EFFICIENT, EFFECTIVE LOT-SIZING FOR MULTI-
PRODUCT MULTI-STAGE PRODUCTION/DISTRIBUTION
SYSTEMS WITH CORRELATED DEMANDS

by

Robin Roundy

This research was supported by NSF grants MEA-8121429, ECS-8404641 and
DMC-8451984, and also by AT&T Information Systems.
EFFICIENT, EFFECTIVE LOT-SIZING FOR MULTI-PRODUCT MULTI-STAGE PRODUCTION/DISTRIBUTION SYSTEMS WITH CORRELATED DEMANDS

by Robin Roundy

1. INTRODUCTION

Consider a multi-stage, multi-product production/distribution system in discrete time. Both in-process and finished goods are referred to as components, and inventories of the same item held at different locations are considered to be different components.

When an order is placed for a component it is instantly delivered, and the required amounts of the components consumed in producing the given component are simultaneously withdrawn from their respective inventories. Nonzero lead times for production and/or transportation can usually be incorporated using the techniques outlined in [Ro84].

Deterministic external demand can occur for any or all of the components in the system. The external demands for the components are time-varying according to a common seasonal pattern. Specifically, we assume that the external demand for component \( k \) in time period \( t \) is \( v_k \delta_t \) where \( v_k \) is a volume parameter measuring the average external demand for component \( k \) and \( \delta_t \) is the seasonal factor in time period \( t \) which is common to all components. In particular, these conditions are met if there is only one component with positive external demand, as in the single-product assembly system [CWH72, CW73, CWW73, SS75]. All external demand must be met without shortages or backlogging.
Setup costs are associated with families of components. A family is any subset of the set of components, and a component can belong to any number of families. The nonnegative setup cost associated with a family is incurred whenever any one of the components in the family is ordered. For example, at a facility in a distribution network a family might consist of the set of products ordered from a certain supplier. In a manufacturing setting a family might consist of the set of components that use a certain tool.

In addition to setup costs there is a linear holding cost for each of the components. The goal is to schedule orders over a given finite horizon so as to minimize the total cost of holding and setups.

We propose a new heuristic for this problem. It is called the cluster heuristic. The effectiveness of a policy is the ratio of the cost of an optimal policy to the cost of the policy in question. The worst-case (resp., average) effectiveness of a heuristic is the effectiveness of the policy generated by the heuristic, minimized (resp., averaged) over the data. We will show that the worst-case effectiveness of the cluster heuristic is between 50% and 69%. Computational tests in Section 6 indicate that the average effectiveness of the cluster heuristic
effectiveness is very insensitive to the size of the system and to other input parameters.

Review and Discussion of the Literature.

The most general subclass of this problem that has been solved in polynomial time is the simple series system [Lo72]. For joint replenishment systems [Ve69], single-item assembly systems [CW73], and single-item one-warehouse, multi-retailer distribution systems [Ka70], the most efficient known algorithms have exponential running times.

In an effort to make the problem more tractable Veinott devised nested policies [Ve69]. In a nested policy, whenever an order is placed for component $k$ a simultaneous order is placed for all components that consume component $k$. Veinott developed an $O(NT^2)$ algorithm for finding an optimal nested policy in a single-item distribution system.

A number of authors have proposed sequential lot sizing heuristics that compute lot sizes one stage at a time [Mc76, MW76, BGH77, BM82]. Many of these heuristics use adjusted setup and holding costs that take into account the effect of a decision made at one stage on the other stages in the system. Graves proposed a multi-pass sequential heuristic [Gr81].

In empirical tests the best sequential heuristic and Graves' heuristic both had an average effectiveness of over 99%, but the effectiveness deteriorated rapidly as the number of levels in the
system increased from two to five. This indicates that for the large manufacturing systems encountered in practice the effectiveness of these heuristics would probably be much worse. For Graves' heuristic the number of Wagner-Whitin computations required would also be a serious concern if large problems were to be solved.

The constant-demand, infinite-horizon version of this problem has received considerable attention in the literature [TS70, CWH72, JK72, CW72, Sc73, SS75, Sz75, GS77, Wi81, Wi83, MM83, JMM83, Ro83, Ro84, MR85]. Recently an efficient heuristic has been developed that has a worst-case effectiveness of 98% [Ro84], and a similar heuristic has a worst-case effectiveness of 94% relative an optimal nested policy [MM83].

These heuristics for the constant-demand version use clusters, groups of components that order simultaneously because their setup costs and holding costs make simultaneous ordering more economical. This phenomenon has been observed in lot sizing for systems with non-constant demand as well [Gr81, BM82].

In the constant-demand models, if we rescale the external the demand rates of all of the components in the system by a common factor, the clusters do not change. They depend only on the relative magnitudes of the external demands for the components, and on the costs. For systems with correlated demands the relative magnitudes of the external demands for the components are
constant over time, even though the external demand for each individual component is not constant over time. Therefore we would expect the clusters to be invariant throughout the time horizon. The heuristic that we propose is based on this observation.

We show in Appendix A that even in the constant-demand case, it is not optimal to force all components in a cluster to order together. However in the constant-demand case there is always a policy that has this property and is within 2% of optimal [Ro84]. The sensitivity of earlier heuristics to the number of stages in the system and the insensitivity of the cluster heuristic to the number of stages is probably due to the fact that in the cluster heuristic "optimal" clusters are computed for the entire system, whereas in other heuristics the heuristics that are used to create them are to a greater or lesser extent system myopic.

In the next section we introduce a network representation of the system and the way costs are incurred. In Section 3 we present the cluster heuristic. Lower bounds of the cost of an optimal nested policy and on the cost of any policy are presented in Section 4. A worst-case analysis of the cluster heuristic is presented in section 5, and computational tests are found in Section 6.
2. THE COST NETWORK.

It is helpful to use networks to describe the structure of the system, the relationships between different inventories, and the way setup costs are incurred. We call these networks cost networks. Cost networks that are used to compute nested policies are called nested networks and cost networks used to compute nonnested policies are called nonnested networks. Nested networks and nonnested networks are generated differently, but for computational purposes they are treated in the same way.

In this section we first show how nested networks are generated. We then discuss some of the important properties of cost networks, including how policies are defined relative to a cost network and how the cost of such a policy is computed. Finally we show how nonnested networks are generated.

Generating Nested Networks.

The nested network has an inventory node for each component and a setup node for each multi-product family. Inventories are held only at inventory nodes. The purpose of setup nodes is to represent joint setup costs.

For convenience we assume that the index of the component or family are equal to the indices of the corresponding nodes of the nested network. Associated with each node \( n \) is a setup cost \( K_n \), a conventional holding cost \( h_n \), and a volume parameter
\( v_n \). If \( n \) is a component then the setup cost, conventional holding cost, and volume parameter of inventory node \( n \) are equal to those of component \( n \). If \( n \) is a multi-product family then the setup cost of setup node \( n \) is equal to that of family \( n \), and the conventional holding cost and volume parameter of setup node \( n \) are zero.

If component \( m \) is consumed in producing component \( n \) then the nested network has a flow arc \( m \rightarrow n \) from node \( m \) to node \( n \). The gozinto parameter \( \lambda_{mn} \) for the flow arc \( m \rightarrow n \) is the amount of component \( m \) that is consumed in producing a unit of component \( n \). If \( n \) is a multi-product family containing component \( m \) then the nested network has a setup arc \( m \rightarrow n \) from node \( m \) to node \( n \). All setup arcs have gozinto parameter \( \lambda_{mn} \) equal to zero.

Policies.

A policy for a cost network consists of a list of the times at which orders are placed at each of the nodes in the network, an order quantity for each of the orders, and a nonnegative inventory at each node at the end of each time period. If \( n \) is a setup node, the order quantities and inventories at node \( n \) are all zero.

