Appendix C.

COROLLARY 3: A candidate $Z(t_1)$, $Z(t_1 + 1), \ldots, Z(t_2 - 1)$ for an optimal solution on arc (t_1, t_2) must satisfy: $L(t) \leq Z(t) \leq U(t)$ for $t_1 \leq t \leq t_2 - 1$.

Summary of the Search Algorithm to Compute an Arc Cost

To conclude this section, we suggest the following algorithm to enumerate all possible candidates for an optimal schedule on arc (t_1, t_2). Note that the costs of arcs that are not eligible by Lemmas 3 and 5 do not have to be computed.

1. Calculate schedules A and B.
2. Perform Algorithm BOUNDS to set the lower and upper bounds of the shipping quantities.
3. Generate all possible schedules that are candidates to be optimal, by considering all possible values of Z that are between A and B as stated by Theorem 1, but whose shipping quantities are in between the bounds set by algorithm BOUNDS. This step is performed similarly to a branch and bound algorithm. The variables to branch over are the number of vehicles used in each of the periods $t_1, t_1 + 1, \ldots, t_2$, which can vary within the limits set by Theorem 1 and Algorithm BOUNDS. The branch and bound structure is useful because those limits are updated whenever considering a new period, based on the values assigned to previous periods in the path of the tree. Note, however, that no lower and upper bounds for the objective value of the problem are computed at the nodes of the tree and therefore truncating the tree may be performed only as a result of applying the bounds discussed above. Finally, at each leaf of the tree, a vector of aggregate shipping quantities is generated, whose cost will be computed and compared to other candidate solutions; see the next step.
4. For each schedule generated by Step 3, calculate its cost according to the Scheduling Algorithm, and choose the schedule whose cost is minimal.

The worst case complexity bound of the above algorithm is very high, where most of the work is in generating all possible schedules (Step 3 above). While the total amount shipped in the first period is known exactly, the inventory at the end of the period that is designated for the second period may receive up to $C - 1$ values. Thus, the total amount produced in the second period, which is the smallest number of full vehicles that cover the net demand, may be one of two values. In every subsequent period, there may be up to $C - 1$ additional units of inventory and therefore one more possible vehicle in the period. Thus, in the worst case, after t periods there may be $t + 1$ possible values of total shipment in the subsequent period, which results in complexity of $O((t_2 - t_1)!) = O(T!)$ possible schedules for each subproblem. For each schedule, the Scheduling Algorithm is calculated; therefore, the complexity of each subproblem is $O(T!TM)$. Considering that there are $O(T^2)$ subproblems to solve, the resulting complexity for the entire search algorithm is $O(T!T^3M)$. In practice, the complexity is much lower, as demonstrated in our numerical study, presented in the next section.

In the next section we compare the search algorithm with the DP algorithm developed in [1]. To keep this paper as self contained as possible, we give here some basic information on the DP algorithm. We first note that similar to other dynamic lot sizing algorithms, it is based on finding a shortest path in a network that has a node representing each period and arcs that go from a lower to a higher node index. The arc cost represents the cost of a subproblem whose beginning and ending inventories are zero. Most of the work in the algorithm is to find the arc costs (and then the effort to find the shortest path is quadratic). Thus, the DP algorithm is applied to each subproblem.

The DP algorithm is based on a decomposition of each subproblem to segments of periods in which the ending inventory of the j (say) most expensive items is zero. Then it calculates gradually the cost of the segments in which j increases; that is, more items have zero ending inventory. At the end of the subproblem, all items have a zero ending inventory. In this way, the DP algorithm is different from a straightforward formulation, which would have required considering a much larger number of subproblems. The main property behind the saving in computations is the following: when inventory is held, we first hold items with the least expensive unit holding cost, unless holding more expensive items may delay the shipping of an entire batch (in which case, holding costs will be saved). The full details of the DP algorithm are given in [1].

