SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK

TECHNICAL REPORT NO. 731

March 1987

MULTI-ECHELON AND JOINT REPLENISHMENT PRODUCTION AND DISTRIBUTION SYSTEMS WITH NON-STATIONARY DEMANDS

By

DEV JONEJA

*Supported in part by NSF grant number DMC-8451984, and by AT&T Communication Systems and DuPont Corporation.

1. Introduction

A multi-echelon production and/or distribution system involves two or more interrelated activities. The most common notion is of a distribution system in which products flow from a factory to regional warehouses and from there to retailers who satisfy external demand, or of production and assembly lines with several stages. We are considering systems that produce several items whose demand over a certain time in the future (the time horizon) is assumed to be known. This is especially relevant in environments where demand forecasts are used as a basis for planning. We do not require demand to be constant over time, but allow it to vary from one time period to another. This also takes into account seasonality and other non-stationary demand characteristics.

The costs we are considering are ordering costs or setup costs for the various items, and inventory holding costs. An important aspect of the model we are considering is relationships between the various items. It is often the case that families of products exist which, if ordered or produced together, result in lower costs. For example, items supplied to a retailer from the same wholesaler can constitute one family, incurring lower transportation costs if they are ordered together than if they are ordered in separate time periods. Thus it is natural to model these families using a joint setup (ordering) cost which is incurred whenever any member of the family is ordered. We are focussing on decisions about when to order or produce each item, and in what lot sizes, in order to

minimize the total ordering and inventory holding cost over the planning horizon.

2. The Joint Replenishment Problem

As a first step, consider the simplified structure provided by the joint replenishment problem. This problem concerns the reorder intervals of several items over a given time horizon. The external demand for each item must be met in every time period, and backorders are not permitted. Each item has a fixed ordering cost and a linear inventory holding cost. Further, a joint cost is incurred whenever one or more items is ordered, so that economies can be effected by ordering several items together.

This problem arises in a number of settings. In the context of manufacturing, this problem can arise for example when deciding what items to produce when a die is mounted on a machine. The die has several cavities that can be open or closed, depending on which an item may or may not be produced. The joint cost is now associated with the time it takes to stop and restart the operation when any subset of the cavities need to be opened or blocked, while the individual ordering cost is associated with the time required to set up the particular cavity. In a distribution system, consider several retailers who are supplied an item from the same central supplier or warehouse (the one-warehouse multi-retailer problem). Transportation cost is incurred when the warehouse gets its supplies from the factory. Schwarz (1975) argues that very often

the central warehouse does not actually store inventory but only acts as an order processing and transshipment center. It is easy to show that the warehouse places an order only when one or more of the retailers do. Using this fact this problem is clearly reduced to an instance of the joint replenishment problem.

Problems with joint costs have been studied by a number of authors. The infinite time horizon case with constant demands, with the objective of minimizing the average cost per time period, has been studied by Jackson, Maxwell and Muckstadt (1985). They develop a heuristic algorithm which is always within 6% of the optimal solution. The infinite horizon case with constant demands, but in a more general production and distribution model, is considered by Roundy (1984). Enumerative procedures are suggested by Goyal (1974). In the finite horizon case with non-constant demands, Zangwill (1966) and Vienott (1969) propose dynamic programming algorithms. Both these algorithms take exponential time, and can only be used with very small problems. Another dynamic programming formulation is provided by Silver (1979). Kao (1979) also has a dynamic programming formulation, which has a smaller state space than Zangwill's. He also suggests a heuristic for the problem, but with no bounds on its performance.

3. The Model

We model this problem as a two-level network structure as shown in Fig. 1. Each of the item nodes $1,..,N$ corresponds to an

item, and has associated ordering and inventory holding costs, while the node *0* represents the joint cost. Let

d_{nt} = demand for item n in time period t

H_n = inventory holding cost of item n per unit per time period

K_n = ordering cost of item n

K_0 = joint ordering cost

A *nested policy* is defined as one in which every time a node places an order, its successor nodes in the network also place an order at that time. Thus a nested policy on our network assures that the joint cost is incurred whenever any item is ordered. We can easily show that the optimal policy has the *zero order property*: an item is ordered in a period only if the incoming inventory is zero. Using this, we define:

C_{nst} = cost for item n incurred in periods s to (t-1) if it is ordered in periods s and t, and nowhere in between.

