A cyclic policy for the loading of multiple products on a vehicle with different compartment sizes

JOSEPH BUKCHIN1 and SUBHASH C. SARIN2

1Department of Industrial Engineering, Faculty of Engineering, Tel Aviv University, Tel Aviv 69978, Israel
E-mail: bukchin@eng.tau.ac.il
2Grado Department of Industrial and Systems Engineering, Virginia Tech, 250 Durham Hall, Blacksburg, VA 24061, USA
E-mail: sarins@vt.edu

Received August 2001 and accepted August 2003

In this paper, we address a multi-product loading problem in which a vehicle (a truck or a ship) is used to transfer multiple products. The product demands are different but stationary over time. The vehicle consists of compartments of different sizes and each compartment can contain only one product type during each shipment. No shortages are permitted, and we assume that the inventory holding cost is significantly lower than the delivery cost. The objective is to minimize the setup rate, that is, the number of deliveries per time unit. A cyclic policy is shown to be optimal, and a heuristic algorithm is developed to determine the cycle length as well as the assignments of products to the compartments during each of the requisite number of shipments made during that cycle. A comparison of the solutions obtained by the proposed algorithm with the optimal solutions (or a bound) indicate that the algorithm provides solutions with optimal setup rates in most of the problem instances considered and, when not optimal, the setup rates of these solutions are close to optimal values.

1. Introduction: problem definition, a motivational example and literature review

In this paper, we address a multi-product loading problem in which a vehicle (a truck or a ship) transfers multiple products from a supplier to a customer in order to satisfy product demands. The vehicle can make at most one delivery in a unit of time, which can be a day, week or an appropriate fixed time duration. We designate this unit of time as a period. The demands of the products are different but stationary over periods. The vehicle is divided into several fixed sized compartments, and each compartment can contain only one product type during a delivery. The mixing of products in a compartment is not allowed, as it may ruin individual products, or in the case of chemical products may even cause a hazardous reaction. We assume that enough storage space is available at the customer location to accommodate the amount delivered. Also, enough quantity of each product is assumed to be available at the supplier so that any amount specified by a delivery plan can be loaded on to the vehicle. In addition, no shortages are allowed. However, the inventory holding costs are assumed to be insignificant in comparison to the transportation cost. Consequently, the objective is to determine the best loading policy that satisfies product demands in every period by using the minimum number of deliveries per period, also called the setup rate. Note that the maximum achievable setup rate is one, in which case a delivery occurs in every period.

To illustrate the above scenario, consider an example in which two products (A and B) are to be transported in a vehicle containing two compartments. Both of these products have, say, an identical demand per period of one unit. For the sake of simplicity and without loss of generality, we assume identical volumes of different products and, consequently, each compartment can contain an identical number of these products. Suppose one compartment can contain only one unit of a product while the other can contain three units. Figure 1 presents a loading policy for this example. In this figure, each arc represents an assignment of a product type to a compartment during a delivery (product A is associated with the upper arcs and product B with the lower arcs). The origin node of each arc denotes the period in which that delivery is made and the length of an arc represents the number of periods over which the demand of the corresponding product is satisfied by the assignment. Note that two deliveries are performed in four periods. In the first delivery, performed in period 1, both compartments are fully filled, compartment 1 with one unit of product A and compartment 2 with three units of product B. In the second delivery, the loading is reversed, that is, compartment 1 is filled with one unit of product B while compartment 2 with three units of product A. The demands for periods 3 and 4 are automatically satisfied by these two deliveries. If
Fig. 1. Delivery schedule for the example problem.

this policy is repeated every four periods, the average delivery rate, in accordance with the resultant cyclic policy, will be 0.5, namely, an average of one delivery every two periods. Our aim is to determine an optimal cyclic policy which specifies the number of deliveries in a cycle as well as the assignments of products to the compartments in each delivery so that the setup rate (number of deliveries (setups) per period) is minimized.

An instance of the transportation of non-intermixable products is encountered most frequently in the oil industry for distribution of petroleum products. The distribution costs from refineries to gas stations are estimated, in the US, to be close to three billion dollars (Ronen, 1995). These products are transported by tanker trucks or by sea tankers having multiple compartments. The mixing of different types of petroleum is not allowed in the same compartment as it may result in a different and undesirable product type. Other examples include transportation of chemical products, food related products, or products associated with the process industry.

The transportation costs involved in the above examples are significantly higher than the inventory holding costs, especially when the mode of transportation involved is ships, specialized trucks and the like. Another justification for addressing only the delivery cost is associated with the nature of the market for these product types. Typically, excess inventories of these products are held at the source regardless of the short-term loading policy. Consequently, the minimization of the holding cost at the destination will not reduce the total holding cost, since the inventory will be held either at the destination or at the source. This argument, of course, assumes similar holding costs at the source and destination. Although we do not explicitly consider inventory costs, we believe that the inventory levels resulting from the proposed approach at the destination will remain relatively low due to the nature of the solution approach. We elaborate on this in Section 4.1.

Although the vehicle routing and scheduling problems have been covered extensively in the literature, there is little reported on the problem on-hand in view of the loading of non-intermixable products in its compartments. Fagerholt and Christiansen (2000) combine the ship scheduling and allocation problems and propose a set-partitioning-based approach. They assume that each ship has a cargo hold which is flexible and can be adjusted to give different sized sub-compartments. Only a single product is assumed for loading in each of these compartments during a delivery. Time windows are associated with the loading and unloading times of products, and there is a penalty cost for violating the time windows. A modification of this model, which allows soft time windows due to the introduction of a penalty cost that is varied as a function of the deviation from the targeted time window, is presented in Fagerholt (2001). Another related problem, that has been widely addressed in the literature, is the Segregated Storage Problem (SSP) presented in Shifler and Naor (1961). In this problem, different types of grains are to be stored in the compartments of a silo such that there is at most one product in each compartment. It is assumed that product demands, storage capacities i.e., the sizes of the compartments in the silo, and the variable storage costs are known. Also, an external storage facility of infinite capacity is assumed to be available to accommodate overflow. The objective is to find an assignment of products to compartments such that the total cost, due to holding in the external storage, is minimized. Improvements in the solution procedures for SSP have been proposed by White and Francis (1971), Dannenbring and Khumawala (1973), Evans and Cullen (1977), Neebe (1987) and Evans and Tsubakitani (1993). The SSP is a special case of our problem as, in the SSP, an assignment is sought only for a single period. However, a solution for the SSP problem which does not require any external storage is equivalent to a feasible solution to our problem.