Heuristics

Schedules B and A that serve as a basis for the enumeration procedure presented in this section are in fact good schedules that may serve as heuristics by themselves or as a

basis for a heuristic. These schedules capture two different attractive characteristics of the problem's parameters; therefore, we expect that a procedure that combines them would be very effective. We suggest and test two heuristics that are based on schedules DS and AS. *The basic heuristic* calculates the costs of schedules DS and AS for the entire problem and chooses the better of them. This is a very simple heuristic and still quite efficient as we demonstrate in the next section. The *enhanced heuristic* calculates schedules DS and AS for every subproblem and chooses the better of them to be the arc cost. Then, a shortest path procedure is used to find the best schedule for the entire problem based on the heuristically calculated arc costs. The enhanced heuristic proved to be exceptionally good, as demonstrated in the next section.

The simplicity and effectiveness of the above described heuristic procedures make them an attractive alternative to the exact and non-polynomial procedures. This is particularly true for large or difficult problem categories.

3. NUMERICAL STUDY

In this section we describe a numerical study that we performed with respect to the search algorithm of Section 2 and two other exact methods that solve the problem. We also include results regarding the two heuristic procedures mentioned at the end of Section 2. We had several goals in performing the numerical study: first, we wanted to compare the running time of the search algorithm to other existing exact solution methods and to identify characteristics of the problem parameters under which the search algorithm is particularly useful. Second, we were interested in identifying an approximate limit on problem sizes that can be solved within a reasonable amount of time. Third, we wanted to conduct a sensitivity analysis study, that is, to learn how the problem parameters are likely to affect the running time of the algorithm. Finally, we wanted to learn what is the effectiveness of the suggested heuristics. We used a Pentium III 1.2 GHz personal computer; the search algorithm was programmed in Turbo C++ and the DP algorithm was programmed in Pascal, and we used CPLEX 7.0 to solve (P1), the MILP presented in Section 1. Both programs used simple data structures and basic search strategies. However, the DP algorithm is harder to program due to the complex decomposition structure. We put a limit of 5 hours on the running time of each instance by each of the solution methods.

We conducted the numerical study with four data sets. In the first data set we used $T = 20$ and $M = 5$, which are conservative values for the parameters that are likely to have the strongest effect on the running time. The other parameters were chosen to what we call *basic values* as follows: $C = 20$, $K = 200$, $D \sim U[0, 20]$, and $h \sim U[1, 2M]$. We denote a specific set of parameter values as *problem category*. Then, an additional 11 problem categories were generated where each category is obtained by replacing one of the basic parameters by one of the following 11 parameter specifications: $C = 5, 10, 30, 40$, $D \sim U[10, 30]$, $D \sim U[3, 5]$, $D \sim U[8, 12]$, $h \sim U[1, 5]$, $K = 100, 300, 500$. Throughout our numerical study we generated and ran five independent instances that belong to the same problem category (that is, for each instance different demand and holding cost realizations were used). For data set 1 this resulted in a total of 60 runs for each of the three exact solution methods. Our main conclusion from data set 1 is that the CPLEX algorithm, applied to formulation (P1) of Section 1, was significantly slower than the search and the DP algorithms and also had a very high variability in the solution time. We note, however, that (P1) is a straightforward formulation of the problem and it is possible that with a more sophisticated formulation the performance of the CPLEX algorithm would be enhanced. This is currently work in progress and is out of the scope of this paper. From now on, we refer to the CPLEX algorithm applied to formulation (P1) as CPLEX(P1). The search and DP algorithms ran very fast and it was not possible to identify significant trends as a function of the problem category; therefore, the results are summarized in Table 1 in an aggregated format. Table 1 shows the average time (in seconds) and standard deviation (in parentheses) for each of the three solution methods. Note that CPLEX(P1) was not able to solve 3 of the 60 instances within the time limit of 5 hours.

Table 1. Average (standard deviation) in seconds for $T = 20$ and $M = 5$.

Search algorithm	0.08 (0.33)
DP algorithm	0.2 (0.68)
CPLEX(P1)	435.7 (2474.8)*

* Three instances did not finish within 5 hours.