$$= K_n + H_n \sum_{i=s+1}^{t-1} (i-s)d_{ni}$$

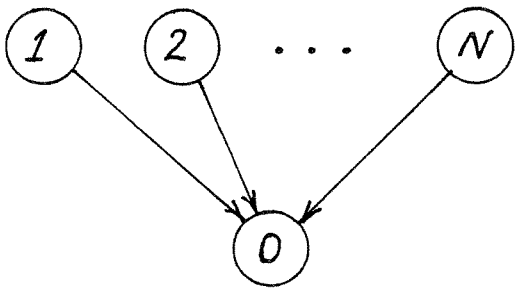

Fig. 1: Network Structure of the Joint Replenishment Problem.

$$X_{nst} = \begin{cases} 1, & \text{if item n is ordered in periods s and t, and} \\ & \quad \text{nowhere in between.} \\ 0, & \text{otherwise.} \end{cases}$$

$$X_t = \begin{cases} 1, & \text{if any item is ordered in period t.} \\ 0, & \text{otherwise.} \end{cases}$$

For each item *n*, consider a network where there is a node for each time period. In this network, from any node *s* there is an arc to each node $t > s$, with the associated cost (length) C_{nst}, as shown in Fig. 3. Consider any path from node 1 to node *T* on this network. If an arc X_{nst} lies on this path we interpret it as item *n* being ordered in time periods *s* and *t*, and nowhere in between, at a cost of C_{nst}. The path then represents a feasible ordering policy for item *n*, and the length of the path gives the total cost of following this policy. Then the shortest path from node *1* to node *T* on this network gives the optimal ordering policy for item *n* alone. An equivalent formulation will be to send an integer flow of 1 unit on this network from node *1* to node *T* at minimum cost. The constraints associated with this formulation then allow an integer programming formulation of the joint replenishment problem as shown in Fig. 2. The constraint sets (i) and (ii) are the network flow constraints, and ensure that each item node places orders from time period *1* to *T* in a feasible way to meet all demand. The constraint set (iii) are the nestedness constraints, and assures that if any item node places an order in any time period, then the joint cost is also incurred in that period. We will refer to this problem as problem (JRP).

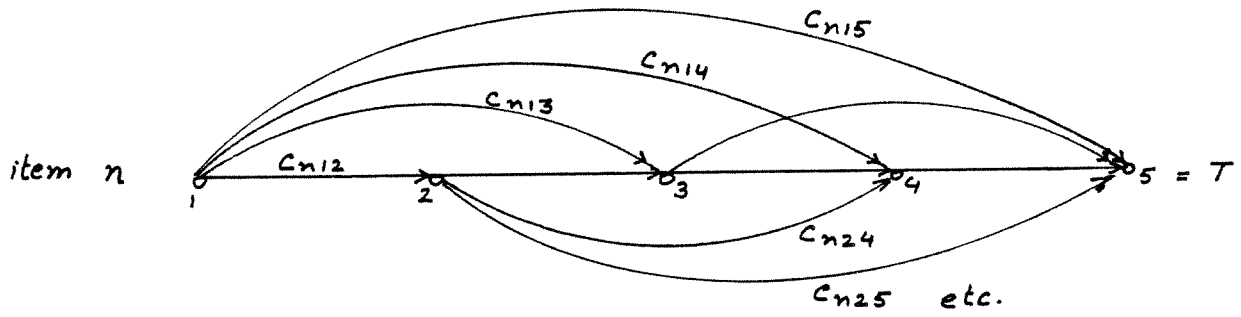

Fig. 2: The item-network for item n. There is one such network for each item.

$$\text{Min.} \quad \sum_{n=1}^{N} \sum_{s=1}^{T-1} \sum_{t=s+1}^{T} C_{nst} X_{nst} + \sum_{t=1}^{T} K_0 X_t$$

s.t.

$$\sum_{t>s} X_{nst} - \sum_{k<s} X_{nks} = 0 \qquad \forall\, n = 1 \ldots N, \ \forall\, s = 2 \ldots (T-1) \qquad \text{(i)}$$

$$\sum_{s>1} X_{n1s} = 1 \qquad \forall\, n = 1 \ldots N \qquad \text{(ii)}$$

$$X_t - \sum_{t>s} X_{nst} \geq 0 \qquad \forall\, n = 1 \ldots N, \ \forall\, s = 1 \ldots (T-1) \qquad \text{(iii)}$$

$$X_{nst},\ X_t = 0 \text{ or } 1$$

Fig. 3: Integer Programming Formulation of the Joint Replenishment Problem.