Yuceer (1997) addresses the problem of allocating products to a vehicle having compartments of different sizes such that at most one product is assigned to a compartment. The objective is to maximize the minimum number of periods over which the demands of all the products are satisfied by a single delivery. Consequently, the solution of this problem gives the time interval between consecutive (and identical) deliveries. If we apply the policy suggested in Yuceer (1997) to the example presented above, only one
period's worth of demands of products A and B can be satisfied with a single delivery. The same policy is repeated every period. We call this policy a static policy. Contrast this with the policy in Fig. 1, which we will term as dynamic. The time interval between the successive deliveries of the policy of Yuceer (1997) is thus one period. Due to the size of the smaller compartment, only one-third of the second compartment is utilized. By filling more items in this compartment, one does not increase the number of periods for which the demand is satisfied by this policy; only more items are shipped than needed in each period. Yuceer (1999) presents an extension of this model which includes both a fixed delivery cost and an inventory holding cost. Unlike the single assignment model of Yuceer (1997, 1999), we allow a multiple-assignment solution in which these assignments can be different. Hence, we explore cyclic solutions where, in each cycle, a series of different assignments of products to compartments are performed in order to satisfy the demand over the maximal number of periods. This will always generate at least as good of a solution, in terms of the setup rate, as that obtained by Yuceer (1997, 1999) because our cyclic approach subsumes that of Yuceer’s. In the example problem presented above, the setup rate for Yuceer’s (1997) solution is one while that for our cyclic policy is 0.50.

The remainder of the paper is organized as follows. The problem of minimizing the delivery/setup rate for a given cycle time along with its model formulation are presented in Section 2. We develop some useful properties of the optimal solution and show its convergence to a lower bound with increment in cycle length. An experimental investigation demonstrates that a close to optimal solution can be obtained using a relatively short cycle length. In Section 3, we consider the inverse problem of determining the maximum number of periods over which the demand of all the products can be satisfied, without incurring shortages, for a given number of setups. This problem is shown to be more efficient to solve than the original problem. In Section 4, we present a solution approach for solving the inverse problem, based on the principles of Toyota’s goal-chasing approach presented by Monden (1998). A detailed experimentation is conducted to evaluate the performance of this procedure. This is presented in Section 5. The solutions obtained by the proposed procedure are compared with the optimal solutions (for moderately sized problems) and with an upper bound (for large sized problems) and are found to be close to optimal, when not optimal.

2. Minimizing the setup rate for a given cycle length

Although, an infinite planning horizon is addressed in this problem, we seek a cyclic solution, namely, a delivery plan for a finite number of periods which is to be repeated infinitely. Each cycle of this plan will include several deliveries with possibly different assignments of products to the compartments of the vehicle in each. The amounts of products shipped, in these deliveries, should be enough to satisfy their demands over the entire cycle length, without causing any shortages. We believe that a practical loading policy should be based on a relatively short cycle length. This is motivated by the continuous advancements in technology typically leading to the development of new products with short product lives. If a relatively large cycle length is determined, only a part of the cyclic policy may actually be implemented, a fact that will probably impair the quality of the solution. In some practical situations, the cycle length may be given in advance due to some global company policy (i.e., determination of the delivery schedule for a quarter, etc.). Hence, it is important to examine the impact of cycle length on solution quality. Indeed, we discuss this issue later, and show that the setup rate of a cyclic solution approaches a lower bound value with an increment in cycle length. Moreover, experiments indicate that, in most cases, we can obtain a close to optimal solution value by using cycles of relatively small lengths.

Due to the assumption that the cycle length is finite and relatively short, and in order to simplify the problem, we first address the problem of determining a delivery schedule for a given cycle length of \( T \) periods. In case the cycle length is not known a-priori, the current problem should be solved for several values of \( T \) until the desired solution is obtained.

2.1. The model formulation for minimizing setup rate for a given \( T \)

**Definition 1.** The setup rate is the average number of setups required per period in a cyclic solution.

Note that the setup rate is always smaller than or equal to one (the value of one is obtained when a setup occurs in every period). Next, we present a model formulation for the problem of minimizing the setup rate for a given cycle length of \( T \) periods. With given \( T \), the objective of minimizing the setup rate is equivalent to minimizing the total number of setups during the cycle.

We first define the notation.

**Parameters**

\[ d_j \] = the demand rate for product \( j, j = 1, \ldots, n \);
\[ q_i \] = the capacity of compartment \( i, i = 1, \ldots, m \).

Since the initial inventory is assumed to be zero and no shortages are permitted \( m \geq n \). In addition, for feasibility, we assume \( \sum_{i=1}^{n} q_i \geq \sum_{j=1}^{m} d_j \), that is, a single delivery should be able to satisfy the total demand of at least a single period.

**Decision variables**

\[ x_{ijt} = \begin{cases} 1, & \text{if product } j \text{ is assigned to compartment } i \text{ in period } t; \ i = 1, \ldots, n, \ j = 1, \ldots, m, \\ 0, & \text{otherwise.} \end{cases} \]

\[ s_t = \begin{cases} 1, & \text{if a setup is incurred in period } t; \ t = 1, \ldots, T, \\ 0, & \text{otherwise.} \end{cases} \]
The problem can, then, be formulated as follows:

$$\min \sum_{t=1}^{T} s_t$$

subject to:

$$\sum_{i=1}^{m} x_{ijt} \leq 1, \quad i = 1, \ldots, n, \quad t = 1, \ldots, T,$$

$$s_t \geq \frac{\sum_{i=1}^{m} \sum_{j=1}^{m} x_{ijt}}{m}, \quad i = 1, \ldots, T,$$

$$\sum_{k=1}^{m} \sum_{i=1}^{m} x_{ijk} q_i \geq t(d_i), \quad t = 1, \ldots, T, \quad j = 1, \ldots, m,$$

$$x_{ijt}, s_t = (0, 1).$$

The number of variables is $T(nm + 1)$, and the number of constraints is $T(m + n + 1)$.