Policies for a nested network correspond to policies for the system in an obvious way. Inventories at inventory nodes correspond to inventories of components. An order placed at node
n in the nested network corresponds to an order placed for component n if n is a component, and to a point in time when the setup cost for family n is incurred if n is a family.

A policy is said to be nested on \( m \rightarrow n \) if an order is placed at node n simultaneously with each order placed at node m. Nestedness on setup arcs is necessary for a policy to be feasible because it implies that the setup costs of all families that contain a certain component are incurred whenever the component is ordered. Nestedness on flow arcs is equivalent to the well-known concept of a nested policy.

There is always an optimal policy for which the following properties hold.

**Property 1.** The inventory at each node at the end of the last time period is zero.

**Property 2.** If an order is placed at node n in time period t then the inventory at node n at the end of time period \( t - 1 \) is zero.

It is clear that there is always an optimal policy for which Property 1 holds. Property 2 was proven by Zangwill [Za66] and Veinott [Ve69].

Recall that the external demand at node n is \( v_n \delta_t \). Let \( I^t_n \) be the inventory at node n at the end of time period t. If an order is placed at node n at time t we let \( Q^t_n \) be the
order quantity, and we set \( Q_n^t = 0 \) otherwise. The balance of mass constraint, which is necessary for feasibility, can be written as

\[
i_n^{t-1} + Q_n^t = \nu_n \delta_n^t + I_n^t + \sum_{n \rightarrow q} \lambda_{nq} Q_q^t.
\] (1)

Note that only one of the quantities on the left-hand side of (1) can be nonzero, depending on whether or not an order is placed for component \( n \) in time period \( t \). Given this fact and the times at which orders are placed, (1) can be used to sequentially determine both the order quantities and the inventories. One starts with a node that has no successors and with the last time period, and recursively uses \( I_n^t \) and \( \sum_{n \rightarrow q} Q_q^t \) to determine \( Q_n^t \) and \( I_n^{t-1} \) for all \( n \) and for all \( t \).

**Measuring Echelon Inventories.**

It is convenient to use a nonstandard unit of measure for our echelon inventories. The unit of measure we will use for the echelon inventory at \( n \) is \( V_n \), the total amount of component \( n \) required to make \( V_m \) units of all final products \( m \) in the system. Clearly

\[
V_m \equiv V_m + \sum_{m \rightarrow n} \lambda_{mn} V_n.
\] (2)
Let $E_n^t$ be the echelon inventory at node $n$ at the end of time period $t$. The echelon inventories can be calculated recursively from the conventional inventories $I_n^t$ by the formula

$$V_m E_m^t = I_m^t + \sum_{m \rightarrow n} \lambda_{mn} V_n^t E_n^t.$$  

(3)

The echelon holding cost at node $n$ is $H_n \equiv (h_n - \sum_{m \rightarrow n} \lambda_{mn} h_m) V_n$ where $h_n$ is the conventional holding cost at node $n$. Multiplying (3) by $h_m$, summing, and rearranging we have

$$\sum_n h_n I_n^t = \sum_n H_n E_n^t.$$ 

The total cost of a policy can then be written as

$$\sum_n [J_n K_n + \sum_t H_n E_n^t]$$  

(4)

where $J_n$ is the total number of orders placed at node $n$.

Using (1) and (3), it is easily shown by induction that

$$E_{n}^{t-1} - E_{n}^{t} = \delta_{t} - Q_{n}/V_{n}.$$ 

In particular, if no order is placed at node $n$ at time $t$ then
\[ E_{n}^{t-1} - E_{n}^{t} = \delta_{t} \]  

This important property is the reason for our choice of units for \( E_{n}^{t} \).

Suppose that a policy is nested on all arcs of the nested network. Using Property 2, (3), and induction on \( n \) it is easily shown that

\[ E_{n}^{t}Q_{n}^{t-1} = 0 \text{ for all } n, t. \]  

Thus in a nested policy, orders are placed at a node only when the echelon inventory there is zero.

**Concepts Used in Working with Non-Nested Schedules.**

We now show how to generate a nonnested network. With a nonnested policy the size of the echelon inventory of component \( k \) depends on the routes that the units of component \( k \) will follow as they move through the system. For this reason we will find it helpful to distinguish inventories by routes rather than by components.

To this end, we define a **finished product** to be a component that has positive external demand, i.e., a component with a positive volume parameter. Note that a finished product can also be consumed in making other components in the system. A route is
a sequence of components \( n = <k_1, k_2, \ldots, k_i> \) such that component \( k_{i-1} \) is consumed in producing component \( k_i \) for all \( i \), and \( k_i \) is a finished product. Each route \( n \) that contains more than one component has a unique sub-route formed by removing the first component in \( n \) therefrom.

We say that a unit of component \( k_1 \) follows a route \( n = <k_1, k_2, \ldots, k_i> \) if it is processed into higher-level components in the sequence \( k_1, k_2, \ldots, k_i \) and is used to satisfy the external demand for the finished product \( k_i \). Let \( R_k \) be the set of all routes whose first component is \( k \), including the route \( <k> \) if \( k \) is a finished product.

We assign a unit of the inventory of component \( k \) to the inventory of route \( n \in R_k \) if it will follow route \( n \). In this way each unit of the inventory of component \( k \) is assigned to a unique route \( n \) that starts at \( k \). Note that when an item leaves the inventory at route \( n \) it enters the inventory at the sub-route of \( n \) if \( n \) has more than one component, and otherwise it leaves the system.

**Non-Nested Networks**

The non-nested network has an inventory node for each route and a setup node for each family. As before, we assume that the index of the route or family is equal to that of the corresponding node. If \( n \) is a family then the setup cost of node \( n \) is equal to that of family \( n \) and the conventional holding cost of node
n is equal to zero. If \( n \) is a route then the conventional holding cost at node \( n \in R_k \) is equal to that of component \( k \) and the setup cost is zero.

External demand occurs only at routes that consist of a single component. Consequently if a route \( n \) contains only one component \( k \) then the volume parameter \( v_n \) at node \( n \) is equal to that of component \( k \). If \( n \) is either a multi-product route or a family then the volume parameter \( v_n \) of node \( n \) is equal to zero.

If route \( n \) is the sub-route of route \( m \) then there is a flow arc \( m \to n \) in a nonnested network from node \( m \) to node \( n \). The gozinto parameter \( \lambda_{mn} \) of the flow arc is the amount of the first component in route \( m \) that is consumed in producing a unit of the first component in route \( n \). If \( n \) is a family containing component \( k \) and \( m \in R_k \) is a route that starts with component \( k \) then there is a setup arc \( m \to n \) from node \( m \) to node \( n \) in the nonnested network. The gozinto parameter \( \lambda_{mn} \) of the setup arc is zero. Note that a nonnested network is circuitless, as all arcs lead from routes to shorter routes or from routes to families.

Policies, Nestedness on Arcs, and Costs in Nonnested Networks.

Policies for a nonnested network determine policies for the system in a natural way. If \( n \in R_k \) is a route then an order at inventory node \( n \) in the nonnested network corresponds to an
order placed for component \( k \). The amount of the order at node \( n \) is the number of units ordered that will follow route \( n \). If \( n \) is a family then an order placed at node \( n \) in the nonnested network corresponds to a point in time when the setup cost for family \( n \) is incurred.

As before, nestedness on setup arcs is necessary for feasibility because it implies that the setup costs of all families that contain a certain component are incurred whenever the component is ordered. In nonnested networks, nestedness on flow arcs implies that the flow of goods along each route is nested, i.e., whenever we order goods that will flow along a route \( n \) we also order goods that will flow along the sub-route of route \( n \).