Consider the Lagrangean relaxation of problem (JRP) obtained by relaxing the nestedness constraints. Using Lagrangean multipliers λ_{ns} we get the relaxed problem (JRP_λ):

$$\text{Min. } v_\lambda = \sum_{n=1}^{N} \sum_{s=1}^{T-1} \sum_{t=s+1}^{T} (C_{nst} + \lambda_{ns}) X_{nst} + \sum_{s=1}^{T} (K_0 - \sum_{n=1}^{N} \lambda_{ns}) X_s$$

s.t. (i), (ii) above.

Clearly this problem (JRP_λ) decomposes by item *n* into N independent problems of the Wagner-Whitin type, with the cost on arc X_{nst} adjusted to $C_{nst} + \lambda_{ns}$. Solving the subproblems just involves finding the shortest path on the network for each node with these adjusted costs. The theory of Lagrangean duality tells that v_λ is a lower bound for problem (JRP). The tightest lower bound is then got by the Lagrangean dual problem (D):

$$(D): \qquad \text{Max} \{ v_\lambda \mid \lambda_{ns} \geq 0 \}$$

Further insight into the solution comes from the following proposition:

<u>Prop. 1</u> If $X_{nst} > 0$ in the solution to (D) then $K_0 - \sum_{n=1}^{N} \lambda_{ns} = 0$.

Proof: If $K_0 - \sum_{n=1}^{N} \lambda_{ns} > 0$ then in the solution to problem $(JRP)_\lambda$ we must have $X_s = 0$. Then $X_{nst} > 0$ means that increasing λ_{ns} will increase v_λ, which contradicts the fact that v_λ has been maximized in (D).

QED.

Thus if any node *n* places an order in time period *t*, then in that period the joint cost K_0 is fully shared among the items by way of the dual costs λ_{ns}.

4. Direction of the Proposed Research

In this paper we will present the results of preliminary computational results on the linear programming relaxation of our model. We will then discuss the structure of the basis in this

linear program. The idea will be to develop a specialized version of the simplex algorithm that solves this problem faster by exploiting its special structure. The understanding should also help in the design of heuristics to get a integer solution from the linear programming solution, and in the performance analysis of the heuristics.

The computational complexity of the joint replenishment problem has been investigated, and we have proved that this problem is NP complete (see Appendix A for details of the proof). An important goal of the research is to design fast and efficient heuristics for this problem.

Exact solution of the problem will be more difficult. One way to obtain an optimal solution is to use a branch and bound methodology. The Lagrangean relaxation discussed earlier will then provide a useful lower bound in the bounding strategy. A subgradient optimization procedure can be used to find a sequence of Lagrangean multipliers converging to the optimal solution to problem (D). Another way of getting a lower bound will be to use the linear programming relaxation of problem (JRP). The bound obviously will be the same as that obtained from Lagrangean relaxation. The actual implementation of the branch and bound algorithm has not yet been done since our current interest is more in understanding the structure of the dynamic joint replenishment problem. However, it is believed that the Lagrangean relaxation will yield tight bounds for the problem (JRP). This is based on the

linear programming relaxation of problem (JRP), which was implemented and solved on a random selection of problems. The results are discussed in the next section.

Our immediate efforts are directed towards understanding and exploiting the structure of the model. The motivation is to enable us not only to design efficient heuristics, but also to enable handling of joint cost structures in more general systems, including multi-echelon systems and where several families of products are involved. We are also interested in theoretical results on the quality of the heuristics developed, including worst case error bounds and probabilistic analysis.

5. Preliminary Computational Results

The linear programming relaxation of problem (JRP) was solved for a selection of randomly generated problems. A matrix generator was written in FORTRAN and interfaced with LINDO, which was then used to solve the problem. The problems generated had 4 items and a planning horizon of 20 time periods. The test problems had the following characteristics:

- Two joint setup costs (35 and 55) were used.
- Individual setup costs for the items were generated from a uniform (10,20) distribution.
- The demands for each item in each time period were generated from uniform distributions with specified mean and variance. The demand patterns used in the various problems varied from

constant demand to cases with high variance (up to a uniform (1,12) distribution).

A total of 156 test problems were generated and solved on an IBM personal computer.

Three important observations from the solutions were that:

1. In 76.9 percent of the problems, the optimal solution to the LP was integral. Thus it also solved problem (JRP) optimally.

2. In 12.2 percent of the problems, the optimal solution was not integral, but an integral optimal solution existed and could be easily constructed from the solution. Here again the LP solved the problem (JRP) optimally.