The constraint set (2) ensures that not more than one product is assigned to a single compartment during a setup. The setup variable, $s_t$, is set to one in period $t$, if at least one product type is delivered in that period. This is captured in constraint (3). The capacity constraints (4) guarantee that no shortage occurs in any period for any product. The binary nature of the variables $x_{ijk}$ and $s_t$ is dictated by constraint (5), while the objective of minimizing the number of setups, for a given cycle length, $T$ is represented by Equation (1).

Regarding the computational complexity of the problem, it is easy to show that the two-partition problem can be reduced to this problem. Consider $T = 1, n = 2$, $(m \geq n)$, $d_1 = d_2 = (\sum_{i=1}^{m} q_i)/2$ and the number of setups is equal to one. Then, clearly, a solution to the problem on hand exists only if the two-partition problem has a solution, i.e., the $m$ compartments can be divided into two disjoint sets $S_1$ and $S_2$, such that $\sum_{i \in S_1} q_i = d_1 (=d_2)$ and $\sum_{i \in S_2} q_i = d_2 (=d_1)$. Thus, the problem on hand is NP-hard.

2.2. Some useful properties

2.2.1. Properties of the optimal solution

**Proposition 1.** Let $D(= \sum_{i=1}^{m} d_i)$ denote the total demand per period and $Q(= \sum_{i=1}^{m} q_i)$ the total vehicle capacity. Then, the value of $D/Q$ constitutes a lower bound of the setup rate.

**Proof.** Assume that the problem is relaxed in the sense that each compartment is no longer limited to accommodate one product type, and in fact can accommodate any mixture of product types. Then, the ratio between the total demand per period, $D$, and the total vehicle capacity, $Q$, gives the minimum possible number of setups required to satisfy the demand per period, namely, the setup rate.

**Proposition 2.** Suppose $S$ setups are performed with full compartments each time. If the leftover inventory of all the products is equal to zero at the end of some common period $T$, then the allocation of products to compartments in these $S$ setups is optimal.

**Proof.** Since all compartments’ capacity is utilized in each setup, and the inventory level of all the products in some common period $T$ is equal to zero, then it follows that there is an equality between the total supply and total demand (as shortages are not permitted). That is, $S \sum_{i=1}^{m} q_i = T \sum_{j=1}^{m} d_i$. Consequently, the setup rate:

$$\frac{S}{T} = \frac{\sum_{i=1}^{m} d_i}{\sum_{i=1}^{m} q_i} = \frac{D}{Q},$$

which is a lower bound of the setup rate (by Proposition 1).

**Corollary 1.** A lower bound on the setup rate for a given cycle length, $T$, can be given by: $[TD/Q]/T$ (where $[x]$ represents the smallest integer greater than or equal to $x$).

**Proof.** This lower bound is a consequence of Proposition 2, where the numerator represents the smallest (integral) number of setups required to satisfy the product demands over $T$ periods.

The smallest of such $T$’s for which the starting and ending inventory levels are zeros, gives an optimal cycle length. When the optimal cycle length is relatively large, one may still want to use a cyclic solution of shorter duration even though the inventory of all the products is not zero at the end of that cycle. To that end, the following two questions arise: (i), how does the setup rate value of a cyclic solution relate to its lower bound, given by Proposition 1; and (ii), what is a reasonable cycle length that can yield an almost optimal solution? In the following proposition we answer the first question by showing that a cyclic solution converges to the lower bound, given by Proposition 1, with an increment in cycle length. The second question, in fact, alludes to the rate of convergence of the setup rate with cycle length. We show, experimentally, that this rate of convergence is rather fast and, indeed, close to optimal solutions are obtained by using cycles of short lengths.

**Proposition 3.** The setup rate value of a cyclic solution converges to the lower bound, $D/Q$, as the cycle length approaches infinity.

**Proof.** Without loss of generality, assume that the demands, $d_1, \ldots, d_m$, and the compartments’ sizes, $q_1, \ldots, q_n$, are integers (because, if not, they can be converted into integers by multiplying their values by the greatest common multiple). Also, assume that the shortages are allowed, and no additional cost is associated with it. In this case, an optimal delivery plan, $f_s$, can easily be implemented as follows: the first $d_1$ setups of the vehicle are only filled with product type 1. The next $d_2$ setups of the vehicle are only filled with product 2, and so on. After $D = \sum_{i=1}^{m} d_i$ periods, the demands of all product types have been satisfied exactly for $Q = \sum_{i=1}^{m} q_i$ periods. Hence, no more deliveries need to be
A multi-product loading problem

performed in periods \( D + 1, \ldots, Q \). The solution is optimal (according to Proposition 2), and the cycle that should be repeated is of \( Q \) periods.

If \( s_{jk} \) denotes the shortage of product type \( j \) in period \( k \), let,

\[
\mathbf{s} = \{s_1, \ldots, s_m|s_j = \max_{k=1,D} s_{jk}\}.
\]

be the vector of the maximum shortages that occur for each product during the first \( D \) periods (no shortages can occur in periods \( D + 1, \ldots, Q \)). Since the vehicle capacity is larger than the total demand per period (\( Q > D \), otherwise, in an equality situation, a delivery should be made in every period), we can create a delivery plan, \( f_w \), for a finite number of periods, \( w \geq 1 \), for which no shortages occur during delivery and the inventory level at the end of period \( w \) is equal to \( i = \{i_1, \ldots, i_m\} \), where \( i_j \geq s_j \) for all \( j \). Then, we have a feasible solution for the non-shortage situation in which the \( f_w \) policy is applied during the first \( w \) periods, and the \( f_s \) policy is implemented every \( Q \) periods afterwards. The inventory level after every \( Q \) periods will then be equal to \( i \).