A policy that is nested on all of the arcs of a nonnested network is said to be nested on routes. All nested policies are also nested on routes, but not vice-versa. With constant demand there is always a policy that is nested on routes and is within 2% of optimal, but the worst-case effectiveness of an optimal nested policy is zero [Ro84]. Still, it is not always possible to find an optimal policy that is nested on routes, as an example in Appendix A shows. Except in Section 4 where we prove the Lower Bound Theorem for Non-Nested Policies, we consider only policies which are nested on routes.
It is not hard to verify that any policy that is feasible for the system corresponds to a policy that is feasible for the nonnesting network and is nested on setup arcs, and vice-versa. It is also easily verified that Properties one and two can be assumed to hold in the nonnesting network, and that echelon inventories can be defined and costs can be calculated relative to the nonnesting network just as they were for the nested network.

Suppose that a family consists of a single component $k$. If $k$ is a final product then the node for family $(k)$ can be collapsed into the node for route $<k>$. If $k$ is not a final product and if $R_k = \{n\}$ then we can collapse the node for family $(k)$ into the unique node $n \in R_k$. In collapsing nodes the setup cost, echelon holding cost, and volume parameters are added. In every case one of the two numbers that are added together is zero.

Examples.

Consider two examples. The first is a manufacturer of two-piece pajamas. The components in our simplified system are material (component 1), tops (component 2), pants (component 3), and packaged pajamas (component 4). Material is used in both the tops and the pants. The product structure for this system is shown in Figure 2.
The second example is a table manufacturer who produces both a large table that can be expanded by inserting leaves, and a small table. The components in the system are table legs (component 1), finished small tables (component 2), end pieces to large and small table tops (component 3), leaves for large tables (component 4), and finished large tables (component 5). The product structure for the table manufacturer is also shown in Figure 1.

Figure 1. The Product Structures for the Pajama and Table Manufacturers.

The Pajama Manufacturer.       The Table Manufacturer.

The pajama manufacturer has a setup cost associated with the cutting and sewing operation (family \(\{2,3\}\)). The table
manufacturer has setup costs associated with the wood shop (family \((3,4)\)) and with assembly (family \((2,5)\)). The cost networks used to compute nested and non-nested policies for these systems are shown in Figures 2 and 3, respectively. The nodes with negative indices correspond to families and the nodes with indices greater than nine correspond to routes. In both cases the digits of the index correspond to the components in the family or route. In both of the nonnested networks in Figure 3 certain pairs of nodes have been collapsed.

Figure 2. Nested Networks for the Pajama and Table Manufacturers.

The Pajama Manufacturer.  The Table Manufacturer.
3. HEURISTICS FOR THE CORRELATED-DEMAND LOT-SIZING PROBLEM.

As we mentioned earlier, a cluster is a set of nodes in the cost network that order simultaneously because their setup costs and holding costs tend to make simultaneous ordering more economical. Optimal clusters can be computed using very efficient network algorithms in $O(N^4)$ time, where $N$ is the number of nodes in the cost network [MM83, Ro84]. An important property of
these clusters is that for any pair of clusters \( C^1 \) and \( C^2 \), if there is a directed path from a node of \( C^1 \) to a node of \( C^2 \), then there can not be directed path from a node of \( C^2 \) to a node of \( C^1 \).

In this section we propose several heuristics for solving the correlated-demand lot-sizing problem. The heuristics we propose produce only policies that are nested on all arcs of the cost network, and in which orders are placed simultaneously at all of the nodes in each cluster. Since we consider only policies of this type the clusters can be aggregated, resulting in a much smaller problem.

**Aggregating Clusters.**

We denote the clusters by \( \{ C^i, 1 \leq i \leq I \} \). The sets \( C^i \) form a partition of the nodes in the cost network. Consider a policy that is nested on all arcs of the cost network, and in which orders are placed at all of the nodes in each cluster simultaneously. Let \( s_{it} \) be the next time after time \( t \) that an order will be placed at the nodes in cluster \( i \), and let \( s_{it} = T + 1 \), where \( T \) is the number of time periods, if no orders are placed in cluster \( i \) after time \( t \). If \( n \in C^i \) then by equations (4) and (6) we have
\[ E_n^t = \sum_{r=t+1}^{s_{it-1}} \delta_r \equiv E_{it}. \] (7)

Thus the echelon inventory is the same for all nodes in the cluster.

Since all nodes in a cluster order simultaneously and have the same echelon inventory, the setup and echelon holding costs of the nodes in the cluster can be aggregated. Let \( K_i^i = \sum_{n \in C_i^i} K_n \) and \( H_i^i = \sum_{n \in C_i^i} H_n \). Then we can rewrite (4) as

\[ \sum_{n} \left[ J_i^i K_i^i + \sum_{t} H_i^{i,t} \right] \] (8)

where \( J_i^i \) is the number of orders placed at each of the nodes in cluster \( i \).

The ratio \( K_i^i / H_i^i \) is called the setup ratio for cluster \( i \). We can and do assume that any two clusters that have the same setup ratio have been aggregated. We obtain the cluster network from the cost network by collapsing all nodes in each cluster \( C_i \) into a single node \( i \). The cluster network is necessarily circuitless, and for each arc \( i \rightarrow j \) in the cluster network we have \( K_i^i / H_i^i > K_j^j / H_j^j \) [MM83, Ro84].

The problem of finding a minimum-cost policy that is nested on all arcs of the cost network and in which orders are placed at
all of the nodes in each cluster simultaneously can be written as

\[
(P) \quad \min: \quad \sum_{i} [J_{iK}^i + \sum_{t} H_{iE}^i] \tag{8}
\]

subject to: \( J_{i} = \) the number of orders placed at the nodes in cluster \( i \),

\[
E_{it} = \sum_{r=t+1}^{s_{it-1}} \delta_{r}, \quad \text{and} \tag{7}
\]

the policy is nested on all arcs. \( \tag{9} \)

**Heuristics.**

Although the problem (P) is much smaller than the original, the best known algorithm for solving it still has exponential running time. Rather than solving (P) directly we convert the cluster network into one that has distribution structure, i.e., into an acyclic network in which each node has at most one predecessor. We then apply Veinott's \( O(IT^3) \) algorithm for computing optimal nested policies in distribution systems to the resulting problem \([Ve69, Lo72]\). This approach is called the cluster heuristic.

We can transform the cluster network into a network that has distribution structure by applying the following algorithm.
Algorithm for Converting to Distribution Structure.

**Step 1.** Among the nodes in the network that have more than one direct predecessor, pick the node $j$ that has the smallest setup ratio. If there are no nodes that have more than one direct predecessor, stop.

**Step 2.** Among the direct predecessors of node $j$, let $i$ be the one that has the smallest setup ratio.

**Step 3.** For each arc $k \rightarrow j$, $k \neq i$, create the arc $k \rightarrow i$ and delete the arc $k \rightarrow j$. Then go to Step 1.

Consider the constraint (9) that a policy be nested on all arcs. Each arc added to the cluster network effectively adds another constraint of this type to (P). All arcs deleted from the cluster network correspond to redundant constraints. Therefore any policy that is feasible for the transformed system is also feasible for (P).

A simpler approach would be to convert the cluster network to a series system, or a simple chain. This is done by arranging the nodes $i$ of the cluster network in descending order of the setup ratios $K^i_i$. This approach results in a system that is more tightly constrained than the distribution system just described,
and consequently applying the cluster heuristic to this system will result in a policy whose cost is higher than it would have been if a distribution system were used. Veinott's algorithm can still be used.

A third option is to use sequential lot-sizing. We suggest that sequential lot-sizing be done on the cluster network rather than on the cost network. In sequential lot sizing we determine when orders will be placed at each cluster separately using the Wagner-Whitin algorithm. Later decisions are constrained by earlier ones because of the need to maintain nestedness.

There are several ways to sequence the clusters. In top-down sequential lot sizing we start with the cluster that has the largest setup ratio and work our way down, and in bottom-up sequential lot sizing we start with the cluster that has the smallest setup ratio and work our way up. In highest-cost sequential lot sizing we sequence the in decreasing order of \( K^iH^i \), a measure of the relative total cost associated with cluster \( i \).