3. In the remaining 10.9 percent of the problems no optimal solution was integral. In each case one integral solution was constructed by observation by using the paths generated in the LP solution. In each case the integer solution differed from the LP solution by less than 0.6%, and by less than $0.15K_0$. Thus in each case the LP solution was within 0.15 joint costs (at worst) of the optimal solution to problem (JRP).

These results indicate that the linear programming relaxation is very close to the optimal integer solution, and actually provides an optimal integer solution in a majority of cases. The Lagrangean relaxation, which has the same objective function value as the LP relaxation, thus should provide us with very tight lower bounds to problem (JRP), and thus should be very effective in a branch and bound algorithm.

An analysis of the problems further suggests that instances in which the variability of demand from one time period to another is high mostly had integer solutions to the LP. Most of the non-integer solutions occurred in cases where the demand has low or zero variability. These appear to be the more "difficult" instances to handle. However, when demand is constant, Jackson, Maxwell and Muckstadt (1985) have an O(N logN) heuristic to solve the problem, whose worst case performance is within 2% of the optimal, and so the more "difficult" instances to handle are in fact very easy. Thus it is expected that the LP relaxation will be very close to optimal, and heuristics based on this should perform very well.

One problem in solving the LP was that the standard package we used (LINDO) took a very large number of iterations to solve the system. In our 156 test problems, each constraint matrix had 172 rows and 950 columns, with a density of 2.6%. The average number of pivots made was 2002, which is much larger than expected. Detailed analysis of the pivots in some problems showed that fewer than 100 of these were non-degenerate. We shall discuss later the possible causes of degeneracy. Besides, a large number (about 60 percent) of iterations were being used to get an initial feasible solution. However, we know that the solution in which $X_{n1T} = 1$ for all n (each node places only one order, in time period 1), is a feasible solution. Using this feasible starting solution should considerably reduce the number of iterations used by the simplex method.

6. Structure of the Basis

In this section we discuss the structure of the basis in the linear programming relaxation of problem (JRP). As shown earlier, the simplex algorithm has a large number of degenerate pivots in solving the LP. We would like then to develop a different algorithm along the lines of the simplex algorithm, which will move from one basis to another *making a non-degenerate pivot*. Unlike a simplex pivot however, these two bases may not be adjacent in order to avoid degeneracy. Thus at each step it will move from one cluster of bases to another. This will significantly speed up the solution of the LP. To this purpose, we need to understand the structure of the basis, and develop rules that show which basis to pivot to in order to have a strict improvement in the objective function. We have succeeded in defining the basis structure in cases where each item network has one or two basic paths from node 1 to node T. In these cases we can also show how a non-degenerate pivot can be made, and what the updated basis will be. We can further define the cases in which this pivot will lead to three basic paths on one of the item networks.

With N items, the constraint matrix of the LP is of full rank with N(2T-3) constraints. We convert the nestedness constraints to equalities by introducing non-negative slacks S_{nt}. This LP, as discussed before, can be interpreted as sending a unit flow from node 1 to node T on each item-network of the form shown in Fig. 2, subject to the nestedness constraints. Then on each item-network,

the flow constraints (i) and (ii) in Fig. 3 will yield a tree as a basis. The nestedness constraints will introduce further arcs into the basis. The unit flow will occur along paths made of basic arcs from node 1 to *T*. Besides, basic arcs that do not lie on a path from node 1 to *T* will have no flow. Clearly, if there is only one path on any network, then the flow on this path is unity, and so is integral.

Next, we introduce the concept of *link*. If S_{nt} and S_{mt} are nonbasic, then we get $\sum_{k>t} X_{ntk} = \sum_{k>t} X_{mtk} = X_t$, i.e. the net flow out of node *t* on networks *m* and *n* is forced to be equal. In the next pivot, unless one of these two slack variables is introduced as basic, the flows will continue to be equal. In this case we say that these two nodes have a link in between.

Some concepts of the structure of the basis are clarified by considering 2 items with 2 basic paths on the network for each item. It can then be shown that the basis consists of:

1. A tree on each network, and one extra arc on each network yielding the second path.
2. All the X_t's.
3. Either S_{1t} or S_{2t} for any (T-4) time periods.
4. There are 2 links between the 2 networks. The situation is depicted in Fig. 4. It is further possible to show that for this solution to be basic, the two links must be from each path on one network to the same path on the other network.