Let \( S_w \) be the number of setups performed in the first \( w \) periods. The number of setups associated with each additional \( Q \) periods is \( D \), and the setup rate for \( w + lQ \) periods is equal to:

\[
\frac{S_w + lD}{w + lQ},
\]

where \( l \) represents the number of \( Q \) period cycles. Then, we can see that:

\[
\lim_{\ell \to \infty} \frac{S_w + lD}{lQ + w} = \frac{D}{Q},
\]

namely, as the number of cycles approaches infinity, the setup rate approaches the lower bound, \( D/Q \).

2.2.2. Experimentation to study the quality of the optimal solution as a function of the cycle length

Next, we experimentally examine the quality of solutions obtained with variation in the cycle length in order to determine if close to optimal solutions can be obtained in shorter cycle lengths. To that end, 12, 12-compartment problems were randomly generated, and were optimally solved via the CPLEX solver by varying the given cycle length from one to 27 periods. Four problems were solved for three products, another four for five products and the rest for eight products. The compartment sizes and the demands for products were randomly generated such that, in half of the problems, the demand per period was relatively close to the total capacity (lower bound between 0.7 and 0.95, tight problems), and for the other half, the capacity was much larger than the demand per period (lower bound between 0.3 and 0.5, loose problems). The sample results for two such problems, one tight and the other loose, are shown in Table 1.

The first column of Table 1 presents the cycle length, \( T \). The next four columns are associated with a loose problem (lower bound = 0.461) and the last four columns with a tight problem (lower bound = 0.817). The optimal number of setups obtained for a loose problem is depicted in the second column (and sixth column for the tight problem) of Table 1 while the corresponding optimal setup rate is shown in the third column (and seventh column for the tight problem). Each problem is solved for 27 different cycle lengths. The lower bound, \( D/Q \), of the loose problem (see Proposition 1) is presented in column four (eight for the tight problem) of Table 1. Note that this lower bound value is independent of cycle length. The percentage differences, between the optimal setup rates and the lower bound for each case, are included in the fifth column (ninth for the tight problem).

For each of the above loose and tight problems, a graph was plotted in order to show the nature of the closeness of the value of the optimal solution to lower bound with variation in cycle length. It is interesting to see that two different structures characterize the nature of this relationship, depicted in Fig. 2, (a and b) depending on whether the problem is loose or tight. Figure 2(a) is associated with a loose problem, while Fig. 2(b) is associated with a tight situation. Note that in both graphs, the optimal setup rate

![Fig. 2. Setup rate as a function of the cycle length: (a) for the loose problem; and (b) for the tight problem.](image-url)
<table>
<thead>
<tr>
<th>Cycle length (number of periods)</th>
<th>Loose problem</th>
<th>Tight problem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Cycle length (number of periods)</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Lower bound (LB)</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Closeness to LB</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 1. Results of a tight and a loose problem to illustrate the closeness of the value of optimal solution to the lower bound with variation in the cycle length.
of maximizing the cycle length over which the demands of all the products can be simultaneously satisfied by a given number of setups. As no inventory cost is assumed, these setups can be implemented in successive periods starting from the first period of a cycle. The optimal setup rate can be obtained more efficiently by using this inverse problem than by the original problem, discussed in Section 2. We show this next.

**Observation 1.** The inverse problem involves a reduced solution space for determining the cycle length (in the case where the cycle time is not given as an external parameter).

We demonstrate the underlying idea behind observation 1 through an example. This example involves six compartments and three products. This problem was optimally solved for the cycle lengths varying from one to 40 periods. The setup rate obtained is plotted in Fig. 3. Recall that the setup rate cannot be larger than one since at most one setup can be performed in each period. The horizontal line in Fig. 3 is the lower bound, $D/Q$, of the setup rate for this problem. Note that three setups are required to satisfy the product demands for five periods (setup rate 3/5 = 0.6). Hence, the solution for the cycle length of four periods is dominated by that for the cycle length of five periods. In other words, the optimal solution for the inverse problem will directly result in the cycle length of five periods (skipping $T = 4$) while in the original problem we would have also determined an optimal solution for the cycle length of four periods. This observation is valid for all cases in which the same number of setups are required to satisfy product demands over several cycle lengths. In all such cases, only the maximal cycle length, corresponding to a certain number of setups, needs to be considered. Hence, in relation to the original problem, the inverse problem considers only a subset of possible cycle lengths. For the illustrative example on-hand, the best inverse problem solutions are shown in Fig. 3 as circled points for various numbers of setups. For instance, 18 setups give the best setup rate for a cycle length of 35 periods. The number of inverse problems to be solved in order to get this solution, in a sequential search, is then 18 (circled points in Fig. 3), instead of 35 (all points on the graph) that would have been needed if the original approach were applied.
4. Solution approach

4.1. Description of the algorithm

The proposed heuristic aims to solve the inverse problem, namely, the problem of maximizing the cycle length over which the demands of all the products are simultaneously satisfied by a given number of setups, |S|. In addition, since this model does not consider the inventory holding cost, we assume that all setups are performed sequentially in the first S periods of the cycle.

Proposition 4. An upper bound on the number of periods over which the demands of all the products are satisfied by S setups is given by \( \lfloor \frac{SQ}{D} \rfloor \), where \( \lfloor x \rfloor \) represents the largest integer smaller than or equal to x.

**Proof.** Using the same relaxation as performed in Proposition 1, we can see that the above expression represents the integral number of periods satisfied by the total capacity, SQ.

The proposed heuristic algorithm aims to divide the total capacity, SQ, in proportion to the demand of the products per period. If the capacity can be exactly divided in accordance with the demand proportions, and all the compartments are fully utilized, then one can claim that the demands of all the products will be satisfied for exactly the same number of periods, being equal to the upper bound given in Proposition 3. Still, such a solution is not necessarily feasible. The feasibility of the solution is associated with satisfying the demand on time, namely avoiding shortages in every period. Since we assume that all setups are performed sequentially starting at period 1 and ending in period S, and since the vehicle capacity is greater than the demand per period (otherwise, there is no feasible solution to the problem), the cycle length over which the demands are satisfied by these setups is at least S. Then, we have to ensure that no shortages occur in periods 1 to S (no shortages can occur in the rest of the periods until the end of the cycle since the demands of these periods are satisfied by the leftover inventory of period S).