In the second data set we investigated the effect of increasing parameters T and M on the running times of all three algorithms. We set all other parameters to their basic values as described in data set 1 and changed either T or M, as specified in Table 2. We can see from Table 2 that in all problem categories, except one, CPLEX(P1) was significantly slower than the other two solution methods. The running times of all three algorithms increased considerably when T was increased, but more significantly for CPLEX(P1). While for $T = 40$ the running time of CPLEX(P1) increased only moderately (and even outperformed the search algorithm in this category), for $T = 50$ the average solution time was 38.5 minutes for the three instances that were solved, while two other instances could not be solved within the limit of 5 hours. For $T = 40$ the

Table 2. Average (standard deviation) time in seconds for problem set 2.

	$M = 10$	$M = 20$	$M = 30$	$T = 10$	$T = 30$	$T = 40$	$T = 50$
Search	<0.01 (<0.01)	0.4 (0.5)	0.2 (0.4)	<0.01 (<0.01)	0.4 (0.5)	897.2 (1993.9)	406.2 (768.8)
DP	<0.01 (<0.01)	0.6 (0.9)	0.2 (0.4)	<0.01 (<0.01)	0.6 (0.9)	207.6 (427.2)	346.4 (600.7)
CPLEX(P1)	4.6 (4.6)	20.8 (38.7)	5 (7.3)	1.2 (0.4)	118.8 (138.6)	303.6 (319.2)	2312.3* (653.9)

* Two instances did not finish within 5 hours

solution time using the search algorithm was extremely high for one instance (4464 seconds), which increased significantly both the average and the standard deviation for this category. Indeed, all algorithms are characterized by a very high standard deviation of the solution time. The search and the DP algorithms performed quite similarly for this set of parameters, with an average solution time of 6.8 and 5.8 minutes, respectively, for $T = 50$.

We conclude from data sets 1 and 2 that the search and the DP algorithms clearly dominate CPLEX(P1), with significantly faster running times. Therefore, for the rest of our numerical study we focused on those two algorithms only. In problem set 3 we increased the T and M parameters together, to identify approximately the limit of the problem sizes that can be solved by these algorithms (all other parameters are at their basic level). The results, summarized in Table 3, indicate that problem sizes of up to $T = 40$ and $M = 40$ were solved within a few minutes. When one or more of these parameters was increased to 50, some instances could not be solved within the time limit. We note that for the categories $T = 50$ with $M = 40$ where one instance could not be solved by both algorithms, and $T = 50$ with $M = 50$ where two instances could not be solved by both algorithms, the instances that could not be solved were the same ones for both algorithms. In most problem categories (except for the largest, $T = 50$ with $M = 50$) the search algorithm was on average faster than the DP algorithm. In fact, we checked and found that this was also true for the specific solution times of the vast majority of instances. Among 41 instances that were solved by both algorithms, for 39 the solution of the search algorithm was faster than that of the DP algorithm and for 2 instances the reverse was true. One more instance was solved by the DP algorithm only, and 3 more instances were not solved by either algorithm. The standard deviation of the running times was usually proportional to the average, with a similar coefficient of variation for both algorithms. We conclude from this set of experiments that the search algorithm usually outperforms the DP algorithm, with a few exceptions.

The goal of the fourth and final data sets was to perform sensitivity analysis on the running time as problem parameters other than T and M are varied. We ran again the search and the DP algorithms; however, all trends that could be identified with one of them was also identified in the other one, and the relationship between the two algorithms remained as in problem set 3. Therefore, we omit the results of the DP algorithm and report on those obtained for the search algorithm. In this data set we use $T = 40$ and $M = 40$, the rest of the parameters were set at their basic levels, and then one parameter was varied in each problem category, as indicated in Tables 4a and 4b. For comparison purposes, we also denote the results of the basic category (this is a category that we reported on in Table 3 under the heading of $T = 40$ and $M = 40$). We divided this data set into two parts, examining the impact of the parameters C and D in data sets 4a and 4b, respectively. Overall, as indicated in Tables 4a and 4b, the running time is very sensitive to the capacity and demand problem parameters. The running time increases when the vehicle capacity increases or when the average demand per item decreases.