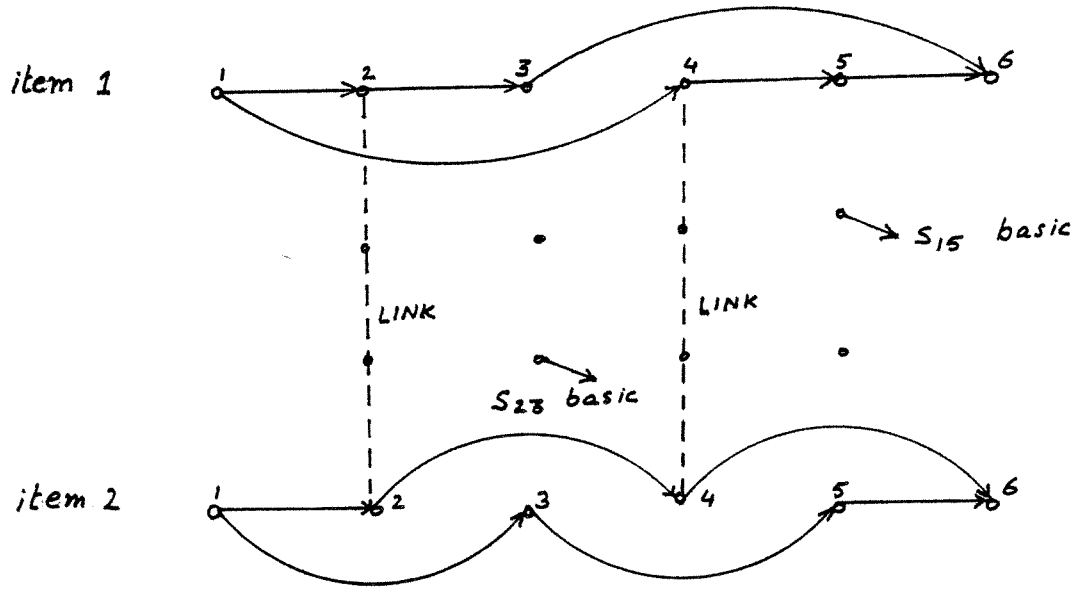

Fig. 4: The structure of the basis for 2 items, T = 6, 2 basic paths on each network. The nonbasic arcs are not shown.

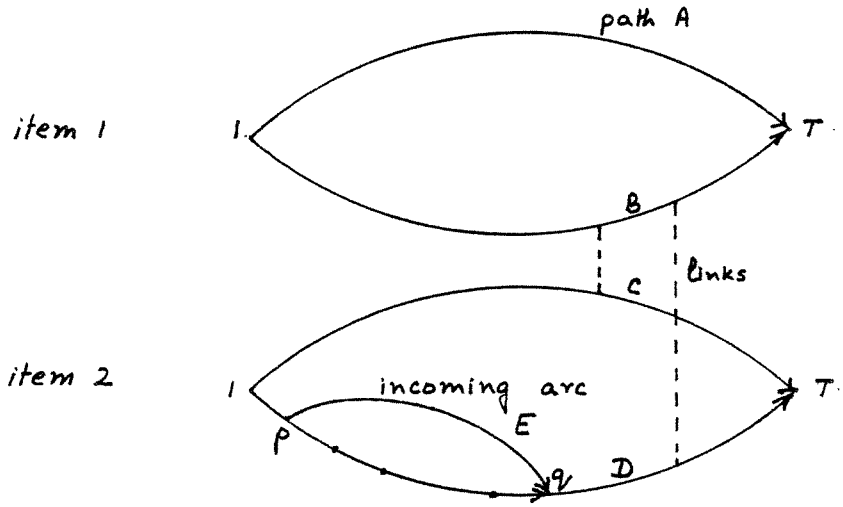

Fig. 5: Structure of a pivot. X_E is the flow on the incoming arc.

If X_A, X_B, X_C, and X_D are the flows on the 4 paths, then all the constraints clearly reduce to :

$$X_A + X_B = 1$$
$$X_A = X_C \qquad \text{(from the link)}$$
$$X_B = X_C \qquad (\quad ,, \quad)$$