Let us assume that an assignment of product \( j \) to compartment i in period \( t \), \( x_{ijt} \), for \( t = 1, \ldots, (l-1) \) (that is, for setups 1, \ldots, \( l-1, l < S \)) is given, and we are about to determine the assignment of the \( l \)th setup.

**Definition 2.** The effective demand of product \( j \) in period \( l \), \( d'_jl \), that is, the number of units of product \( j \) that the vehicle must carry during the \( l \)th setup, is given as follows:

\[
d'_jl = \max \left\{ d_j - \left( \sum_{t=1}^{l-1} \sum_{i=1}^{m} x_{ijt} q - (l-1)d_j \right), 0 \right\}.
\]

This is essentially the difference between the demand per period of product \( j \) and the leftover inventory of product \( j \) from the previous \( l-1 \) periods if larger than zero, and zero otherwise. A negative value of this difference implies availability of sufficient inventory from previous periods to cover demand of product \( j \) in period \( l \).

**Definition 3.** The slack at the start of period \( l \), \( SK_l \), is defined as the difference between the total capacity per period and the sum of the effective demands of all product types in period \( l \), namely, \( SK_l = Q - \sum_{j=1}^{n} d'_jl \).

**Proposition 5.** Assume that a partial assignment of setup \( l \) is given that follows the \( l-1 \) previous setups. Let \( p_{jl} \) be the number of units of product \( j \) assigned in setup \( l \). If \( \sum_{j=1}^{n} \max\{0, (p_{jl} - d'_jl)\} > SK_l \), then no feasible solution can be obtained.

**Proof.** Clearly follows by the definition of feasibility.

The left-hand side of the inequality in Proposition 5 represents the extra amount of units supplied in period \( l \) that
A multi-product loading problem

are not needed for this period, and hence, it will satisfy the demands of the products in later periods. The slack represents the difference between the total effective demand and the capacity of period \( l \), namely, the maximal number of units that can be kept for satisfying demand in later periods. In case the condition in Proposition 5 holds, the number of units required for later periods is larger than the slack, thereby implying infeasibility of the current assignment.

The proposed heuristic incorporates sequential assignments of products to the compartments along with the implementation of a feasibility check following every assignment. The product assignment utilizes principles of the Toyota goal-chasing method, presented in Monden (1998). This method was developed to determine product sequences in assembly lines. The objective of this approach is to maintain a stable withdrawal rate of the components from previous production stages. We use the principle of this approach in order to keep the proportions of the assigned products as close as possible to the proportions of their demands. After each assignment, the feasibility check is performed. In the case where a partial solution cannot yield a feasible solution, the last assignment is canceled, the procedure backtracks to a previous assignment, and a different assignment is performed.

4.1.1. The heuristic algorithm CPL (Cyclic Product Loading)

Let \( S \) be the given number of setups to be performed. Hence, there are \( mS \) compartments to be assigned with a total capacity of \( SQ \). The total capacity should be divided among the products in such a way that the capacity associated with each product, \( j \), is as close as possible to:

\[
SQ \left( \frac{d_j}{\sum_{l=1}^{n} d_l} \right).
\]

To that end, the following steps are followed.

Initialization. Rank the compartments in the descending order of their sizes and assign the products to the compartments in this order at every setup (i.e., first finish assigning products at the first setup, then at the second and so on.). Let \([K]\) denote the current assignment number, and set \([K] = 0\).

Let \( K_{\text{max}} = [mS] \) denote the total number of assignments (since \( m \) assignments are determined in each setup), and \( l = [\lfloor K \rfloor/m] \), be the current setup number, where \( \lfloor x \rfloor \) represents the smallest integer larger than \( x \). Let \( P^c \) be the complete set of all products, and \( P = P^c \) at start) be a set of current candidates to be assigned.

Step 1. (Assignment of products to compartments.) Set \([K] = [K] + 1\). In the \([K] \)th assignment of this step, assign product \( J_{[K]}^* (j_{[K]}^* \in P) \) to compartment \([K]\)

(according to the order determined in Step 1) such that:

\[
J_{[K]}^* = \arg \min_{j \in \phi} \left[ \sum_{j=1}^{n} \frac{L_{j}^{[K]} q_{[K]} - d_j}{\sum_{l=1}^{n} d_l} \right]^2,
\]

where \( q_{[K]} \) denotes the capacity of the compartment filled in the \([k] \)th assignment in order, and \( L_{j}^{[K]} \) equals one when product \( j \) is assigned to the \([k] \)th compartment, and zero otherwise.

Step 2. (Feasibility check of the assignments.) If \( \sum_{j=1}^{n} \max\{0, (p_j - d_j')\} \leq SK_i \), set \( P = P^c \), and select the next compartment in order and go to Step 4. If \( \sum_{j=1}^{n} \max\{0, (p_j - d_j')\} > SK_i \), cancel the assignment of the last product to the current compartment, \( J_{[K]}^* \), and set \( P = P - J_{[K]}^* \). If \( P \neq \phi \), go to Step 2. Otherwise, \( (P \) is an empty set, \( [K] = [K] - 1 \), set \( P = P^c - J_{[K]}^* \) and go to Step 2.

Step 3. (Termination check.) If \( [K] = K_{\text{max}} \), products at all setups have been assigned, stop. Otherwise, go to Step 2.

Since the total capacity per period, \( Q \), is larger than the total demand per period, \( D \), we can expect that the infeasibility problem will be less likely to occur with an increase in the number of periods. In addition, when a feasible solution has been obtained in the first period, this solution can be repeated in the next period as well. Hence, the feasibility issue is mainly associated with the first period.

Note that, although the proposed approach does not explicitly consider inventory costs, it still has the tendency to minimize the inventory at the destination. This follows from the fact that high inventory levels are most likely the result of an unbalanced loading policy, namely shipping quantities much higher than the demands for some products (e.g., loading a vehicle with only one product type). Since, the vehicle capacity is constrained, such a policy will result in a higher setup rate. The proposed loading policy, on the other hand, aims at dividing the total capacity in proportional to the product demands, and hence, the products’ inventory levels held at the destination will stay relatively balanced and will not result in extreme values.

