Inventory Lot Sizing with Supplier Selection

Chuda Basnet
Department of Management Systems
The University of Waikato, Private Bag 3105
Hamilton, New Zealand
chuda@waikato.ac.nz

Janny M.Y. Leung
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong, Shatin, N.T.
Hong Kong
janny@se.cuhk.edu.hk

Abstract

This paper presents a multi-period inventory lot-sizing scenario, where there are multiple products and multiple suppliers. We consider a situation where the demand of multiple discrete products is known over a planning horizon. Each of these products can be sourced from a set of approved suppliers, a supplier-dependent transaction cost applying for each period in which an order is placed on a supplier. A product-dependent holding cost per period applies for each product in the inventory that is carried across a period in the planning horizon. The decision maker needs to decide what products to order in what quantities with which suppliers in which periods. An enumerative search algorithm and two heuristics are presented to address this decision.

1 Introduction

The single product, multi-period inventory lot sizing model is well known in the production and inventory management literature. The definitive work on this model was done by Wagner and Whitin (1958), who presented a dynamic programming solution algorithm. An obvious extension of this model is to extend it to multiple products and multiple suppliers. This paper presents such an extended model. We consider a situation where the demand of multiple discrete products is known over a planning horizon. Each of these products can be sourced from a set of approved suppliers, a supplier-dependent transaction cost (or ordering cost) applying for each period in which an order is placed on a supplier. A product-dependent holding cost per period applies for each product in the inventory that is carried across a period in the planning horizon. The decision maker needs to decide what products to order in what quantities with which suppliers in which periods.

Bahl et al. (1987) provide a comprehensive review of inventory lot sizing literature. They provide four categories for classifying work in this area: 1) single level unconstrained resources, 2) single-level constrained resources, 3) multiple level constrained resources, and 4) multiple level unconstrained resources. Levels here refer to the different levels in a bill of material structure where dependency of requirements

exists, and constrained resources refer to production capacity limitations. The scenario discussed in this paper belongs to the first category, since we do not consider level dependencies and capacity limitations. Extensive work has been done in this area, starting from the work of Harris (1915), and is continuing. Wagner and Whitin (1958) provided a dynamic programming algorithm for this problem. Many authors (Aggarwal and Park, 1993; Federgruen and Tzur, 1991) have extended the Wagner-Whitin model and presented algorithms to solve the models. Recent work on lot sizing have added new features to the problem such as capacity limits (Shaw and Wagelmans, 1998), time windows for demand satisfaction (Lee et al., 2001), random supplier capacity (Hariga and Haouari, 1999), overtime decisions (Ozdamar and Bozyel, 2000), and carrying over of setups (Sox and Gao 1999). A number of authors have worked also on improving the performance of the Wagner-Whitin algorithm by finding efficient data structures and dominance rules to reduce the search space (Evans, 1985; Jacobs and Khumawala, 1987; Saydam and McKnew, 1987; Heady and Zhu, 1994).

With the advent of supply chain management, much attention is now devoted to supplier selection. Ganeshan (1999) has presented a model for determining lot-sizes that involves multiple suppliers while considering multiple retailers, and consequent demand on a warehouse. In a similar vein, Jayaraman et al. (1999) have proposed a supplier selection model that considers quality (in terms of proportion defectives supplied by a supplier), production capacity (this limits the order placed on a supplier), lead time, and storage capacity limits. This is also a single period model that attaches a fixed cost to dealing with a supplier. Jayaraman et al. (1999) formulate a mixed integer linear programming model to solve the problem.

To our knowledge, there is no multi-period supplier selection model presented in the literature. This paper considers demand of multiple products in multiple periods: one or more supplier could be selected in each of these periods for the purchase of products. The model presented in this paper thus attempts to bridge the gap between the earlier multiple period discrete inventory lot-sizing models and the newer supplier selection models.