In highest-cost sequential lot sizing we recommend that the cluster network be converted to a serial structure to simplify the problem of obtaining a reasonable nested schedule. If top-down sequential lot sizing is used, we recommend that the cluster network be converted to a distribution structure using the algorithm given above.
If bottom-up sequential lot sizing is used, we recommend that the cluster network be converted to a assembly structure. This can be done using the same algorithm used to convert to distribution structure, with the following modifications. Successors are used rather than predecessors, the cluster with the largest setup ratio is chosen in Step 2, and if the direction of the arcs is reversed in Step 3.

In Section 5 we study the worst-case effectiveness of several of these heuristics.

4. THE LOWER BOUND THEOREM.

In this section we prove the Lower Bound Theorem. This theorem gives a lower bound on the cost of any policy that is nested relative to the original system, and a lower bound on the cost of any feasible policy.

Note that if we relax (9) in (P), the problem decomposes by cluster. The resulting subproblem for cluster $i$ is a simple lot-sizing problem with setup cost $K^i$, holding cost $H^i$, and demands $\delta_t$, $0 \leq t < T$. We denote the optimal solution to this problem by $C^i$. The lower bound is $\Sigma_i C^i$. They are computed by performing a Wagner-Whitin computation for each cluster.

The fact that the lower bound is found by relaxing (9) implies that $\Sigma_i C^i$ it is a lower bound on the cost of all
schedules which are nested relative to the cluster network and in which orders are placed at all nodes in a cluster simultaneously. The Lower Bound Theorem states that $\sum_i c^i$ is a lower bound on the cost of any policy that is nested relative to the original system if the cost network is a nested network, and it is a lower bound on the cost of any feasible policy if the network is a nonnested network.

Clearly the cost of an optimal policy must lie between the lower bound and the cost of the policy generated by the cluster network. If the original system is a simple series system, the cost of the schedule produced by the cluster network is optimal [Za85]. In Appendix A we show that the cost of an optimal schedule can be arbitrarily close to the lower bound. Consequently one can never be sure where the cost of an optimal schedule lies relative to the cost of the cluster heuristic and the lower bound.

The proof of the Lower Bound Theorem relies on the fact that it is possible to allocate the holding costs incurred by a policy to the individual nodes in the cost network in such a way that:

* The sum of the holding costs incurred by a policy is greater than or equal to the sum of the holding costs allocated to the nodes,

* If each node independently minimizes the sum of its own setup costs and the holding costs allocated to it, the sum of the minimum costs is the lower bound.
The allocation of holding costs to nodes in the cost network is guided by the reallocated holding costs. These holding costs are the variables $H_n^*$ in the following nonlinear flow problem on the cost network. It is the dual of the convex program used to find the clusters [Ro84].

\[(NP) \quad \text{max:} \quad \sum_{n} \sqrt{K_n^* H_n^*} \]

such that: \[\sum_{n \rightarrow k} x_{nk} - \sum_{m \rightarrow n} x_{mn} + H_n^* = H_n^* \quad \text{for all } n , \quad (10)\]

\[x_{mn} \geq 0 \quad \text{for all } m \rightarrow n \quad \text{and} \quad H_n^* \geq 0 \quad \text{for all } n .\]

The reallocated holding costs satisfy $K_n^*/H_n^* = K_i^*/H_i^*$ for all $n \in C_i$ and for all $i$, and $\sum_{n \in C_i} H_n^* = H_i^*$ for all $i$.

Given a policy for the cost network, we define the echelon inventory $E_n^t$ for all setup nodes $n$ in the cost network using (7), i.e.,

\[E_n^t = \sum_{r=t+1}^{s_n^t-1} \delta_r \]

26
where \( s^t_n \) is the next time after time \( t \) that an order will be placed at node \( n \) in the cluster network.

Lemma 1. If a policy for the cost network is nested on all arcs, or if it is nested on all setup arcs and the cost network is a nonnested network, then \( E^t_m \geq E^t_n \) for all \( m \rightarrow n \) and for all \( t \).

Proof. The result follows directly from (7) if the policy is nested on all arcs. If the policy is nested on setup arcs and the cost network is a nonnested network, the result follows from (7) for setup arcs. If \( m \rightarrow n \) is an inventory arc, then node \( m \) has only one successor that is a route, namely node \( n \). For all other arcs \( m \rightarrow k \) leading out of node \( m \) the gozinto parameter \( \lambda_{mk} \) is zero. Therefore in (2) we have \( V_m = \lambda_{mn} V_n \), and consequently (3) becomes \( V_mE^t_m = I^t_m + \lambda_{mn} V_mE^t_n \geq V_mE^t_n \). Q.E.D.

Theorem 1 (The Lower Bound Theorem). If the cost network is a nested network then \( \Sigma_i C^i \) is a lower bound on the cost of an optimal nested policy for the original system. If the cost network is a nonnested network then \( \Sigma_i C^i \) is a lower bound on the cost of an optimal policy for the original system.
Proof. Suppose that we are given a policy that is nested on setup arcs if the cost network is a nonnested network, and that is nested on all arcs otherwise. By (4), (10), and Lemma 1 the cost of the policy is

\[ \sum_{n} \left( J_n K_n + \sum_{t} H_n E_n^t \right) \]

\[ = \sum_{n} \left( J_n K_n + \sum_{t} H_n^* E_n^t \right) + \sum_{t} \sum_{m \rightarrow n} x_{mn} (E_m^t - E_n^t) \]

\[ \geq \sum_{n} \left( J_n K_n + \sum_{t} H_n^* E_n^t \right). \]

Let \( C(K,H) \) be the cost of an optimal solution to the simple lot sizing problem with setup cost \( K \), holding cost \( H \), and demands \( \delta_t \), \( 0 \leq t < T \). Then \( C(\alpha K, \alpha H) = \alpha C(K, H) \). Therefore the cost of the policy is at least

\[ \sum_{i \in C} \left( J_{i n} K_n + \sum_{t} H_n^* E_n^t \right) \geq \sum_{i \in C} \sum_{n} C(K_n, H_n^*) \]

\[ = \sum_{i} \left( K_n / K_i \right) C(K_i^*, H_i^*) = \sum_{i} C(K_i^*, H_i^*) = \sum_{i} C_i \]. Q.E.D.
5. WORST-CASE ANALYSIS.

In this section we give examples to show that the effectiveness of the cluster heuristic can be as low as 70\% and that the effectiveness of the top-down and bottom-up sequential heuristics can be as low as 50\%. We then show that the worst-case effectiveness of the top-down sequential heuristic when applied to a serial cluster network is at least 50\%. Therefore the worst-case effectiveness of the cluster heuristic is between 50\% and 70\%, that of the top-down sequential heuristic is 50\%, and that of the bottom-up sequential heuristic is at most 50\%.

Examples

In presenting the examples, we allow the time intervals between successive time periods to be unequal. Since a problem with periods of rational, unequal lengths is equivalent to a problem with a larger number of periods of equal length in which many of the demands are zero, the distinction is not important from the point of view of establishing the worst-case effectiveness of our heuristics. Without loss of generality we assume that the demands in all time periods are strictly positive.

The proof of the following Lemma is given in Appendix C.
Lemma 2. The worst-case effectiveness of the cluster heuristic relative to the lower bound remains unchanged if the setup costs incurred at time zero considered to be free.

In all of the examples given below, it is assumed that $\epsilon \omega = 1$ and that $\epsilon$ is small. The orders at time zero are free, and the elapsed time between period $t-1$ and period $t$ is $L_t$.

The first example shows that the effectiveness of the cluster heuristic can be lower than 70%. The data for the example is given below. Also given are the times that orders are placed in both the one-stage problems that are solved to get the lower bound, and in the three optimal nested schedules (nested relative to the cluster network). The effectiveness of the optimal nested policy relative to the lower bound is $(9/13)(100\%) < 70\%$.