4.1.2. Modification of algorithm CPL

A major factor that may affect the run time of algorithm CPL is the search for a feasible solution. The feasibility problem is mainly relevant in the first period, since in this period, \( l = 1 \), the value of \( SK_i = Q - \sum_{j=1}^{n} d_j \) is minimal, as each effective demand, \( d_j' \), is equal to its maximal values, \( d_j \). In the second period, some leftover inventory from the first period decreases the effective demand of some product(s), and consequently increases the value of \( SK_i \), and so on in the subsequent periods. Hence, the difficulty of finding a feasible solution decreases as we move away from
the first period. Accordingly, an extension was added to algorithm CPL as follows. In the case where no solution is obtained by the algorithm within a prespecified amount of time, the one-setup problem is solved by the CPLEX solver, and the solution is fed into the algorithm. The algorithm CPL is then continued from the second period onwards. In the case where no feasible solution for the second period is obtained again within the prespecified time limitation, the first period solution is duplicated for the second period as well, and the algorithm proceeds from the third period onward. However, in our experimentation, no case was observed in which a duplication of the first period solution was required, namely, a feasible solution was relatively easy to find from the second period onwards. It is important to note that this extension is aimed at medium sized problems. For small problems, it is likely that this procedure is not needed. On the other hand, for a large problem, the solution by CPLEX of even a single setup problem can involve extensive computational requirements.

5. Computational results

5.1. Small to medium sized problems

A mixed level, full factorial design was performed to examine the performance of the proposed algorithm. The purpose of using factorial design is twofold; (i) first, to ensure that the problem set generated is a representative sample, and it characterizes a wide range of problems, and (ii) to identify significant effects of the problem parameters on the efficiency of the procedure.

The data of the compartment sizes and product demands was generated randomly and five factors were selected with two and three levels. The detailed design is summarized in Table 3.

The problems involved two vehicle capacities, namely, 10 and 15 compartments, and different values for the number of products. The ratio between the number of products and the number of compartments, called the product tightness, was set at three levels, namely at approximately, 0.3, 0.5 and 0.8. Accordingly, the number of products for the 10-compartment problem are three, five and eight and those for the 15 compartment problem are four, seven and ten.

Different variability levels of the compartment sizes and product demands were also examined (designed as high and low). Another factor that was considered is the ratio of the total demand to the total capacity which is called the capacity tightness. Two levels were considered, namely, tight and loose. A tight capacity level indicates that the total demand is close to the total capacity whereas a loose capacity level indicates a relative excess capacity compared to the demand. The fifth factor was the number of setups, and three levels were considered namely, three, five and seven. The last factor in the experimentation addressed the problem size, expressed by the number of compartments. By taking all the levels of the five factors into consideration, we observe that the total number of possible combinations of these factors is $2^2 	imes 2 	imes 3 = 144$ i.e., a total of 144 data combinations for the full factorial design.

The dependant variable (or the response) is the closeness of the solution obtained to optimality. All solutions were first compared to their upper bounds, and when there was a difference between the two, the problem was optimally solved using the CPLEX solver.

The effects of the problem parameters on the solution, examined using ANOVA, and presented in Table 4, indicate that four out of the six main effects are significant (for a $p$-value smaller than 0.05). Algorithm CPL performs better for a larger variance in the compartments’ sizes, probably because of the nature of the algorithm that assigns products to the compartments in a descending order of their sizes. Varied compartment sizes help in the better utilization of this principle. The variability in demand also has a small but significant negative effect on the performances. The product tightness seems to have the most significant effect on the performance. As the number of products increases (tight problem), the heuristic performance deteriorates. This may be due to the fact that the demand of more products has to be satisfied by the same number of compartments. The last significant effect is the capacity tightness. Results show that the heuristic performs better in solving tight problems. This follows from the fact that, for these problems, the optimal

### Table 3. The experimental design

<table>
<thead>
<tr>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Products tightness</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>Variance of compartment size</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Variance of product demand</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Capacity tightness</td>
<td>Tight</td>
<td>Loose</td>
</tr>
<tr>
<td>Number of setups</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Problem size</td>
<td>10 compartments</td>
<td>15 compartments</td>
</tr>
</tbody>
</table>

### Table 4. ANOVA results regarding the effects of the problem parameters on the performance of the algorithm

<table>
<thead>
<tr>
<th>Source</th>
<th>Sum of squares</th>
<th>DF</th>
<th>Mean square</th>
<th>F value</th>
<th>Prob &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>8.583</td>
<td>8</td>
<td>1.073</td>
<td>7.470</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>Variance of compartment sizes</td>
<td>1.000</td>
<td>1</td>
<td>1.000</td>
<td>6.963</td>
<td>0.0093</td>
</tr>
<tr>
<td>Variance of product demands</td>
<td>0.694</td>
<td>1</td>
<td>0.694</td>
<td>4.835</td>
<td>0.0296</td>
</tr>
<tr>
<td>Product tightness</td>
<td>4.181</td>
<td>2</td>
<td>2.090</td>
<td>14.554</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>Capacity tightness</td>
<td>2.250</td>
<td>1</td>
<td>2.250</td>
<td>15.666</td>
<td>0.0001</td>
</tr>
<tr>
<td>Number of setups</td>
<td>0.431</td>
<td>2</td>
<td>0.215</td>
<td>1.499</td>
<td>0.2271</td>
</tr>
<tr>
<td>Problem size</td>
<td>0.028</td>
<td>1</td>
<td>0.028</td>
<td>0.193</td>
<td>0.6608</td>
</tr>
<tr>
<td>Residual</td>
<td>19.389</td>
<td>135</td>
<td>0.144</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cor total</td>
<td>27.972</td>
<td>143</td>
<td>0.144</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
solution tends to be closer to the trivial solution of performing a setup in each period. Both the number of setups and problem size appear to have no significant effect on the solution quality.