2 Multi-period lot sizing with supplier selection problem (MLSSP)

We formulate the multi-product multi-period lot-sizing with supplier selection problem using the following notation.

Indices:

$i = 1 \ldots I$ index of products (inventory items).
$j = 1 \ldots J$ index of suppliers.
$t = 1 \ldots T$ index of time periods.

Parameters:

D_{it} = Demand of product i in period t.
P_{ij} = Purchase price of product i from supplier j.
H_i = Holding cost of product i per period.
O_j = Transaction cost for supplier j.

Decision variables:

X_{ijt} = Number of product i ordered from supplier j in period t.
Y_{jt} = 1 if an order is placed on supplier j in time period t.

Intermediate variable:

R_{it} = Inventory of product i, carried over from period t to period $t + 1$.

With the above notations, the MLSSP may be stated as follows.

$$Min \sum_t \sum_j \sum_i P_{ij} X_{ijt} + \sum_j \sum_t O_j Y_{jt} + \sum_i \sum_t H_i \left(\sum_{k=1}^{t} \sum_j X_{ijk} - \sum_{k=1}^{t} D_{ik} \right)$$

s.t.

$$R_{it} = \sum_{k=1}^{t} \sum_j X_{ijk} - \sum_{k=1}^{t} D_{ik} \geq 0, \text{ for all } i \text{ and } t$$

$$\left(\sum_{k=t}^{T} D_{ik} \right) Y_{jt} - X_{ijt} \geq 0, \text{ for all } i,\ j, \text{ and } t$$

$$Y_{jt} = 0 \text{ or } 1, \text{ for all } j \text{ and } t$$

$$X_{ijt} \geq 0, \text{ for all } i,\ j, \text{ and } t$$

This is a mixed integer programming (MIP) formulation of the MLSSP. The objective function consists of three parts: 1) the purchase cost of the products, 2) the transaction cost for the suppliers, and 3) the holding cost for remaining inventory in each period. The first constraint stipulates that all requirements must be filled in the period in which they occur: shortage or backordering is not allowed. The second constraint says that one cannot have an order without charging an appropriate transaction cost.

If there is only one supplier to be considered, the price of the products can be ignored, and all the theorems and assumptions of Wagner and Whitin (1958) would still apply for the multi-product case, except we are dealing with a vector of inventories, rather than a single product as discussed by Wagner-Whitin. In particular, the optimal ordering programme can be found by forward recursive dynamic programming formulation as shown by Wagner-Whitin, and the planning horizon theorem also applies which reduces the recursion involved.

Wagner-Whitin have shown for the single supplier case that it is sufficient to consider only solutions that do not involve carrying inventory into a period simultaneously with ordering inventory in that period, that is, $R_{i,t-1} X_{ijt} = 0$, for all i, t and j. This theorem applies to the current problem, for the same reasons as cited by Wagner-Whitin, and the theorem can be extended further in the multi-supplier problem to say that for any given i and t, $X_{ijt}\ X_{ikt} = 0$, (for all j and k; $j \neq k$), that is one needs to consider only the solutions where in a given period there can be only one supplier for a particular product. If there are more than one supplier for the same product for the same period in an optimal solution, it should not increase the total cost when all the purchase quantities are assigned entirely to the cheaper (or equal) supplier.

In this paper we explore some properties of this problem and develop an enumerative solution procedure using a bound and some cuts. This search algorithm proved quite time consuming for practical sized problems, so we have also developed two heuristics to solve the problem within reasonable time.

2.1 Numerical example

Consider three products A, B, and C, whose requirements are given by the demand matrix D_{it} [D_{1t} = {12, 15, 17, 20, 13}, demand of A over a planning horizon of 5

periods; D_{2t} = {20, 21, 22, 23, 24}; D_{3t} = {20, 19, 18, 17, 16}]. There are three suppliers, α, β, and γ ; and their price matrix is P_{ij} [P_{1j} = {30, 33, 32}, price of product A charged by the three suppliers; P_{2j} = {32, 35, 40}; P_{3j} = {45, 43, 45}]. The holding cost vector H_i is (1, 2, 3), i.e., the cost of holding one unit of A over one period is 1. The transaction cost vector O_i is (110, 80, 102), i.e., the transaction cost for placing an order with α is 110.