In Table 4a we observe that the time increases significantly when the vehicle capacity increases. This effect may be explained by the dependency of the search algorithm on

Table 3. Average (standard deviation) time in seconds for problem set 3.

	$T = 30$ $M = 10$	$T = 30$ $M = 20$	$T = 30$ $M = 30$	$T = 30$ $M = 40$	$T = 30$ $M = 50$	$T = 40$ $M = 40$	$T = 40$ $M = 50$	$T = 50$ $M = 40$	$T = 50$ $M = 50$
Search	3.6 (3.6)	4.8 (4.5)	2.6 (3.6)	4.6 (6.4)	28.8 (53.1)	91.6 (74.4)	152.8* (178.9)	258.7* (158)	6621*** (6385.2)
DP	6.8 (6.8)	11 (10.7)	64.6 (106)	33.6 (54.2)	45.2 (66)	197.8 (178.3)	233.2** (228.6)	612.7* (341.2)	6085.7*** (7569.3)

* One instance did not finish within 5 hours.
** The average and standard deviation correspond to the four instances that were solved by the search algorithm; the fifth instance was solved within 2998 seconds.
*** Two instances did not finish within 5 hours.

Table 4a. Average (standard deviation) time in seconds for problem set 4a.

	$C = 5$	$C = 10$	$C = 15$	$C = 20$	$C = 25$	$C = 30$
Search alg.	1.6	2.0	14.2	91.6	501*	—
	(0.5)	(1.4)	(21.8)	(74.4)	(784)	—

* Two instances did not finish within 5 hours.
—, None of the instances in this category were solved within 5 hours.

the closeness of the bounds $L(t)$ and $U(t)$.[1] Specifically, the algorithm performs better when these bound are close, since then the number of alternatives that must be examined decreases. Indeed, when calculating the average per period difference, given by the expression $\sum_{t=1}^{T} (U(t) - L(t))/T$, in a sample of complete instances and their subproblems with T in the range of 38–40, we found that this expression increased steadily with the increase in the capacity. The values obtained for instances with capacity levels of 15, 20, 25, and 30 were 6, 20.5, 54.3, and 71.6, respectively. We believe that this trend explains the time increase when C increases.

In Table 4b we present the sensitivity of the algorithm with respect to the demand distribution of the items. In the left four columns of the results we used an identical distribution for all items and observe generally a decrease in time when the average demand value increases. As above, this may be explained by the expression $\sum_{t=1}^{T} (U(t) - L(t))/T$, which increased (in a sample similar to the above) as the average demand decreased. The sensitivity with respect to capacity and average demand suggests that the ratio between capacity and average demand affects the gap between the upper and lower bound, which in turn affects the time of the search algorithm. In the rest of the columns of Table 4b we used different distributions for a certain percentage of the items, as indicated. Although the trend is not entirely consistent due to the high variability in the running time of different instances, the general observation is that the running time is some weighted average of the running times that are obtained with all items taken from the same distribution.

[1] We thank the Associate Editor for suggesting this explanation.

We conclude from the numerical study that the search and the DP algorithms are effective in solving problems that are quite large. If, for example, periods represent days, then a problem with 4–5 weeks and 5–6 days per week can usually be solved in less than a minute. CPLEX(P1), while capable of solving small problems, cannot be used within a reasonable amount of time for problems of higher dimension. This result is not surprising, as CPLEX is a general commercial software that is not designed particularly for this type of problem and does not exploit its special structure, especially when formulation (P1) is used. The search and DP algorithms both do exploit the structural properties of an optimal solution and are indeed much more effective in solving large problems. Throughout the numerical study the search algorithm was faster than the DP algorithm for most instances, with only a few instances where the reverse was true. We found that both algorithms are sensitive to some problem parameters (other than the obvious T and M) so the largest problem size that we can expect to solve is when both T and M are in the range of 30 to 50. Larger T and M require relatively small vehicle capacity and larger average per period demand.