**EXAMPLE 1: THE CLUSTER HEURISTIC.**

**THE CLUSTER NETWORK.**

```
1 --2 --3
```
### The Costs and the Schedules

<table>
<thead>
<tr>
<th>$i$</th>
<th>$k_i$</th>
<th>$H_i$</th>
<th>$k_i/H_i$</th>
<th>Times that Orders are Placed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$\varepsilon$</td>
<td>$\omega$</td>
<td>1, 4, 6, 1, 4, 6, 1, 2, 4, 5, 6, 1, 3, 4, 5, 6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>$2\varepsilon^3$</td>
<td>$\omega^3$</td>
<td>2, 5, 1, 4, 6, 2, 5, 1, 3, 5</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>$2\varepsilon^{11}$</td>
<td>$\omega^{11}$</td>
<td>3, 1, 4, 1, 5, 3</td>
</tr>
</tbody>
</table>

**Costs**

<table>
<thead>
<tr>
<th>Opt1</th>
<th>Opt2</th>
<th>Opt3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$7+O(\varepsilon)$</td>
<td>$13+O(\varepsilon)$</td>
<td>$13+O(\varepsilon)$</td>
</tr>
</tbody>
</table>

### The Demands and Intervals Between Time Periods

\[
\begin{array}{c|ccccccc}
 t & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
L_t & - & 1 & \varepsilon^{14} & \varepsilon^{16} & \varepsilon^{18} & \varepsilon^{24} & \varepsilon^{28} \\
\hline
 d_t & 1 & 1 & \omega^{10} & \omega^{12} & \omega^{18} & \omega^{22} & \omega^{28} \\
\end{array}
\]

\[L_s d_t, \; s \leq t.\]

\[
\begin{array}{c|ccccccc}
 t & 6 & 5 & 4 & 3 & 2 & 1 \\
\hline
 s & \omega^{28} & \omega^{14} & \omega^{12} & \omega^{10} & \omega^{4} & 1 \\
\hline
 6 & \omega^{28} & \omega^{14} & \omega^{12} & \omega^{10} & \omega^{4} & 1 \\
 5 & \omega^{22} & \omega^{8} & \omega^{6} & \omega^{4} & \varepsilon^{2} & 1 \\
 4 & \omega^{18} & \omega^{4} & \omega^{2} & 1 & 1 & 1 \\
 3 & \omega^{12} & \varepsilon^{2} & \varepsilon^{4} & 1 & 1 & 1 \\
 2 & \omega^{10} & \varepsilon^{4} & 1 & 1 & 1 & 1 \\
 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]
The second example shows that the effectiveness of the bottom-up sequential heuristic can be as low as 50%.

EXAMPLE 2: THE BOTTOM-UP SEQUENTIAL HEURISTIC.

THE CLUSTER NETWORK

```
1 -- 2
```

THE COSTS AND THE SCHEDULES

<table>
<thead>
<tr>
<th>i</th>
<th>$K^i$</th>
<th>$H^i$</th>
<th>$K^i/H^i$</th>
<th>Times that Orders are Placed</th>
<th>Lower Bound</th>
<th>Opt1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1,3</td>
<td>1,3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$\omega^2$</td>
<td>1</td>
<td>$\omega^2$</td>
<td>2</td>
<td>1,3</td>
<td></td>
</tr>
<tr>
<td>Cost</td>
<td></td>
<td></td>
<td></td>
<td>$\omega^2 + O(1)$</td>
<td>$2\omega^2 + O(1)$</td>
<td></td>
</tr>
</tbody>
</table>

THE DEMANDS AND INTERVALS BETWEEN TIME PERIODS

<table>
<thead>
<tr>
<th>$t$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_t$</td>
<td>-</td>
<td>1</td>
<td>$\epsilon^4$</td>
<td>$\epsilon^6$</td>
</tr>
<tr>
<td>$d_t$</td>
<td>1</td>
<td>$\omega$</td>
<td>$\omega^3$</td>
<td>$\omega^7$</td>
</tr>
</tbody>
</table>
\[ L_{s, t}, s \leq t. \]

<table>
<thead>
<tr>
<th>( t )</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>( \omega )</td>
<td>( \omega^3 )</td>
<td>( \omega )</td>
</tr>
<tr>
<td>2</td>
<td>( \omega^3 )</td>
<td>( \varepsilon )</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( \omega )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The next example shows that the effectiveness of the top-down sequential heuristic can be arbitrarily close to 50%. In this example the number of time periods is an odd integer \( T \).

**EXAMPLE 3: THE BOTTOM-UP SEQUENTIAL HEURISTIC.**

**THE CLUSTER NETWORK**

\[ \begin{array}{c}
1 \\
\rightarrow \\
2 
\end{array} \]
### THE COSTS AND THE SCHEDULES

<table>
<thead>
<tr>
<th>$i$</th>
<th>$K^i$</th>
<th>$H^i$</th>
<th>$K^i/H^i$</th>
<th>Times that Orders are Placed</th>
<th>Lower Bound</th>
<th>Opt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$\omega$</td>
<td>$\varepsilon$</td>
<td>$t$ odd</td>
<td>all</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$\omega^2$</td>
<td>1</td>
<td>$\omega^2$</td>
<td>$t$ even</td>
<td>$t$ even</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cost</td>
<td></td>
<td></td>
<td>$\frac{T+1}{2} + O(\varepsilon)$</td>
<td>$T + O(\varepsilon)$</td>
<td></td>
</tr>
</tbody>
</table>

### THE DEMANDS AND INTERVALS BETWEEN TIME PERIODS

<table>
<thead>
<tr>
<th>$t$</th>
<th>$t$ odd</th>
<th>$t$ even</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_t$</td>
<td>$\varepsilon^{3t-3}$</td>
<td>$\varepsilon^{3t-2}$</td>
<td>-</td>
<td>1</td>
<td>$\varepsilon^4$</td>
<td>$\varepsilon^6$</td>
</tr>
<tr>
<td>$d_t$</td>
<td>$\omega^{3t-3}$</td>
<td>$\omega^{3t-4}$</td>
<td>$\varepsilon^3$</td>
<td>1</td>
<td>$\omega^2$</td>
<td>$\omega^6$</td>
</tr>
</tbody>
</table>

$L_s d_t$, $s \leq t \leq 5$

<table>
<thead>
<tr>
<th>$s$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>$\omega^{12}$</td>
<td>$\omega^8$</td>
<td>$\omega^6$</td>
<td>$\omega^2$</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>$\omega^8$</td>
<td>$\omega^4$</td>
<td>$\omega^2$</td>
<td>$\varepsilon^2$</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$\omega^6$</td>
<td>$\omega^2$</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$\omega^2$</td>
<td>$\varepsilon^2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Lower Bound on the Effectiveness of the Cluster Heuristic.

Throughout the remainder of this section, it is assumed that among optimal schedules for a one-stage lot-size problem, an optimal schedule in which the number of orders placed is maximal is selected. Among optimal schedules in which the number of orders placed is maximal, the one in which the orders occur as early in the time horizon as possible is selected. This is the schedule that would be chosen if the problem were perturbed by subtracting $\varepsilon^t$ from the setup cost in period $t$, where $\varepsilon$ is a very small positive number. With this in mind, the optimal schedule will be considered to be unique.

Let $[P^i_r, s]$ be the one-stage lot-sizing problem with setup cost $K^i$, holding cost $H^i$, and demands $\delta^r_t$, $r \leq t < s$. Let the cost of the optimal schedule for this problem be $C^i_{r,s}$ and let the number of orders placed in the optimal schedule be $N^i_{r,s}$.

Consider a serial cluster network. Suppose that the clusters are indexed so that $K^i/H^i > K^{i+1}/H^{i+1}$ for all $i$. In the top-down sequential heuristic, we begin with the optimal schedule for $[P^1_{0,T}]$ where $T$ is the number of time periods in the original problem. Let $[t_0 = 0, t_1, \ldots, t_{N^1_{0,T-1}}]$ be the times at which orders are placed in this schedule. These are the times at which orders will be placed in cluster 1. In order to be sure that
the policy is nested, we require that orders be placed at these points in time in all of the other clusters as well.