One feasible solution of this problem is to place the following orders. Order 44 of product 1 from supplier 1 in period 1, X_{111} = 44. X_{231} = 20. X_{331} = 20. X_{232} = 43. X_{332} = 37. X_{134} = 33. X_{234} = 47. X_{334} = 33. All other X_{ijt} = 0.

Cost calculation for this solution is as follows: Number of orders from supplier 1 = 1 (in period 1). Transaction cost = 1× 110 = 110. Purchase cost for product 1 = 44 × 30 = 1320. There are no orders from supplier 2. Number of orders from supplier 3 = 3 (in periods 1, 2, and 4). Transaction cost = 3× 102 = 306. Purchase cost for product 1 = 33 × 32 = 1056. Purchase cost for product 2 = (20 + 43 + 47) × 30 = 3300. Purchase cost for product 3 = (20 + 37 + 33) × 45 = 4050.

The carried-over inventory matrix is R_{it} [R_{1t} = {32, 17, 0, 13, 0}; R_{2t} = {0, 22, 0, 24, 0}; R_{3t} = {0, 18, 0, 16, 0}]. The first entry R_{11} represents $X_{11} - D_{11}$ = 44 – 12 = 32 etc. Thus holding cost for product 1 = $H_1 \sum R_{1t}$ = 1 × (32 + 17 + 0 + 13 + 0) = 62; holding cost for product 2 = $H_2 \sum R_{2t}$ = 2 × (0 + 22 + 0 + 24 + 0) = 92; holding cost for product 3 = $H_3 \sum R_{3t}$ = 3 × (0 + 18 + 0 + 16 + 0) = 102.

Thus the total cost for this solution = 110 + 1320 + 306 + 1056 + 3300 + 4050 + 62 + 92 + 102 = 10398.

3 Enumerative algorithm

In this section we present an enumerative algorithm for this problem. In this algorithm, the X_{ijt} decisions are all made in chronological order. Each node represents a time period, and at each node, the algorithm determines the earliest epoch at which some product requirements are not met: the branches are sprouted from this node, each branch representing one feasible combinations of orders that can be placed at that point in time. When all the time periods are exhausted, one possible solution has been arrived at, and it is compared with any previous solution and the best solution is saved. At each node, a lower bound of the possible total cost at that node is calculated, and if it is more that the previous best solution objective, the node is fathomed, and search continues with the last unresolved node (this is a depth-first search tree). The search tree is pruned also by using some observations given later.

A partial search tree, for the numerical example presented earlier, is shown in Figure 1. Let $\boldsymbol{U}$ represent the vector of time periods (u_1, u_2, … u_I) up to which the requirements of product 1, 2, …I are exactly covered, with no remaining inventory to carry over at that time period, i.e., at any stage of the search u_1 represents the period up to which the demand of product 1 is fully met . The root node of the search tree is thus represented by the vector $\boldsymbol{U}$ = (0, 0, …0), no products ordered, no requirements met. At this node, orders need to be placed for each of the products for time period 1. For each product, the orders may be placed for any period between 1 and T, with any supplier 1 and J. Thus the number of branches sprouted from the root node is $T^I J^I$. Each branch from this root node represents a X_{ij1} matrix (a set of order decisions for the period 1).