The optimal solution of all the problems but one, namely of 143 problems, were found to be equal to their corresponding upper bounds on the cycle length. Since the upper bound, ⌊SQ/D⌋ is a rounded integral part of SQ/D, it seems that the difference between the two terms represents the excess inventory in the case where the optimal solution satisfies an integral value of periods. Then, one can expect that this value should affect the performance of the heuristic in the sense that it is easier to find an optimal solution where that difference is relatively large. This conjecture was examined by performing a regression model between the above value and the difference between the heuristic and the optimal solution. Results support this conjecture and show that the model is significant with a t-value and the difference between the heuristic and the optimal solution tends to be closer to the optimal solution in about 78% of the problems solved, while for over 99% of the problems, algorithm CPL obtains solutions that are within one period away from the optimal solution. Table 5 also demonstrates that the quality of the solutions obtained by algorithm CPL is consistent over different problem sizes.

Another aspect of algorithm CPL is the time it takes to solve the above problems. The run time of the proposed procedure is mainly affected by the check for feasibility. If a feasible solution is found easily, then it takes just a few seconds to solve any reasonably sized problem. In these experiments, the problems can be classified into three sets. The first set includes the problems that were solved almost immediately. About 90% of the problems belong to this set. The second set contains the problems in which a feasible solution was difficult to obtain, but still were solved within 5 minutes of CPU time. Only six such problems were observed (about 4.1%). The third set, contains problems that could not be solved within 5 minutes of CPU time. These problems were eventually solved by the modified algorithm, in which a CPLEX solver was used to find a feasible solution for the first period. The results, then, were fed into the regular procedure that provided the solution. Nine such problems required this step (see the number in the parenthesis in Table 5), which demonstrate the need for the modified procedure.

### 5.2. Large problems

In the last set of experiments, large sized problems were solved and the values of the solutions obtained were compared with the corresponding upper bounds. The purpose of this experimentation was to examine the performance of algorithm CPL in solving problems, which cannot be solved by an optimal procedure such as CPLEX. Forty, 60-compartment problems were generated. Results are presented in Table 6. As shown in the second column of Table 6, 20 problems involve 20 product types and the remaining problems involve 40 product types (indicating a different product tightness). The number of setups was randomly generated from a discrete, uniform distribution, and varied between three and eight (see the third column in Table 6). Two values, one low and one high, were set for the ratio between the total capacity and the total demand per period to indicate different capacity tightnesses. The lower value was set around two and the higher value around five. The values of the compartment sizes and the demands were randomly generated from a uniform distribution while keeping the coefficient of variation relatively stable.

The fifth and sixth columns of Table 6 depict the average difference between the total capacity and the total demand per period to indicate different capacity tightnesses. The lower value was set around two and the higher value around five. The values of the compartment sizes and the demands were randomly generated from a uniform distribution while keeping the coefficient of variation relatively stable.

The fifth and sixth columns of Table 6 depict the upper bound on the cycle length and the cycle lengths, obtained by algorithm CPL, respectively. The differences between the upper bound and solution values are shown in Column 7. In 20 out of the 40 problems, the solution values, obtained by algorithm CPL, are equal to the upper bound values. The solution values of the other 15 problems are one period below the upper bound, three problems with two periods below the upper bound and two problems with three periods below the upper bound.