It is interesting to recall that the complexities of the search and the DP algorithms are $O(T!T^3M)$ and $O(T^{M+5}M)$, respectively. It is evident that both algorithms are much faster in practice. Moreover, while the complexity expression of the DP algorithm suggests that its running

Table 4b. Average (standard deviation) time in seconds for problem set 4b.

	$D \sim U$ [3, 5]*	$D \sim U$ [8, 12]	$D \sim U$ [0, 20] (Basic)	$D \sim U$ [10, 30]	25% $D \sim U$ [3, 5] 75% $D \sim U$ [10, 30]	50% $D \sim U$ [3, 5] 50% $D \sim U$ [10, 30]	75% $D \sim U$ [3, 5] 25% $D \sim U$ [10, 30]
Search alg.	5430.0	35.4	91.6	1.2	1.6	1.8	—
	(6303.1)	(64.3)	(74.4)	(0.4)	(0.55)	(0.45)	

	25% $D \sim U$[8, 12] 75% $D \sim U$[10, 30]	50% $D \sim U$[8, 12] 50% $D \sim U$[10, 30]	75% $D \sim U$[8, 12] 25% $D \sim U$[10, 30]
Search alg.	1.6	1.4	14
	(0.55)	(0.55)	(6.78)

* Three instances did not finish within 5 hours.
— none of the instances in this category were solved within 5 hours.

Table 5. Performance of the heuristic solution.

Optimal = DS = AS	Optimal = DS ≠ AS	Optimal = AS ≠ DS	DS ≠ Optimal ≠ AS
36.6%	52.4%	2.7%	8.4%

time would increase very rapidly as M increases, the increase in practice seems more moderate. The reason is that by exploiting the properties of the problem along with the specific problem parameters, many possibilities can be excluded from considerations. We also note that the search algorithm is much easier to program than the DP algorithm.

Finally, we recall the suggested heuristics for problem sizes that are larger than the limit indicated above. The basic heuristic suggests to choose the better of the two extreme solutions that are used in the search algorithm, namely, DS and AS. To test the effectiveness of this heuristic we chose six problem categories (the basic problem and the variations of $K = 100$, $K = 300$, $K = 500$, $C = 5$, and $C = 10$, total of 30 problem instances) from data set 4 ($T = M = 40$) that included a total of 12,300 sub-problems that needed to be solved by one of the exact algorithms. Table 5 indicates the percentage of these sub-problems that were solved optimally by either the DS or the AS schedules, or both. It is surprising to find out that only 8.4% of all sub-problems were not solved to optimality by either the DS or the AS schedules, or both. As expected, the DS schedule is particularly attractive, as it delays release of vehicles as much as possible, and by that keeps relatively small amounts of inventory.

The above results suggest that the enhanced heuristic procedure in which we combine the shortest path procedure with the above heuristic solution to each of the subproblems may perform very well. We tested the enhanced heuristic on 10 instances from data set 4 ($T = M = 40$), which belong to categories $C = 20$ and 25 (5 instances each). The results were exceptionally good: in 9 of 10 instances the optimal solution was achieved, and in the last instance the optimality gap was negligible, only 0.01%. When the basic heuristic was run for the same instances, none of the 10 instances was solved optimally, and the average optimality gap was 0.3% with a maximum optimality gap of 1.57%. We conclude that both heuristics perform very well. The basic heuristic takes linear time only, while the enhanced heuristic takes $O(T^3)$ since we need to find the solution of schedules DS and AS (which takes linear time) for every subproblem (and there are $O(T^2)$ subproblems). The increase in solution time for the enhanced heuristic comes with a better performance.