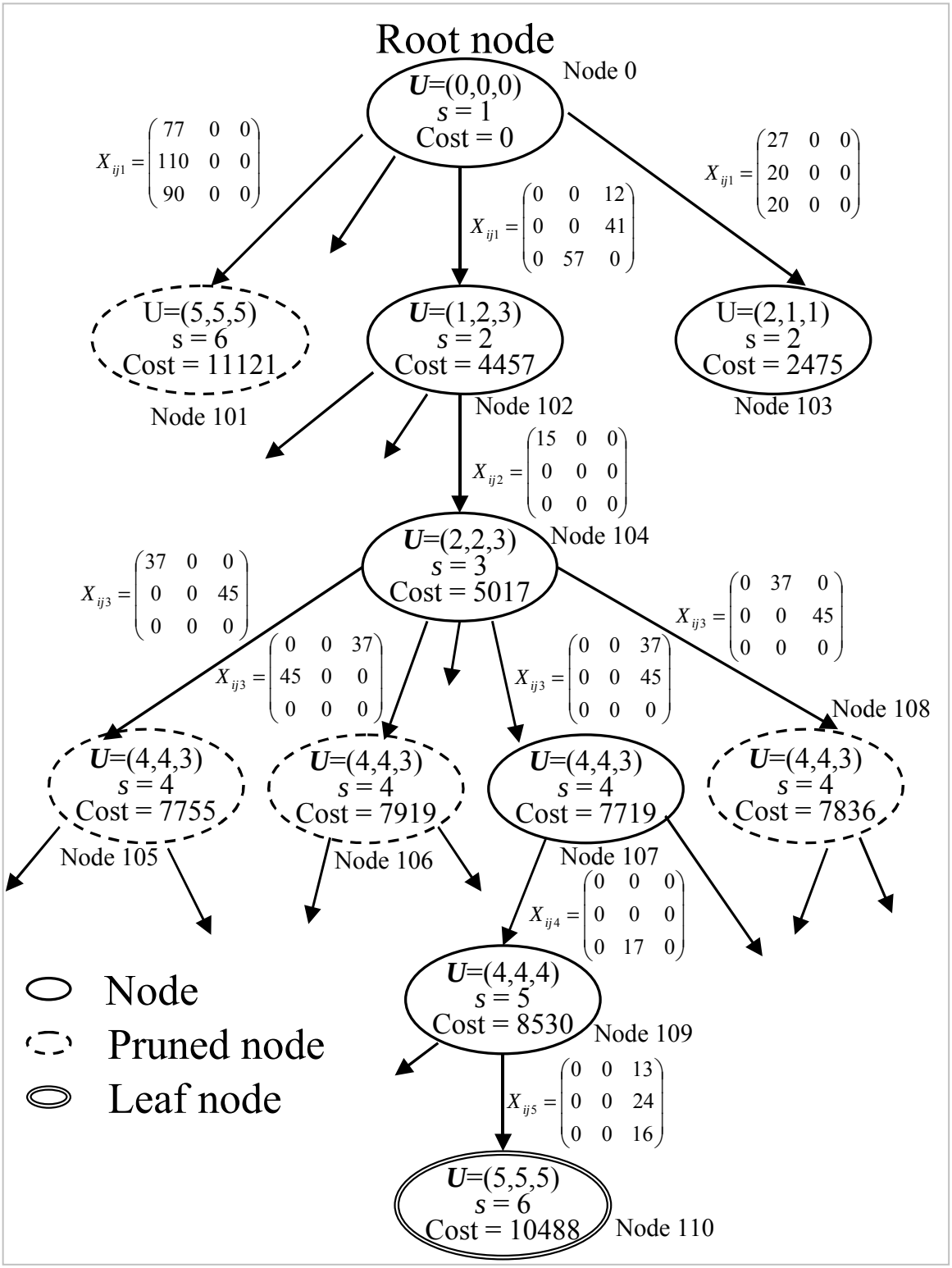

Figure 1. A partial search tree

At any future node, the coverage period vector $\boldsymbol{U} > \mathbf{0}$, and the time period $s = 1 + \min(u_1, u_2, \ldots u_I)$ is the period for which demand of some products is not covered ($u_i = s - 1$ for these products). Each branch from this node represents the decision matrix X_{ijs}. ($\sum_i X_{ijs} \geq D_{is}$, for all i such that $u_i = s - 1$; $X_{ijs} = 0$ otherwise; in other words X_{ijs} indicate which suppliers are chosen for the demand for those products that must be covered for the current period s and the corresponding lot-size at the current node). When $s = T + 1$, all the requirements are covered, the search has arrived at a leaf node with no more branching. The cost associated with each node is the *nodal cost* incurred so far to cover the demand for all the products up to the time period vector U:

$$\sum_j \sum_i \sum_{t=1}^{s-1} P_{ij} X_{ijt} + \sum_j \sum_{t=1}^{s-1} O_j Y_{jt} + \sum_i \sum_{t=1}^{u_i} H_i \left(\sum_{k=1}^{t} \sum_j X_{ijk} - \sum_{k=1}^{t} D_{ik} \right)$$

3.1 A Lower Bound

It should be clear from the above discussion that each node represents a vector ***U*** of time periods up to which the requirements are covered. To calculate a lower bound on the optimal cost at this node, one can imagine a single hypothetical supplier with index *k*, whose ordering cost and prices are minimal among all the suppliers. That is, O_k = min (O_j), and P_{ik} = min (P_{ij}) for each *i*. The cost of covering all the demand from *s* onwards, until *T*, using such a single supplier is obviously a lower bound on the further cost at a node. Furthermore, as pointed out before, such a cost for a single supplier can be found by using an efficient version of the Wagner-Whitin algorithm.

3.2 Pruning the search tree

Some rules for pruning the search tree are presented below.

Rule 1. As discussed above, each node represents the decisions at the time period $s = 1 + \min (u_1, u_2, \ldots u_I)$, at which time the demand for one or more products are not covered. The sprouts from this node are the different feasible combinations of suppliers and the periods up to which the demand is to be covered. Each product that needs to be ordered at this juncture ($t = s$) could be ordered in enough amounts to cover the demand up to period s, $s + 1$, $s + 2$, … or T periods. Each of these orders could be placed on one of J suppliers. Thus if the number of products at this decision point is n, $n = |\{i: u_i = s - 1\}|$, the number of possible nodes to be sprouted from this node for a complete enumeration is $(T - s + 1)^n J^n$.

However, the future costs associated with a node with coverage period ***U*** are independent of the path followed to get to this node. Thus if there are a number of branches that start from a node with coverage period ***U*** and result in different nodes with the same coverage period ***V***, one needs to sprout only one of these nodes for this ***V***, the one with the least nodal cost. The other nodes may be pruned. In other words, all the sprouted nodes should have a unique ***V***. It may be pointed out that the ***V*** vector is different from ***U*** only in those elements where u_i=s-1 (the number of products ordered could in fact be much smaller than *I)*, and the optimisation problem is an uncapacitated plant location problem.

In Figure 1, four branches are shown stemming from node 104, all of which lead to the coverage period (4, 4, 3). Actually nine such branches are possible. However node 107 has the lowest nodal cost of 7719, and the other branches may be pruned.

Rule 2. The above rule establishes only the need to conduct a combinatorial search at each node in order to prune the tree. It does not indicate how this combinatorial search may be carried out. Each ***V*** entails purchasing a set of products for which there are a number of candidate supplier sets. Each product that needs to be ordered at this juncture ($t = s$) could be ordered on one of J suppliers. Thus if the number of products at this decision point is n, $n = |\{k: u_k = s - 1\}|$, the supplier sets could be of cardinality 1, 2, min (n, J), and the total number of possible supplier sets is:

$$\binom{J}{1} + \binom{J}{2} + ... + \binom{J}{Min(n, J)}$$

However, for a given product set, not all of these supplier sets need to be enumerated since some sets may dominate others if one set has lower or equal total ordering costs, and lower or equal purchase prices for all products.