The only possible solution to this is $X_A = X_B = X_C = X_D = 0.5$. Now let us pivot as shown in Fig. 5. Introduction of the new arc creates a new path on this network, with flow X_E. After the pivot, the new flows on the paths will be $X_B = X_C = X_D = 0.5$, $X_A = 0.5 - X_E$. For the pivot to be non-degenerate, it is then essential that S_{1t} for each t which path A touches between nodes p and q should be basis (then $S_{1t} = X_E$ after the pivot). If this condition is not satisfied, then X_E will enter the basis only at the zero level, and the pivot will be degenerate. Similarly, for other cases where different arcs are introduced into the basis, we can work out conditions about which S_{nt}'s must be basic in order for the pivot to be non-degenerate. It is then clear that any other combination of basic S_{nt}'s (which are quite possible form the 4 rules for a basis we have on the last page) will lead to a degenerate pivot. We believe this is the reason why the standard linear programming package we used in our computational experiments made a very large number of (degenerate) pivots. This analysis also suggests that given a set of paths and links, if we select an incoming arc for a pivot, then we could select a set of basic S_{nt}'s so that the next pivot is non-degenerate.

A similar set of rules defines the basis when there are more than 2 item nodes. Further, having selected a set of basic S_{nt}'s for a non-degenerate pivot, it is easy to set up the equations to define the flow changes when the incoming arc is introduced. This allows an easy update of the basis and determination of the new basic feasible solution by flow modification. We can also show from this that the new basis either has at most two basic paths from node 1 to T on each item-network, or at most one network has three basic paths.

7. Conclusions and Extensions

The joint replenishment problem is the first step towards investigating the more general problem having several echelons in the production or distribution system. Such models are highly relevant to logistics control in large manufacturing and distribution environments. These systems are very complex, with no known efficient optimal solution procedures. We expect the understanding gained from the simpler problems to be useful in designing efficient heuristics for the more complex models.

We have also so far considered all items as belonging to the same family, incurring the same joint cost. A further extension is to have families of items, and a joint cost associated with each family. Besides, an item may belong to several families. For example, in a grocery store, meat and produce can be two families. One way of handling this is by having a set of nestedness

constraints on the integer programming formulation corresponding to each family. We expect to gain insights from the joint replenishment problem to efficiently handle these constraints.

The joint cost structure we have considered has a fixed joint cost whenever any item in a family is ordered. In several situations it is more natural that the ordering cost is a function of the set of items being ordered jointly. In the infinite horizon case with costs that are a concave function of the number of items being jointly replenished, this problem has been solved by Chakravarty, Orlin and Rothblum. It would be of interest to extend our model to include general joint cost functions.

References

Chakravarty,A.K., J.B.Orlin and U.G.Rothblum, "A Partitioning Problem with an Application to Optimal Inventory Groupings For Joint Replenishment," *Operations Research,* **30,** 5, pp. 1018-1022 (1982).

Garey,M.R. and D.S.Johnson, "Computers and Intractability," New York: W.H.Freeman and Co. (1979).

Goyal,S.K., "Determination of Optimum Packaging Frequency of Items Jointly Replenished," *Management Science,* **21,** 4, pp. 436-443 (1974).

Jackson,P., W.L.Maxwell and J.A.Muckstadt, "The Joint Replenishment Problem with a Power-of-Two Restriction," *IIE Trans.,* **17,** 1, pp. 25-32 (1985).

Kao,E.P.C., "A Multi-Product Dynamic Lot-Size Model with Individual and Joint Set-up Costs," *Operations Research,* **27,** 2, pp.279-289 (1979).

Roundy,R.O., "A 98% Effective Lot Sizing Rule for a Multi-Product Multi-Stage Production Inventory System," *T.R. 642, School of OR&IE, Cornell University (1984).*

Silver,E.A., "Coordinated Replenishments of Items Under Time-Varying Demand: Dynamic Programming Formulation," *Naval Res. Logistics Quarterly,* **26,** pp. 141-151 (1979).

Vienott,A.F., "Minimum Concave Cost Solution of Leontief Substitution Models of Multi-Facility Inventory Systems," *Operations Research,* **17,** pp. 262-291 (1969).

Zangwill,W.I., "A Deterministic Multi-Product Multi-Facility Production and Inventory Model," *Operations Reserach,* **14,** pp. 486-507 (1966).

APPENDIX A

Proof of NP Completeness of the Joint Replenishment problem

(Joint work with E.Arkin and R.Roundy)

NP-Completeness Proof

This is a note to prove the following *Joint Replenishment Problem* (JRP) NP-complete:

Instance: A set of N products, $n = 1, \ldots, N$, and T time periods, $t = 1, \ldots, T$, inventory holding costs h_n, $n = 1, \ldots, N$, ordering costs K_n, $n = 1, \ldots, N$, demands d_{nt}, $n = 1, \ldots, N$, $t = 1, \ldots, T$, a joint ordering cost K_0 and an integer B. Create the following graph: For each product we have nodes (n, t) for all t. Arcs from (n, s) to (n, t) exist for all $s < t$ with cost $c_{nst} = h_n \sum_{i=1}^{t-s}(t - s - i)d_{t-i} + K_n$. The joint setup cost K_0 is incurred for each t where a node (n, t) is used in the solution paths for any n.