4. CONCLUSIONS

In this paper we developed a new search method for the MIMV problem. Although the worst case complexity of the search method is exponential, it performs quite well on relatively large problems that we tested. We also note that we chose to implement these algorithms in quite a straightforward way, so further efficiency improvements may be possible by investing more efforts in producing a more sophisticated code. We compared the search method to two other exact solution methods for the problem, namely, the commercial CPLEX algorithm applied to formulation (P1) and the DP algorithm suggested by Anily and Tzur [1]. Neither of these algorithms for the MIMV problem was evaluated before, and it was particularly interesting to find out how the DP algorithm performs. Our conclusion from the numerical study is that CPLEX(P1) may be used for small problem sizes only, while both the search and the DP algorithms are quite fast for large problems. The search algorithm outperforms the DP algorithm in most cases. Finally, we observed that a very simple heuristic provided the optimal solution to over 90% of the sub-problems investigated. An enhanced heuristic performed extremely well with a moderate increase in computation time. Therefore, both heuristics could be useful for large or difficult problem categories.

The analysis of the quite different three solution methods provides a good overall treatment of the problem. We also recall, as was shown in [1], that the above solution methods for the MIMV problem may also be used to solve the MISV problem. As mentioned earlier, possible valid inequalities and/or reformulations to the problem may enhance the efficiency of solving the problem via general mathematical programming solvers; this is currently work in progress.

APPENDIX A

The Scheduling Algorithm

Given a vector $(X(1), \ldots, X(T))$, the scheduling algorithm uses the following procedure in order to determine the quantities $X_i(t)$ for $1 \le i \le M$ and $1 \le t \le T$: start at the last period and allocate to it $X(T)$ units by giving priority to the most expensive items that are demanded in this period (i.e., start with the most expensive item and allocate to it its demand, continue with the next expensive item, down to the least expensive item). If $X(T) < D(T, T + 1)$ then shift the excessive demand in period T, i.e., $D(T, T + 1) - X(T)$ least expensive units, to period $T - 1$ and repeat the process backward to period 1.

The algorithm below also performs a test, checking whether any part of Lemma 1(i)–(iv) is violated; if any part is violated then the given vector is not a candidate to be an optimal schedule for periods $1, \ldots, T$. Finally, the algorithm performs a feasibility test, checking whether there exists a period up to which the cumulative aggregate demand exceeds the cumulative aggregate capacity.

The Scheduling Algorithm

Input. $X(t)$ $1 \le t \le T$

Output. An infeasibility message, or the best-detailed schedule and its cost or a message that the schedule cannot be optimal.

Feasibility Test. (A feasible schedule exists if and only if $\sum_{\tau=1}^{t} X(\tau) \geq D(1, t+1)$ for $1 \leq t \leq T-1$ and $\sum_{t=1}^{T} X(t) = D(1, T+1)$.)
$t=1$
while $t \leq T-1$ do begin
  if $\sum_{\tau=1}^{t} X(\tau) < D(1,t+1)$ then goto (s) (no feasible solution exist)
  $t=t+1$
endwhile
If $\sum_{\tau=1}^{T} X(\tau) \neq D(1,T+1)$ then goto (s) (no feasible solution exist)
  Construction of the best detailed schedule.
Begin (initialization step)
$I(0)=0$
for $t=1,\ldots, T$
  $I(t)=0$
  for $i=1,\ldots,M$
    $d'_{it}=d_{it}$
    $I_i(t)=0$
  endfor;
endfor;
$V=K\sum_{t=1}^{T}\lceil X(t)/C\rceil$ (current cost consists of dispatching costs only)
$t \leftarrow T$
(b) $W \leftarrow X(t)$
  $F=0$ (F counts the number of units shipped in each period for future periods)
  $i \leftarrow M$
(a) $X_i(t) \leftarrow \min\{d'_{it}, W\}$
  If $X_i(t) < W$ then begin
    $W \leftarrow W - X_i(t)$
    $i \leftarrow i-1$
    goto (a)
  endif
  otherwise do begin
    if $t>1$ do begin
      $d'_{i(t-1)} = d'_{i(t-1)} + d'_{it} - W$ (shift the remaining demand of i
      $I_i(t-1) = d'_{it} - W$ from period t to period $t-1$)
      $F=F+I_i(t)-\max\{0, I_i(t-1)-d_{it}\}$
      $I(t-1) = I(t-1) + I_i(t-1)$
      $V=V+h_iI_i(t-1)$ (add holding cost of i from $t-1$ to t to V)
      if $i>1$ do begin
        for $k=1,\ldots,i-1$ do begin
          $I_k(t-1)=d'_{kt}$
          $d'_{k(t-1)} = d'_{k(t-1)} + d'_{kt}$
          $F=F+I_k(t)-\max\{0, I_k(t-1)-d_{kt}\}$
          $I(t-1)=I(t-1)+I_k(t-1)$
          $V=V+h_kI_k(t-1)$ (add to V the holding cost of item k from $t-1$ to t)
        endfor;
      endif;
    endif;
  endotherwise
  if $t=1$ then $F=I(1)$;
  if [$I(t-1)>0$ and $X(t)\mathrm{mod}C > 0$] or $F \geq C$ or $I_M(t) \geq C$ or ($F \geq X(t)\mathrm{mod}C$ and $X(t)\mathrm{mod}C > 0$) or [$I(t-1)=0$ and $\sum_{i=1}^{M} d_{it}\mathrm{mod}C = 0$ and $I(t)>0$] then do begin
    "*it is not an optimal solution*"
    stop.
  endif;
  $t \leftarrow t-1$
  if $t=0$ stop (the best allocation is found)
  goto (b)
(s) "*no feasible solution exist*"
  endAlgorithm