In Figure 1, consider branching from the node 103, where orders need to be placed in period $s = 2$ for the product set {2, 3} since $u_2 = u_3 = 1$. The total number of possible supplier sets is six. Of these, comparing the two supplier sets {1}, and {3}, the ordering cost for the first set is 110, which is dominated by that for the second set, which is 102. The purchasing price vector for the first supplier set is (32, 45), which is dominated by the purchasing price vector (30, 45) for the second supplier set. Thus in considering the product set {2, 3} the first supplier set, i.e. {1}, may be ignored.

Rule 3. However, given one set of products, and one set of candidate suppliers, there is no need to look at all the combinations of products and suppliers: there is only one least cost allocation of orders - for each product select the supplier in the supplier set that has the least price. This selection can be done prior to the search.

In Figure 1, branching from node 104, the two nodes 105 and 106 both refer to the ordering of product set {1, 2} with the same supplier set {1, 3}. Of these, the price of product 1 is lower with supplier 1, and the price of product 2 is lower with the supplier 3. Thus given this product-supplier set combination, only the node 102 needs to be investigated. For each product set and each of its non-dominated supplier sets, an order allocation table can be established based on prices, prior to the search.

Rules #2 and #3 may be applied together, and a look-up table of non-dominated supplier sets with order allocation for each product set can be created prior to the search.

Rule 4. When the product set has only one product, then the sum of inventory purchase and ordering costs for all the suppliers are of the form $P_{ij} X + O_j$ where X is the inventory requirement to be covered. We are looking for

$$j^* = \arg\min_j [P_{ij} X + O_j]$$

We can locate the intersection points formed by the j lines on a x-y plane and create a look-up table which gives j^* for all X-values for each product. Considering node 102 in Figure 1, only product 1 needs to be ordered in period 2. The sum of purchase and ordering costs for the three suppliers are $30X + 110$, $33X + 80$, $32X + 102$ respectively. Solving for the intersection of these lines, orders need to be placed with supplier 2 for quantities up to 10, and with supplier 1 for quantities more than 10.

Rule 5. For a product combination of cardinality n, ($n \leq J$), the supplier set formed by referring to the look-up table in # 4, above,

$$S^* = \{ j_i^* \mid j_i^* = \arg\min_j [P_{ij} X_i + O_j] \quad \text{for each } i \}$$

dominates all supplier sets whose cardinality is n. This is obvious since each product is supplied by a distinct supplier in this case, and the total of purchase and order cost for this supplier cannot be lower than the one found from # 4 above.

Rule 6. Another dominance rule to prune the search tree is the following. At any node with inventory coverage period $\boldsymbol{U}$, as above, one of the branching decisions is how much future demand coverage to purchase. If a product i has $u_i = s - 1$, this product needs to be ordered. The inventory coverage choices are up to period s, $s + 1$, $s + 2$, … or T .

Following Heady and Zhu (1994), if for this product and for a coverage period *t*, the following condition applies,

$$H_i D_{it} (t - u_i - 1) > O_m \qquad m = j \mid MinP_{ij}$$

then inventory coverage of period *t* and beyond need not be sprouted. In other words, if the cost of carrying inventory for period *t* is above the ordering cost of the minimum priced supplier, an order can be placed in period *t* instead of carrying inventory into period *t* at no additional cost.

As discussed earlier, in Figure 1, the ordering decisions at the root node concern products 1, 2, and 3. Product 3 could be ordered to cover demand of up to periods 1, 2, 3, 4, or 5. In other words, the ordering choices are in quantities of 20, 39, 57, 74, or 90. The third of these choices entails carrying an inventory of 18 products from period 1 until period 3. The carrying cost for this product alone is

$$3 \times 18 \times (3 - 0 - 1) = 108 > O_2 = 80$$

Now this carrying cost is higher than the ordering cost of the cheapest supplier, supplier 2. Thus instead of carrying this inventory, it could be ordered in period 3 with this supplier. Hence for this product, at this node, one needs to consider only ordering quantities 20 and 39. The ordering quantities 57, 74 and 90 are dominated, and node 101 is pruned for this reason.