This splits the problem of determining when (and if) additional orders will be placed at the other clusters into smaller problems of the same type as the original. The cluster network for each of the subproblems is the same as the original, except that cluster 1 is removed. The demands for the \( m \)th subproblem are \( \delta_t, \ t_m \leq t < t_{m+1} \) where \( t_{N_0, T}^i = T \). The procedure just described is then applied to each of the subproblems.

Let \( i \) be a cluster index. At a given point in time during the generation of a schedule for the system using the sequential top-down heuristic, we represent the set of times at which cluster \( i \) is currently constrained to place an order by \( J_i = \{ t_0=0, t_1, \ldots, t_{M} = T \} \) where \( t_0 \leq t_{0+1} \) for all \( 0 \leq \alpha \leq M \). Initially \( J_i = \{0, T\} \), and as the algorithm progresses points are added to \( J_i \). We will suppress the superscript \( i \) on \( t_m^i \).

Let \( c^i(J_i) = \sum_{0 \leq m < M} C^i_{t_m^i t_{m+1}^i} \). Then initially \( c^i(J_i) = c^i(\{0, T\}) = C \), and when a schedule for the system has been completely generated using the sequential top-down heuristic, \( \sum_i c^i(J_i) \) will be the cost of the schedule.

The following lemma controls the way in which times are added to the set \( J_i \).
Lemma 3. Suppose that in the process of using the sequential

top-down heuristic, cluster \( i \) is currently constrained to place

orders at times \( \mathcal{J}^i \equiv (t_0=0, t_1, \ldots, t_M=T) \) where \( t_{2q} \leq t_{2q+1} \) for

all \( 0 \leq q \leq M \). Time \( t \), \( t_m < t < t_{m+1} \), is about to be added

to \( \mathcal{J}^i \). Then

\[* c^i(\mathcal{J}^i) + K^i \geq c^i(\mathcal{J}^i \cup \{t\}) \] \quad (11)

\[* \text{If } \frac{N^i_{t_m, t}}{t_m+1} = 2 \text{ then } c^i(\mathcal{J}^i) = c^i(\mathcal{J}^i \cup \{t\}) \] \quad (12)

\[* N^i_{t_m, t} + N^i_{t, t_{m+1}} \leq N^i_{t_m, t_{m+1}} + 1 \] \quad (13)

\[* \text{if either } N^i_{t_m, t} = 1 \text{ or } N^i_{t, t_{m+1}} = 1 \text{ then } \]

\[ N^i_{t_m, t} + N^i_{t, t_{m+1}} \leq N^i_{t_m, t_{m+1}}, \text{ and} \]

\[* N^i_{t_m, t_{m+1}} \geq 2 \] \quad (15)

Proof. The lemma is clearly true if an order is placed at
time \( t \) in an optimal schedule for \( [p^i_{t_m, t_{m+1}}] \). Assume that

this is not the case. Since \( c^i_{t_m, t_{m+1}} + K^i \) is the cost of

inserting an order at time \( t \) into the optimal schedule for

\( [p^i_{t_m, t_{m+1}}] \), it is clearly greater than or equal to \( c^i_{t_m, t} + c^i_{t, t_{m+1}} \). Therefore (11) holds.
Let \( N_{t_m, t_{m+1}}^i = 2 \). Since an order is placed at time \( t \) in the optimal schedule for \([P_{t_m, t_{m+1}}^j, t_{m+1}^j]\) for some \( j < i \), and since \( K_{j, H_j}^j > K_{i, H_i}^i \), we must have \( N_{t_m, t_{m+1}}^j = 2 \). But this contradicts our assumption that there can not be an order at time \( t \) in an optimal schedule for \([P_{t_m, t_{m+1}}^i, t_{m+1}^i]\). Therefore (12) holds.

Equations (13) and (14) are consequences of the following well-known property, which we will state without proof.

**Property 1.** Suppose that \( N \) is the number of orders in the optimal schedule for a one-stage problem whose demands are \( \delta_t \), \( 0 \leq t \leq \tau \). Then the optimal schedule for a problem with the same costs, but whose demands are \( \delta_r \), \( r \leq t \leq s \) where \( 0 \leq r \) and \( s \leq \tau \), has at most \( N \) orders.

Consider the optimal schedule for \([P_{t_m, t_{m+1}}^i, t_{m+1}^i]\). Let the order immediately before time \( t \) be at time \( s \) and let the order immediately after time \( t \) be at time \( u \). Equation (13) follows by applying Property 1 to the pair of problems \([P_{t_m, u}^i, t_{m+1}^i]\) and to the pair of problems \([P_{s, t_{m+1}}^i, t_{m+1}^i]\).

Let \( N_{t_m, t}^i = 1 \), and let \( s \) be the time at which the first order after time \( t_m \) is placed in the optimal schedule for \([P_{t_m, t_{m+1}}^i, t_{m+1}^i]\). Then \( N_{s, t_{m+1}}^i < N_{t_m, t_{m+1}}^i \). Since an order is placed at time \( t \) in the optimal schedule for \([P_{t_m, t_{m+1}}^j, t_{m+1}^j]\) for some
j < i, and since $K_j^j / H_j^j > K_i^i / H_i^i$, we must have $s \leq t$. Equation (14) now follows by applying Property 1 to the pair of problems 
$[P_s^i, t_{m+1}^i], [P_t^i, t_{m+1}^i]$. A parallel argument establishes (14) in the case $N_{t_i}^i, t_{m+1}^i = 1$.

Finally, (15) is a direct consequence of (13) and (14).

Q.E.D.

Theorem 2. The worst-case effectiveness of the cluster heuristic is at least 50%.

Proof. Consider cluster $i$. Let $s^i$ be the final value that the set $J^i$ takes on in the top-down sequential heuristic, i.e., the set of all times at which cluster $i$ places an order in the schedule generated by the top-down sequential heuristic. We will show that $c^i(s^i) \leq 2c_i^i$. Since the lower bound is $\Sigma_i c_i^i$ and the cost of the schedule computed by the heuristic is $\Sigma_i c_i^i(s^i)$, this will establish the result.

We want to find a function $f$ such that $f(N_{0_i}^i, T)$ is an upper bound on $[c_i^i(s^i) - c_i^i]/k_i^i$. By Lemma 3 we can set $f(1) = f(2) = 0$, and

$$f(n) = \max \left\{ \{f(n-1) + f(1)\}, \max_{1 \leq m \leq n-2} \{f(m+1) + f(n-m) + 1\} \right\}.$$
The solution to the above equation is clearly given by
\[ f(n) = n - 2 \quad \text{for all} \quad n \geq 2. \]
Therefore
\[ c^i(s^i) - c^i \leq k^i[N^{i}_{0,T} - 2] \leq c^i. \]
Q.E.D.

6. COMPUTATIONAL TESTING.

In this section we report on the results of computational tests of the cluster heuristic and the bottom-up sequential heuristic. In the tests, the cost of the heuristics were all compared to the lower bound, not to the optimal solution. Cost data was generated directly for the cluster network rather than for the components, and only cluster networks with serial structure were used.

Therefore the actual effectiveness of the heuristics are higher than the numbers reported here, and the number of components in the system is much larger than the number of clusters. For example, the industrial system illustrated in [MM83] has 50 components and 7 clusters.

The data needed to describe an instance of the problem is the following: \( I \), the number of clusters; \( T \), the number of time
periods; $K^i/H^i$, $1 \leq i \leq I$, proportional to the square of the
natural reorder interval for cluster $i$; $K^i/H^i$, $1 \leq i \leq I$, a
measure of the magnitude of the costs associated with cluster $i$, and
$\delta_t$, $0 \leq t < T$, the demands.