APPENDIX B

Algorithm_DS

Input. Two periods $t_1 < t_2$ for which $D(t_1, t_2)\mathrm{mod}\ C \geq D(t_1, t_1+1)\mathrm{mod}\ C$ or $D(t_1, t_2)\mathrm{mod}\ C = 0$.

Output. The total production quantities $XB(t_1), \ldots, XB(t_2-1)$ and the corresponding ending inventories at periods $t_1, \ldots, t_2-1$, that is: $IB(t_1), \ldots, IB(t_2-1)$ for DS assuming $IB(t_1-1) = IB(t_2-1) = 0$.

begin
  $XB(t_1) = \min\{z: z \geq D(t_1,t_1+1)$ and $z\mathrm{mod}C = D(t_1,t_2)\mathrm{mod}C\}$
  $IB(t_1) = XB(t_1) - D(t_1,t_1+1)$
  $t=t_1+1$
    while $t \leq t_2-1$ then do begin
      if $D(t_1, t+1) \leq \sum_{\tau=t_1}^{t-1} XB(\tau)$ then $XB(t)=0$
      otherwise $XB(t) = \left\lceil \dfrac{D(t_1,t+1) - \sum_{\tau=t_1}^{t-1} XB(\tau)}{C} \right\rceil C$
      $IB(t) = \sum_{\tau=t_1}^{t} XB(\tau) - D(t_1, t+1)$
      $t=t+1$
    endwhile;
end.

The complexity of this procedure is $O(t_2 - t_1)$.

Algorithm_AS

Input. Two periods $t_1 < t2$ for which $D(t_1, t_2)\mathrm{mod}\ C \geq D(t_1, t_1+1)\mathrm{mod}\ C$ or $D(t_1, t_2)\mathrm{mod}\ C = 0$.

Output. The total production quantities $XA(t_1), \ldots, XA(t_2-1)$and the corresponding ending inventories at periods $t_1-1, \ldots, t_2-1$, that is: $IA(t_1), \ldots, IA(t_2-1)$ for AS assuming $IA(t_1-1) = IA(t_2-1) = 0$.

begin
$t=t_1$
$IA(t_1-1)=0$
$XA(t) = \min\{z: z \geq D(t,t+1)$ and $z\mathrm{mod}C = D(t,t_2)\mathrm{mod}C\}$
while $t \leq t_2-1$
  $W=IA(t-1)+XA(t)-D(t,t+1)$
  $IA(t)=IA(t-1)+XA(t)-D(t,t+1)$
  for $i=1, \ldots, M$ do begin
    while $W>0$ do begin
      $IA'_i(t)=(IA_i(t-1)-D_i(t,t+1))^+$
      $IA_i(t)=IA'_i(t)+\min\{W, D_i(t+1,t_2)-IA'_i(t)\}$
      $W=W-(IA_i(t)-IA'_i(t))$
    endwhile
  endfor;
  $t=t+1$
  $XA(t)=\lceil \sum_{i=1}^{M} (D_i(t,t+1)-IA_i(t-1))^+/C\rceil C$
endwhile;

The complexity of this procedure is $O(M(t_2 - t_1))$.