Question: Does there exist a set of paths from nodes $(n, 1)$ to (n, T) for all n for which the sum of the setup costs incurred plus the path costs is not greater than B?

Theorem *The above problem is NP-complete.*

Proof: It is easy to see that the problem is in NP. In fact, algorithms are known which run in polynomial time for fixed T or for fixed N.

To show completeness, we give a reduction from 1-in-3 SAT. (For definitions, see Garey and Johnson, 1979.) Let the variables be $x_1, \ldots, x_n$, and the clauses be $C_1, \ldots, C_m$. We define an instance of (JRP) with $T = M + 2n + 2$ periods and $N = m + n$ products. The periods correspond to: A special period 1 at the beginning, M dummy periods after it, two periods corresponding to each variable x_i, one for x_i and one for the negation of x_i, $\overline{x}_i$, and a special period N at the end. The first n products correspond to variables x_i, and the remaining m products correspond to clauses C_j.

For the x_i products, the cost of a path from node 1 to x_i to N is 2, and the cost of the path from node 1 to $\overline{x}_i$ to N is 2. Any other path is of length greater than or equal to $2 + K_0$. For the C_j products, the cost of a path from node 1 to L_{ij} to N is 2, and the cost of any other path is greater than 2. (L_{ij} are the literals that appear in clause C_j.) We show later how to create a set of demands and costs to fit this pattern. We set $B = 2(m + n) + nK_0$.

We now show that the above-defined instance of (JRP) has the desired set of paths if and only if the corresponding 1-in-3 SAT instance is satisfiable. First, note that if the formula is satisfiable, then the following set of paths has cost B: For each product x_i, go from 1 to either x_i or $\overline{x}_i$ (according to which is true in the satisfying assignment), and from there to N. The cost incurred will be $2n + nK_0$. For each product C_j, go from 1 to the literal that is true in clause C_j, and from there to N. The cost will be $2m$ with no additional setup costs. Clearly, the total length will be $2(m + n)$, and the setup cost will be nK_0.

Now assume that there is a set of paths from 1 to N for each product, such that the sum of path lengths and setup costs is no greater than B. Let p of the x_i products use the path from 1 to either x_i or $\overline{x}_i$ and then to N at cost $2p + pK_0$. The remaining $(n - p)$ x_i products choose any other path, thus possibly saving on the joint setup cost. These will cost at least $(n - p)(2 + K_0)$. Finally, the C_j products will cost at least $2m$. Hence the total cost is greater than or equal to $2p + pK_0 + (n - p)(p + K_0) + 2m$, which is equal to $(m + n)2 + nK_0$ which is B. In fact equality hold only if the C_j products choose a path from node 1 to L_{ij}

to N with a cost 2 for which we have already incurred the setup costs. Clearly, this results in a satisfying truth assignment.

Finally, it remains to be seen that we can indeed generate demands and cost, so that the paths have the lengths described above. For the x_i products let

$$\begin{aligned} d_{x_i} &\stackrel{\text{def}}{=} \frac{K_0}{M+2(i-1)+1}, \\ d_{\overline{x}_i} &\stackrel{\text{def}}{=} K_0, \\ K_i &\stackrel{\text{def}}{=} 2K_0, \\ \text{and} \quad h_i &\stackrel{\text{def}}{=} 1. \end{aligned}$$

The cost of a path from node 1 to x_i to N is $2K_i + d_{\overline{x}_i} = 5K_0$. The cost of a path from node 1 to $\overline{x}_i$ to N is $2K_i + d_{x_i}(M + 2(i-1) + 1) = 5K_0$. The cost of a path from node 1 to t to N for $t < x_i$ is greater than $2K_i + d_{x_i} + 2d_{\overline{x}_i} > 6K_0$, and for $t > x_i$ the cost is greater than the cost of a path from node 1 to N. The cost of the path from node 1 to node N is $K_i + (M + (2(i-1)+1)d_{x_i} + (M+2i)d_{\overline{x}_i} + 3K_0 + (M+2i)K_0 \geq 6K_0$ for $M \geq 1$. This covers all possibilities where product x_i is ordered once or twice. If it is ordered at least three times, the cost of the path will be at least $3K_i = 6K_0$. Setting $K_0 \stackrel{\text{def}}{=} 0.4$ yields the required demand structure. Non integer demands are easliy removed, if necessary, by using a suitable holding cost h_i such as:

$$\begin{aligned} d_{x_i} &\stackrel{\text{def}}{=} 5K_0 = 2, \\ d_{\overline{x}_i} &\stackrel{\text{def}}{=} 5(M + 2(i-1)+1)K_0 = 2(M+2(i-1)+1), \\ \text{and} \quad h_i &\stackrel{\text{def}}{=} \frac{1}{5(M+2(i-1)+1)}. \end{aligned}$$

For clause products let p be the number of time periods from 1 to L_{i1}, q be the number of time periods between L_{i1} and L_{i2}, and r be the number of time periods between L_{i2} and L_{i3}. (Notice that p, q, r depend on the clause, but we omit the subscripts and hope that no confusion arises as a result.) Let

$$\begin{aligned} d_{i1} &\stackrel{\text{def}}{=} (p+q+r)q, \\ d_{i2} &\stackrel{\text{def}}{=} rp, \\ d_{i3} &\stackrel{\text{def}}{=} p(p+q), \\ K_i &\stackrel{\text{def}}{=} 2K_0 = 0.8, \\ \text{and} \quad h_i &\stackrel{\text{def}}{=} \frac{0.4}{p(pq+q^2+pr+2qr)}. \end{aligned}$$

The cost of a path from node 1 to L_{i1} to N is

$$\begin{aligned} & 2(0.8) + qh_id_{i2} + (q+r)h_id_{i3} \\ =& 1.6 + \frac{0.4qr + 0.4(q+r)(p+q)}{pq+q^2+pr+2qr} \\ =& 2 \end{aligned}$$

The cost of a path from node 1 to L_{i2} to N is

$$\begin{aligned} & 2(0.8) + ph_i d_{i1} + rh_i d_{i3} \\ =& 1.6 + \frac{0.4q(p+q+r) + 0.4r(p+q)}{pq + q^2 + pr + 2qr} \\ =& 2 \end{aligned}$$

The cost of a path from node 1 to L_{i3} to N is

$$\begin{aligned} & 2(0.8) + ph_i d_{i1} + (p+q)h_i d_{i2} \\ =& 1.6 + \frac{0.4q(p+q+r) + 0.4r(p+q)}{pq + q^2 + pr + 2qr} \\ =& 2 \end{aligned}$$

The cost of a path from node 1 to $t \neq L_{ij}$ to N is clearly greater than 2. The cost of a path from node 1 to N is

$$\begin{aligned} & 0.8 + ph_i d_{i1} + (p+q)h_i d_{i2} + (p+q+r)h_i d_{i3} \\ =& 0.8 + \frac{0.4q(p+q+r) + 0.4r(p+q) + 0.4(p+q)(p+q+r)}{pq + q^2 + pr + 2qr} \\ =& 1.2 + \frac{0.4(p+q)(p+q+r)}{pq + q^2 + pr + 2qr} \end{aligned}$$

Since we require this last number to be greater than 2, we require

$$p^2 + 2pq + pr + q^2 + qr > 2Q^2 + 2pq + 2pr + 4qr$$

$$\text{or} \quad p^2 - pr > q^2 + 3qr.$$

This is where the M dummy time periods are used. Note that $p \geq M + 1$ and $q + r \leq 2n$, so the inequality always has a solution. To find it we note that

$$\begin{aligned} q^2 + 3qr &= q(q+r) + 2qr \\ &\leq 2qn + 2qr \\ &\leq 2q(3n - q). \end{aligned}$$

This quadratic form is maximized at $q = \frac{3}{2}n$, and we have

$$q^2 + 3qr \leq \frac{9}{2}n^2.$$

Again, using the fact that $r \leq 2n$ we have $p^2 - pr \geq p^2 - 2pn$, so it suffices to have $p^2 - 2pn > \frac{9}{2}n^2$. Equality holds for $p = \frac{2n \pm \sqrt{4n^2 + 18n^2}}{2}$, so for our purposes we can set $M \stackrel{\text{def}}{=} \lceil (1 + \frac{\sqrt{22}}{2})n \rceil$.

For a path that contains three or more orders, the cost is greater than $3K_i = 2.4 > 2$ as required. It is easy to see that this construction can be carried out in polynomial time, thus our reduction is valid. ∎