Values of $K^i/H^i$, which are proportional to the square of the
natural reorder interval for cluster $i$, were generated by
assuming that $\frac{1}{2} \log_2 (K^i/H^i)$ is uniformly distributed on the
interval $[\frac{1}{2}, b + \frac{1}{2}]$. This assumption amounts to saying that
the natural order intervals all lie between $2^{0.5}$ and $2^{b+0.5}$, and
that within this interval a natural order interval is as likely to
lie between 2 and $2\alpha$ as it is to lie between $2\beta$ and $2\alpha\beta$.

Our computational experience confirms the following
conjectures regarding the sensitivity of the effectiveness of the
heuristic to the data.

* The effectiveness of the heuristic is lowest when $K^i/H^i \equiv 1$.

* The effectiveness of the heuristic is decreasing in $T$, but is
  roughly independent of $T$ for $T > 2^{b+1.5}$.

* With $T = 2^{b+1.5}$, the effectiveness of the heuristic is
decreasing in $b$.

* The effectiveness of the heuristic is decreasing in $I$. 

41
* The effectiveness of the heuristic decreases as the variability in demand decreases, and is lowest for the case of constant demand.

The first four conclusions were expected, but the last one came as a total surprise. For constant demand a heuristic based on the same clusters we use was shown to have a worst-case effectiveness of at least 98% or, if the order intervals are required to be multiples of a given basic planning period, of at least 94% [Ro84, MM83]. These results do not apply directly because holding costs are not calculated in the same way in a discrete-time model, but they led us to expect the average effectiveness to be lower for non-constant demand than it is for constant demand.

The reason that the effectiveness is lower for constant demand seems to be that non-constant demand induces all of the clusters to place orders when higher-than-average demands occur. This tends to make the solutions to the one-stage problems solved in computing the lower bound naturally nested, and to make nested schedules near-optimal when the solutions to the one-stage problems are not nested.

Summary statistics from the tests are given in Table 5 below, together with information regarding the way in which the data was
generated. Sample means and 95% confidence intervals are reported. In all of the test runs except the first one we used $k^1/H^1 \equiv 1$, which makes the highest-cost heuristic ineffective. Therefore it was not used. Note that in most cases the experiments were designed to make the cluster heuristic perform poorly.

In all, 2,610 test problems were solved. The lowest value of the effectiveness of the cluster heuristic that was observed in these test problems was 97.73%, and it occurred in a problem with constant demand. Among the 625 test problems on which the sequential heuristic was tested, the lowest effectiveness that was observed was 96.3%. It also occurred in a problem with constant demand.
Table 5.

SENSITIVITY OF MEAN EFFECTIVENESS TO VARIABILITY IN $K_i^H_i$.

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>1</th>
<th>1/2</th>
<th>1/4</th>
<th>1/8</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>.9965</td>
<td>.9958</td>
<td>.9954</td>
<td>.9954</td>
<td>.9952</td>
</tr>
<tr>
<td>$\tau$</td>
<td>.0008</td>
<td>.0009</td>
<td>.0009</td>
<td>.0009</td>
<td>.0009</td>
</tr>
</tbody>
</table>

$K_i^H_i$: Drawn from a gamma distribution with mean 1 and standard deviation $\sigma$.

$K_i^H_i$: As described above, with $b = 3$.

Demands: Drawn from a gamma distribution with mean 1 and standard deviations 1, 1/2, 1/4, 1/8, and 0.

Number of time periods: $T = 16$.

Number of clusters: $I = 5$.

Number of Observations: 125 per column.

SENSITIVITY OF MEAN EFFECTIVENESS TO THE NUMBER OF TIME PERIODS.
(One of several runs with different values of $b$.)

<table>
<thead>
<tr>
<th>$T$</th>
<th>12</th>
<th>14</th>
<th>16</th>
<th>19</th>
<th>22</th>
<th>24</th>
<th>28</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>.9963</td>
<td>.9961</td>
<td>.9950</td>
<td>.9936</td>
<td>.9943</td>
<td>.9935</td>
<td>.9943</td>
<td>.9931</td>
</tr>
<tr>
<td>$\tau$</td>
<td>.0014</td>
<td>.0012</td>
<td>.0014</td>
<td>.0015</td>
<td>.0012</td>
<td>.0014</td>
<td>.0012</td>
<td>.0012</td>
</tr>
</tbody>
</table>

$K_i^H_i \equiv 1$.

$K_i^H_i$: As described above, with $b = 3$.

Demands: Drawn from a gamma distribution with mean 1 and standard deviations 1, 1/2, 1/4, 1/8, and 0.

Number of clusters: $I = 5$.

Number of Observations: 50 per column.
SENSEIVITY OF MEAN EFFECTIVENESS TO VARIABILITY IN THE SETUP RATIOS.

<table>
<thead>
<tr>
<th>b</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>μ</td>
<td>.9993</td>
<td>.9976</td>
<td>.9955</td>
<td>.9948</td>
<td>.9931</td>
</tr>
<tr>
<td>±</td>
<td>.0005</td>
<td>.0006</td>
<td>.0006</td>
<td>.0006</td>
<td>.0006</td>
</tr>
</tbody>
</table>

\[ K_i^H \equiv 1. \]

\[ K_i^H / H_i : \] As described above, with \( b \) as shown above.

Demands: Drawn from a gamma distribution with mean 1 and standard deviations 1, 1/2, 1/4, 1/8, and 0.

Number of time periods: \( T = 2^{b+1.5} \).

Number of clusters: \( I = b, 1.5b, 2b, 3b, 4b, 6b \).

Number of Observations: 150 per column.
SENSITIVITY OF MEAN EFFECTIVENESS TO NUMBER OF CLUSTERS.

<table>
<thead>
<tr>
<th>I</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster Heuristic</td>
<td>μ</td>
<td>.9981</td>
<td>.9951</td>
<td>.9934</td>
<td>.9920</td>
</tr>
<tr>
<td></td>
<td>±</td>
<td>.0007</td>
<td>.0009</td>
<td>.0009</td>
<td>.0009</td>
</tr>
<tr>
<td>Bottom-up Sequential Heuristic</td>
<td>μ</td>
<td>.9967</td>
<td>.9921</td>
<td>.9883</td>
<td>.9874</td>
</tr>
<tr>
<td></td>
<td>±</td>
<td>.0010</td>
<td>.0014</td>
<td>.0014</td>
<td>.0012</td>
</tr>
</tbody>
</table>

\[ K^iH^i = 1 \, . \]

\[ K^i/H^i \] : As described above, with \( b = 3 \).

Demands: Drawn from a gamma distribution with mean 1 and standard deviations 1, 1/2, 1/4, 1/8, and 0.

Number of time periods: \( T = 24 \geq 2^{b+1.5} \).

Number of clusters: \( I \).

Number of Observations: 125 per column.
SENSITIVITY OF MEAN EFFECTIVENESS TO VARIABILITY IN DEMAND.

<table>
<thead>
<tr>
<th>σ</th>
<th>1</th>
<th>1/2</th>
<th>1/4</th>
<th>1/8</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster Heuristic</td>
<td>μ</td>
<td>.9974</td>
<td>.9962</td>
<td>.9939</td>
<td>.9925</td>
</tr>
<tr>
<td></td>
<td>±</td>
<td>.0006</td>
<td>.0007</td>
<td>.0008</td>
<td>.0009</td>
</tr>
<tr>
<td>Bottom-up Sequential Heuristic</td>
<td>μ</td>
<td>.9941</td>
<td>.9919</td>
<td>.9892</td>
<td>.9877</td>
</tr>
<tr>
<td></td>
<td>±</td>
<td>.0013</td>
<td>.0013</td>
<td>.0013</td>
<td>.0015</td>
</tr>
</tbody>
</table>

\[ K_i^{H_i} = 1 \]

\[ K_i^{H_i} : \text{As described above, with } b = 3. \]

Demands: Drawn from a gamma distribution with mean 1 and standard deviations 1, 1/2, 1/4, 1/8, and 0.