4 Heuristics

The enumerative search algorithm above is still computationally expensive. The biggest problem for which we could find solutions has $I = 4$, $J = 4$, and $T = 10$. A simple heuristic for the MLSSP may be constructed based on the Wagner-Whitin algorithm. Since very efficient versions of this algorithm are available (e.g. Heady and Zhu,1994, Shaw and Wagelmans, 1998), the algorithm can be used multiple times. Two heuristics were constructed and tested.

4.1 Heuristic 1 (H1)

This heuristic is based on an efficient version of the Wagner-Whitin algorithm provided by Heady and Zhu (1994). This heuristic consists of two phases:

1. Assigning products to suppliers. In this step each product is assigned to a unique supplier for the entire duration of the planning horizon. The lot sizing of the products assigned to each supplier then becomes a multi-item, single supplier problem, and the Wagner-Whitin algorithm is applicable. Each product is considered in turn, and the increased cost of assigning this product (in addition to any previous assignments) to each supplier is calculated. It is then assigned to the supplier with the least cost increase. This continues until all the products are assigned.
2. Improvement moves. In the above step, we have a set (possibly null) of products assigned to each supplier for the entire planning horizon. The total cost includes orders placed on the supplier, the purchase cost, and the holding costs for inventories. In the current step each inventory item at any period (carried forward or purchased) is investigated to see if it can be combined with another existing order for the same period on another supplier, thus reducing the total cost. If such moves are found, the inventory item is moved to the supplier with the largest saving.

4.2 Heuristic 2 (H2)

This heuristic is based on the capacitated lot sizing algorithm for a single item provided by Shaw and Wagelmans (1998), and consists of two phases:

1. Lot-sizing construction phase: Assigning products to suppliers. In this phase each product is considered in turn, and the optimal lot-sizing schedule for each supplier is calculated, based on Shaw and Wagelmans (1998). The product is then assigned to the supplier with the overall least cost for the entire duration of the planning horizon.
2. Improvement Phase. The solution obtained from the Construction Phase gives a solution where each product is sourced from a single supplier for the entire horizon. Furthermore, if two products are ordered from the same supplier j in one period, two transaction costs O_j are assessed. At first, the order schedule for each product is retained, but for each period when a product is ordered, it considers if ordering from a different supplier might reduce the costs. Then each period is checked to see if orders are issued to two different suppliers. If so, we consider combining the orders to only one supplier if the overall cost can be reduced. Finally, we consider initiating a new order in a period even when the inventory is not zero. Each inventory item at any period (carried forward or purchased) is investigated to see if it can be combined with another existing order for the same period on another supplier, thus reducing the total cost.

5 Computational Experience

We generated some problems to test the efficacy of the enumerative search algorithm and the heuristics on an IBM PC (Intel P4, 1.60 GHz, 256MB RAM, Windows XP). The transaction costs were generated from *int* [1000, 2000], a uniform integer distribution including 1000 and 2000. The prices were from *int* [20, 50], the holding costs from *int* [1, 5], and the demands were from *int* [1,200]. The results are shown in Table 1, where a problem size of *l, m, n* indicates number of suppliers = *l*, number of products = *m*, and number of periods = *n*. The average of 15 test problems for each size is shown. The solution times of the enumerative search algorithm rise sharply with the increase in problem size. However, the heuristics provide the solutions that are close to optimum in a very short time, and thus appear quite suitable for realistically sized problems. Among the two heuristics, the solutions provided are about the same, but H2 takes longer time than H1.