The percentage differences of the solution values obtained by algorithm CPL and the upper bound are shown in the last column of Table 6. The average difference is 3.59%
Table 6. Results of the experiments performed on large problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Number of products</th>
<th>Number of setups</th>
<th>Cap/Dem ratio</th>
<th>UB on cycle lengths</th>
<th>Cycle length obtained by alg. CPL</th>
<th>Gap = Col 6–Col 5 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>5</td>
<td>2.31</td>
<td>11</td>
<td>11</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>4</td>
<td>2.03</td>
<td>8</td>
<td>7</td>
<td>12.5</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>5</td>
<td>2.04</td>
<td>10</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>3</td>
<td>2.12</td>
<td>6</td>
<td>6</td>
<td>0.0</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>4</td>
<td>2.12</td>
<td>8</td>
<td>8</td>
<td>0.0</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>5</td>
<td>2.00</td>
<td>10</td>
<td>9</td>
<td>10.0</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>4</td>
<td>2.07</td>
<td>8</td>
<td>8</td>
<td>0.0</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>3</td>
<td>2.13</td>
<td>6</td>
<td>6</td>
<td>0.0</td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>5</td>
<td>2.03</td>
<td>10</td>
<td>9</td>
<td>10.0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>4</td>
<td>2.04</td>
<td>8</td>
<td>8</td>
<td>0.0</td>
</tr>
<tr>
<td>11</td>
<td>20</td>
<td>7</td>
<td>5.10</td>
<td>35</td>
<td>35</td>
<td>0.0</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>4</td>
<td>5.13</td>
<td>20</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>13</td>
<td>20</td>
<td>7</td>
<td>5.42</td>
<td>37</td>
<td>37</td>
<td>0.0</td>
</tr>
<tr>
<td>14</td>
<td>20</td>
<td>4</td>
<td>5.28</td>
<td>21</td>
<td>20</td>
<td>1.0</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>4</td>
<td>5.16</td>
<td>20</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>16</td>
<td>20</td>
<td>8</td>
<td>5.05</td>
<td>40</td>
<td>39</td>
<td>1.0</td>
</tr>
<tr>
<td>17</td>
<td>20</td>
<td>4</td>
<td>5.32</td>
<td>21</td>
<td>20</td>
<td>1.0</td>
</tr>
<tr>
<td>18</td>
<td>20</td>
<td>5</td>
<td>5.19</td>
<td>25</td>
<td>25</td>
<td>0.0</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>8</td>
<td>5.28</td>
<td>42</td>
<td>41</td>
<td>1.0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>5</td>
<td>4.99</td>
<td>24</td>
<td>24</td>
<td>0.0</td>
</tr>
<tr>
<td>21</td>
<td>40</td>
<td>6</td>
<td>2.21</td>
<td>13</td>
<td>12</td>
<td>1.0</td>
</tr>
<tr>
<td>22</td>
<td>40</td>
<td>3</td>
<td>2.02</td>
<td>6</td>
<td>5</td>
<td>1.0</td>
</tr>
<tr>
<td>23</td>
<td>40</td>
<td>3</td>
<td>2.17</td>
<td>6</td>
<td>6</td>
<td>0.0</td>
</tr>
<tr>
<td>24</td>
<td>40</td>
<td>7</td>
<td>2.14</td>
<td>15</td>
<td>14</td>
<td>1.0</td>
</tr>
<tr>
<td>25</td>
<td>40</td>
<td>6</td>
<td>2.25</td>
<td>13</td>
<td>13</td>
<td>0.0</td>
</tr>
<tr>
<td>26</td>
<td>40</td>
<td>6</td>
<td>2.10</td>
<td>12</td>
<td>11</td>
<td>1.0</td>
</tr>
<tr>
<td>27</td>
<td>40</td>
<td>5</td>
<td>2.04</td>
<td>10</td>
<td>9</td>
<td>1.0</td>
</tr>
<tr>
<td>28</td>
<td>40</td>
<td>5</td>
<td>2.14</td>
<td>10</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>29</td>
<td>40</td>
<td>3</td>
<td>2.32</td>
<td>6</td>
<td>6</td>
<td>0.0</td>
</tr>
<tr>
<td>30</td>
<td>40</td>
<td>7</td>
<td>2.07</td>
<td>14</td>
<td>14</td>
<td>0.0</td>
</tr>
<tr>
<td>31</td>
<td>40</td>
<td>4</td>
<td>5.24</td>
<td>20</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>32</td>
<td>40</td>
<td>4</td>
<td>4.99</td>
<td>19</td>
<td>19</td>
<td>0.0</td>
</tr>
<tr>
<td>33</td>
<td>40</td>
<td>4</td>
<td>5.17</td>
<td>20</td>
<td>18</td>
<td>2.0</td>
</tr>
<tr>
<td>34</td>
<td>40</td>
<td>7</td>
<td>5.31</td>
<td>37</td>
<td>34</td>
<td>3.0</td>
</tr>
<tr>
<td>35</td>
<td>40</td>
<td>7</td>
<td>5.10</td>
<td>35</td>
<td>33</td>
<td>2.0</td>
</tr>
<tr>
<td>36</td>
<td>40</td>
<td>8</td>
<td>5.15</td>
<td>41</td>
<td>38</td>
<td>3.0</td>
</tr>
<tr>
<td>37</td>
<td>40</td>
<td>7</td>
<td>5.21</td>
<td>36</td>
<td>35</td>
<td>1.0</td>
</tr>
<tr>
<td>38</td>
<td>40</td>
<td>7</td>
<td>4.64</td>
<td>32</td>
<td>30</td>
<td>2.0</td>
</tr>
<tr>
<td>39</td>
<td>40</td>
<td>5</td>
<td>5.14</td>
<td>25</td>
<td>24</td>
<td>1.0</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>6</td>
<td>5.05</td>
<td>30</td>
<td>29</td>
<td>1.0</td>
</tr>
</tbody>
</table>

while the maximal value is 16.67%. Since the upper bound value itself can be loose, algorithm CPL provides solutions that are close to optimum.

6. Summary and conclusions

In this paper, the multi-product loading of a vehicle with several compartments was addressed. Each compartment can contain only one product type in each delivery. The demand is assumed to be stationary and no shortage is allowed. The inventory costs are assumed to be relatively smaller than the delivery cost, and consequently, the objective is to find a loading policy which minimizes the number of deliveries per time unit, or in other words, the setup rate. Cyclic delivery schedules are considered where each cycle may contain multiple setups with different assignments of products to compartments in different setups. An Integer Linear Programming (ILP) model formulation of this problem is presented and some inherent structural properties of the problem are identified. In view of these properties, an inverse problem was defined to determine the maximum number of periods over which the product demands can be satisfied by a given number of setups.
An algorithm was developed to solve the inverse problem. Two sets of experiments were performed to test the effectiveness of this algorithm. In the first set, small to medium sized problems were solved and the setup rates obtained were compared with optimal solutions obtained by solving the ILP model directly via the CPLEX solver. The purpose of this set of experiments was two-fold; (i) to identify significant effects of the problem parameters on the performance of the algorithm; and (ii) to examine the quality of the solutions obtained by the proposed algorithm. Results show that the proposed algorithm performs the best for a large variability in the compartment sizes, and for a relatively small number of products. Still, an optimal solution was obtained for about 78% of solved problems, while for more than 99% of the problems, the algorithm provided solution values with a number of periods for a given number of setups that are within one period away from the optimal value. In the second set of experiments, large-scale problems were solved, and the solution values were compared with the upper bound values of the cycle lengths. For at least half of the problems, an optimal solution was obtained (being equal to the upper bound), while the average difference between the solution values obtained and the upper bound was found to be 3.59%.

Acknowledgement

The authors would like to thank the reviewers and the associate editor for many helpful comments during the review of this paper. Their comments definitely helped in improving the exposition of the paper.

References


Biographies

Joseph Bukchin is a faculty member of the Department of Industrial Engineering at Tel Aviv University. He received his B.Sc., M.Sc. and D.Sc. degrees in Industrial Engineering from the Technion Israel Institute of Technology. He is a member of the IIE and INFORMS. He was a Visiting Professor at the Grado Department of Industrial and Systems Engineering at Virginia Tech. His main research interests are in the areas of assembly and manufacturing systems design, assembly line balancing, facility design, design of cellular manufacturing systems, operational scheduling as well as work station design with respect to cognitive and physical aspects of the human operator.

Subhash C. Sarin is the Paul T. Norton Endowed Professor in the Grado Department of Industrial and Systems Engineering at Virginia Tech. His areas of specialty are production scheduling, applied mathematical programming and the design and analysis of manufacturing systems. He has published many papers in these areas. He is the recipient of several prestigious awards at the university, state, and national levels. He has served as an associate editor of several journals. He is a Fellow of the Institute of Industrial Engineers, and a full member of the Institute for Operations Research and Management Sciences.

*Contributed by the Location and Transportation Modeling Department*