Number of time periods: \( T = 24 = 2^{b+1.5} \).

Number of clusters: \( I = 2, 4, 8, 16, 32 \).

Number of Observations: 125 per column.

References


APPENDIX A.

In this appendix we give an example that shows that an optimal policy need not be nested on the flow arcs of a nonnested network. Consider the system whose bill of materials is shown in Figure 4. There is external demand for both of the components. We have $h_1 = h_2 = 1$, $\lambda_{12} = \varepsilon$, and $v_1 = v_2 = 1$.

The nonnested network for the system and the setup and holding cost coefficients for the nodes are shown in Figure 5. The inventory at route $\langle 1,2 \rangle$ is negligibly small, a fact which is reflected in the small value of $H_{12}$. Arc $12 \rightarrow 1$ is a setup arc and arc $12 \rightarrow 2$ is a flow arc.

A policy must be nested on $12 \rightarrow 1$ to be feasible. In an optimal policy we would allow nodes 1 and 2 to schedule orders so as to minimize their own costs, and let node 12 place orders whenever node 1 does. This policy would often not be nested on $12 \rightarrow 2$ (e.g., $\delta_t = 1/2$).
APPENDIX B.

In this appendix we give an example that shows that it may not be optimal for all nodes in a cluster to place orders simultaneously, and that the cost of the optimal policy can be equal to the lower bound. Consider the system whose bill of materials is illustrated in Figure 6 and in Table 2. For this system, both the nested network and the nonnested network are the same as the network in Figure 9.
Figure 6: The System.

Table 2: The Costs.

<table>
<thead>
<tr>
<th>Components</th>
<th>Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>K_n</td>
</tr>
<tr>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>e</td>
</tr>
<tr>
<td>5</td>
<td>e</td>
</tr>
<tr>
<td>6</td>
<td>e</td>
</tr>
</tbody>
</table>

Since the cluster network has serial structure, the cluster network requires the schedule for node 2 to be nested relative to that of node 1 and the schedule for 3 to be nested relative to the schedule for 2. This can clearly result in a cost which is higher than the lower bound (e.g., δ_t ≡ 1).
However, the following schedule is nested in both the nested network and the nonnested network, and its cost is within $\varepsilon$ of the lower bound. For components $i$, $1 \leq i \leq 3$, use the schedule that is optimal for the one-stage problem with setup cost $K^i$, holding cost $H^i$, and with the demands $\delta_t$ of the original problem. For components $i$, $4 \leq i \leq 6$, an order is placed whenever an order is placed at any predecessor of $i$. The cost of this nested schedule is within $O(\varepsilon)$ of the lower bound.

It is easily seen that given any serial cluster network, a corresponding system can be found whose optimal policy is nested and has a cost within $\varepsilon$ of the lower bound.

APPENDIX C.

In this appendix we prove Lemma 2.

Lemma 2. The worst-case effectiveness of the cluster heuristic relative to the lower bound remains unchanged if the setup costs incurred at time zero considered to be free.

Proof. The fact that the orders at time zero are free reduces the cost of each feasible schedule and of the lower bound by $\Sigma_i K^i$. Therefore the worst-case effectiveness of the cluster
heuristic can not be higher when the orders at time zero are free than it would be if they were not.

Suppose we are given the problem (P) of scheduling orders in a serial cluster network with demands \( \delta_t \), \( 0 \leq t \leq T \), in which the elapsed time between period \( t-1 \) and period \( t \) is \( L_t \), \( 0 < t \leq T \), and in which orders at time zero are free for all clusters.

Given (P) we will create another problem (P') with the same clusters, but with different demands and different elapsed times between periods, and in which the orders at time zero are not free. We then show that the effectiveness of the cluster heuristic when applied to (P) is bounded below by a number arbitrarily close to the effectiveness of the cluster heuristic when applied to (P'). This will establish the result.

Let \( M \) be a large integer and let \( \varepsilon > 0 \) be small. We define (P') to be the problem of scheduling orders for the same serial cluster network used in (P), but with the following demands and intervals between time periods.

\[
\delta_{t+mT} = \delta_t \varepsilon^m \quad ; \quad 0 \leq m \leq M \, , \, 1 \leq t \leq T ,
\]
\[
L_{t+mT} = L_t \varepsilon^{-m} \quad ; \quad 0 \leq m \leq M \, , \, 1 \leq t \leq T .
\]

(16)

Note that in time periods \( t \), \( 0 \leq t \leq T \), the demands and the
intervals between time periods are the same for (P) as they are for (P'). We subdivide the time horizon for (P') into smaller intervals as follows. Interval $I_m$ will consist of time periods $t, mT+1 \leq t \leq (m+1)T$, where $0 \leq m \leq M$. The holding costs on goods held during the $L_t$ days between time period $t-1$ and time period $t$ are associated with time period $t$.

Let $S$ be the schedule for (P) that is produced by the cluster heuristic, and let $S'$ be the schedule for (P') that is produced by the cluster heuristic. The orders placed by $S'$ in the interval $I_m$ correspond to a schedule $S_m$ for (P) as follows. An order is placed at each cluster at time 0 in $S_m$, and an order is placed at time $t$ in $S_m$ if and only if an order is placed at time $t + mT$, $1 \leq t \leq T$ in $S'$.

Since the order at time zero in $S_m$ is free, (16) implies that for all $m$ the cost $c(S_m)$ of $S_m$ is equal to the setup costs incurred by $S'$ in $I_m$, plus the holding costs incurred in $I_m$ in holding goods that are used to satisfy the demand that occurs in $I_m$.

Let $H_m(S')$ be the the holding costs incurred in time periods in $I_m$ in holding goods that are used to satisfy the demands that occur after the last time period in $I_m$. By (16), $H_m(S') = O(\epsilon)$. Since $c(S_m) \geq c(S)$, the cost $c'(S')$ satisfies
\[ c'(S') = \sum_{i} K_{i}^{i} + \sum_{m} \left[ c(S_{m}) + H_{m}(S') \right] \]

\[ \geq (M + 1) \left[ c(S) + \mathcal{O}\left( \varepsilon + \frac{1}{M} \right) \right] . \] (17)

Recall that the lower bound \( B \) for (P) is found by solving one-stage problems, one for each individual cluster in the cluster network, and summing the costs of the solutions to these problems. Let \( R \) be the times at which orders are placed in the optimal schedules for these one-stage problems. We create a corresponding schedule \( R' \) for (P') as follows. An order is placed at each cluster at time \( 0 \) in \( R' \), and for each \( m, 0 \leq m \leq M \), an order is placed at time \( t + mT \), \( 1 \leq t \leq T \) in \( R' \) if and only if an order is placed at time \( t \) in \( R \). The cost \( c(R) \) of \( R \) and the cost \( c'(R') \) are related by the equation

\[ c'(R') = \sum_{i} K_{i}^{i} + \sum_{m} \left[ c(R) + H_{m}(R') \right] . \]

\[ = (M + 1) \left[ c(R) + \mathcal{O}\left( \varepsilon + \frac{1}{M} \right) \right] . \]

The fact that \( c(R) = B \) and \( c'(R') \geq B' \) implies that

\[ B' \leq (M + 1) \left[ B + \mathcal{O}\left( \varepsilon + \frac{1}{M} \right) \right] . \] (18)
By (17) and (18),

\[
\frac{B'}{c'(S')} \leq \frac{B}{c(S)} + O[\varepsilon + \frac{1}{M}].
\]

Since the effectiveness of the cluster heuristic is

\[\left[\frac{B}{c(S)}\right](100\%) \text{ for } (P) \text{ and } \left[\frac{B'}{c'(S')}\right](100\%) \text{ for } (P'),\]

the result is established by letting \(\varepsilon + 1/M\) tend to zero. Q.E.D.