APPENDIX C

Algorithm BOUNDS

Input. Two periods $t_1 < t_2$ for which $D(t_1, t_2)\mathrm{mod}\ C \geq D(t_1, t_1+1)\mathrm{mod}\ C$ or $D(t_1, t_2)\mathrm{mod}\ C = 0$. Schedule DS for the arc, i.e., $XB(t_1), \ldots, XB(t_2-1)$ and the corresponding $IB(t_1), \ldots, IB(t_2-1)$ and schedule AS for the arc, i.e., $XA(t_1), \ldots, XA(t_2-1)$ and the corresponding $IA(t_1), \ldots, IA(t_2-1)$.

Output. Upper bounds and lower bounds on the production quantity in each period t, $t_1 \leq t < t_2$ for any schedule that is a candidate for an

optimal solution. We denote the upper bounds by $M(t_1), \ldots, M(t_2 - 1)$, and lower bounds by $L(t_1), \ldots, L(t_2 - 1)$

begin
$t=t_1$
$M(t) = XB(t)$
$L(t) = XA(t)$
while $t<t_2-1$
$W = IB(t)$
for $i=1, \ldots, M$ do begin
  $I_i(t)=\min\{W ; D_i(t+1,t_2)\}$
  $W=W-I_i(t)$
endfor;
$t=t+1$
$M(t) = \lceil \sum_{i=1}^{M} (D_i(t,t+1)-I_i(t-1))^+/C \rceil C$
$L(t) = \lceil (D(t,t+1)-IA(t-1))^+/C \rceil C$
endwhile;

ACKNOWLEDGMENTS

The authors are grateful to Lawrence Wolsey for helpful discussions on the problem. They express their sincere appreciation to the programming support of Eyal Pecht and Ran Etgar. The research was supported by the Israel Institute of Business Research (IIBR), Faculty of Management, Tel Aviv University.

REFERENCES

[1] S. Anily and M. Tzur, Shipping multiple-items by capacitated vehicles—An optimal dynamic programming approach, Transport Sci 39(2) (2005), 233–248.

[2] A. Federgruen, J. Meissner, and M. Tzur, Progressive interval heuristics for multi-item capacitated lot-sizing problem, Oper Res, to appear (2002).

[3] M. Florian and M. Klein, Deterministic production planning with concave costs and capacity constraints, Manage Sci 18 (1971), 12–20.

[4] M. Florian, J.K. Lenstra, and A.H.G. Rinnooy Kan, Deterministic production planning: Algorithms and complexity, Manage Sci 26 (1980), 669–679.

[5] P. Kaminsky and D. Simchi-Levi, Production and distribution lot sizing in a two stage supply chain, IIE Trans 35(12) (2003), 1065–1075.

[6] C.Y. Lee, A solution to the multiple set-up problem with dynamic demand, IIE Trans 21(3) (1989), 266–270.

[7] S. Lippman, Optimal inventory policy with multiple set-up costs, Manage Sci 16 (1969), 118–138.

[8] Y. Pochet and L.A. Wolsey, Lot-sizing with constant batches: Formulation and valid inequalities, Math Oper Res 18 (1993), 767–785.

[9] K. Pryor, C.C. White, and R. Kapuscinski, Multi-item inventory policies with capacitated delivery vehicles and deterministic demand, Working Paper, University of Michigan, 2000.

[10] S. Van Hoesel, H.E. Romeijn, D.R. Morales, and A.P.M. Wagelmans, Integrated lot-sizing in serial supply chains with production capacities, Manage Sci 51(11) (2005), 1706–1719.

[11] C.A. Yano and A.M. Newman, Scheduling trains and containers with due dates and dynamic arrivals, Transport Sci 35(2) (2001), 181–191.