Table 1. Computational results

Problem size	Enumerative search algorithm		Heuristic 1		Heuristic 2	
	Average optimum cost	Average solution time, in mins.	Average cost, based on the heuristic	Average solution time, in millisecs.	Average cost, based on the heuristic	Average solution time, in millisecs.
3, 3, 10	101944	0.05	102584	1	102966	37
3, 3, 15	147207	18.7	147887	1	148863	51
4, 4, 10	124721	11.3	126345	1	126472	64
4, 4, 15	*	*	187320	1	188038	90
20, 20, 100	*	*	5048826	12	5163577	45983
20, 20, 200	*	*	10026074	20	10269228	178348
50, 50, 200	*	*	25373121	127	24382409	1107112

*Solutions could not be found within two hours.

6 Conclusion

We have presented a model that combines supplier selection with traditional inventory lot sizing. This turned out to be a complex combinatorial optimisation problem. An

enumerative search algorithm was developed for the solution of this problem, but even this is not fast enough for practical problems. However we presented two heuristics based on traditional inventory lot sizing, which performed very well on the tested problems. Supplier selection and supplier base reduction is an important goal in supply chain management. However, there are not many quantitative models that are available to address this goal. This appears to be a fertile area for future research.

7 Acknowledgement

We would like to thank Lam Ngok for his help in programming the heuristic algorithm and running some of the computational experiments.

8 References

Aggarwal, A. and J.K. Park. 1993. "Improved algorithms for economic lot-size problems". *Operations Research*, **41**:549-571.

Bahl, H.C., L.P. Ritzman, and J.N.D. Gupta. 1987. "Determining lot sizes and resource requirements: a review". *Operations Research*, **35:**329-345.

Evans, J.R.. 1985. "An efficient implementation of the Wagner-Whitin algorithm for dynamic lot sizing". *Journal of Operations Management*, **5**:235-239.

Federgruen, A. and M. Tzur. 1991. "A simple forward algorithm to solve general dynamic lot sizing models with *n* periods in $O(n \log n)$ or $O(n)$ time". *Management Science*, **37**:909-925.

Ganeshan, R. 1999. "Managing supply chain inventories: a multiple retailer, one warehouse, multiple supplier model". *International Journal of Production Economics*, **59:**341-354.

Hariga, M., and M. Haouari. 1999. "An EOQ lot sizing model with random supplier capacity". *International Journal of Production Economics*, **58:**39-47.

Harris, F.W. 1915. "Operations and cost". *Factory Management Series*, Chicago: A.W. Shaw Co.

Heady, R.B., and Z. Zhu. 1994. "An improved implementation of the Wagner-Whitin algorithm". *Production and Operations Management*, **3**:55-63.

Jacobs, F.R., and B. Khumawala. 1987. "A simplified procedure for optimal single-level lot sizing". *Production and Inventory Management*, **28**:39-43.

Jayaraman, V., R. Srivastava, and W.C. Benton. 1999. "Supplier selection and order quantity allocation: a comprehensive model". *The Journal of Supply Chain Management*, **35**:50-58.

Lee, C.-Y., S. Cetinkaya, and A.P.M. Wagelmans. 2001. "A dynamic lot-sizing model with demand time windows". *Management Science*, **47**:1384-1395.

Ozdamar, L., and M.A. Bozyel. 2000. "The capacitated lot-sizing problem with overtime decisions and setup times". *IIE Transactions*, **32**:1043-1057.

Saydam, C., and M. McKnew. 1987. "A fast microcomputer program for ordering using the Wagner-Whitin algorithm". *Production and Inventory Management*, **28**:15-19.

Shaw, D.X. and A.P.M. Wagelmans. 1998. "An algorithm for single-item capacitated economic lot sizing with piecewise linear production costs and general holding costs". *Management Science*, **44**:831-838.

Sox, C.R., and Y. Gao. 1999. "The capacitated lot-sizing problem with setup carry-over". *IIE Transactions*, **31**:173-181.

Wagner, H.M. and T.M. Whitin. 1958. "Dynamic version of the economic lot-size model". *Management Science*, **5**:89-96.