The Complexity of Safety Stock Placement in General-Network Supply Chains

Ekaterina Lesnaia, Iuliu Vasilescu, and Stephen C. Graves
MIT

***Abstract*— We consider the optimization problem of safety stock placement in a supply chain, as formulated in [1]. We prove that this problem is NP-Hard for supply chains modeled as general acyclic networks. Thus, we do not expect to find a polynomial-time algorithm for safety stock placement for a general-network supply chain.**

***Index Terms*— Complexity, safety stock placement, supply chain planning.**

I. Introduction

In this paper we consider an optimization problem for determining the placement of safety stocks in a supply chain. In particular, we consider the problem formulation developed by Graves and Willems in [1]. This safety stock problem can be formulated as a problem of minimizing a concave function over a polyhedron. The general concave minimization problem is known to be NP-hard. The proof of this fact can be found in [2] or in [3]. However, because the safety stock problem is defined on a particular polyhedron, in some cases we can still develop a polynomial time algorithm. For example, Graves [4] observes that the Simpson's serial system case can be solved by a dynamic program. Graves and Willems [1] develop a dynamic programming algorithm for the spanning tree networks which runs in $O(NM^2)$, where N is the number of nodes and M is the maximum replenishment service time, which is bounded from above by the sum of the lead times at each stage of the supply chain $\sum_{i=1}^{N} T_i$.

There has been no complexity results for the problem of safety stock minimization, except for Shen [5]. In [5] Shen proves that a similar problem with the upper bounds on the outbound service times is NP-hard. Here, we prove that the general problem is also NP-hard.

II. Assumptions and formulation

A. Assumptions

We state here the assumptions for the model, as introduced in [1].

- **Multi-stage network.** We model a supply chain as a network. Nodes and arcs of the network have natural interpretation in terms of the chain. Each node or stage in the network represents a processing function at which we can locate a safety stock. We place an arc from node i to node j if the output product of stage i is needed as input for production at stage j. If a node is connected to several upstream nodes, then the node is an assembly requiring inputs from each of the upstream nodes. The nodes are potential locations for holding a safety-stock of the item processed at the node.
  Due to the interpretation of the network we assume that the network does not have directed cycles. This fact says that a component once processed in a node does not return back to the node in an assembly with other components. Let N be the number of nodes and $\mathbb{A}$ be the set of arcs in the graph representing the chain.
- **Production lead-times.** We assume that each node j has a deterministic production lead-time T_j, where lead-time is the total time of production, including queueing, given that all necessary components are available.
- **Base-stock replenishment policy.** All stages operate under a periodic-review base-stock policy with a common review period. We assume that there is no delay in ordering, therefore, all the nodes see customer demand once it occurs in the demand nodes. Based on the observed demand, each stage replenishes its inventory up to the base stock level.
- **Demand process.** We assume that external demand occurs only in the demand nodes, which we define to be the nodes with zero out-degree. We denote the set of demand nodes as $\mathbb{D}$. For each node j in $\mathbb{D}$ demand $d_j(t)$ comes from a stationary process with average demand per period μ_j.
  Any other node $i \notin \mathbb{D}$ has only internal demand from its successors. We can calculate the demand in node i at time t by summing the orders placed by its immediate successors:
  $$d_i(t) = \sum_{(i,j)\in\mathbb{A}} \theta_{ij} d_j(t),$$
  where a scalar θ_{ij} is associated with each arc and represents the number of units of upstream component i required per downstream unit j. From this relationship, we find the average demand rate for the node i to be
  $$\mu_i = \sum_{(i,j)\in\mathbb{A}} \theta_{ij} \mu_j.$$
  The most important assumption of the model is that demand is bounded. In particular, for each node j there exists a function $D_j(F)$ for $F = 1, 2, \ldots, M_j$, such that

November 2004. This work was supported in part by the Singapore–MIT Alliance and by the MIT Leaders for Manufacturing Program.

Ekaterina Lesnaia holds a PhD from the MIT Operations Research Center, Cambridge MA 02139 USA, and is now a research scientist at ProfitLogic, Cambridge MA 02141 USA (email: lesnaia@alum.mit.edu).

Iuliu Vasilescu is a doctoral candidate in the MIT EECS, Cambridge MA 02139 USA (email: iuliuv@mit.edu).

Stephen C. Graves is with the Sloan School of Management and the Engineering Systems Division at MIT, Cambridge MA 02139 USA (email: sgraves@mit.edu).

1) for any period t

   $D_j(F) \geq d_j(t-F+1)+d_j(t-F+2)+\ldots+d_j(t)$;
2) $D_j(0) = 0$;
3) the function is concave and increasing for $F = 1, \ldots, M_j$;
4) $D_j(F) - F\mu_j$ is increasing in F,

where M_j is the maximum replenishment time for node j.

- **Guaranteed outbound service times.** We assume that node j provides 100% service and promises a guaranteed service time S_j to its downstream nodes. This means that demand $d_j(t)$ that arrives at time t must be filled at $t+S_j$. Note, we assume that each non demand node j quotes the same service time to each of its downstream nodes $i : (j, i) \in \mathbb{A}$.

  Also, we impose bounds on the service times for the demand nodes, i.e., $S_j \leq s_j, j \in \mathbb{D}$, where s_j is a given input that represents the maximum service time for the demand node j. The maximum service time is a parameter of the model known to the end customer. For example, if node i wants to serve its customers immediately, the firm has to set $s_i = 0$.
- **Guaranteed inbound service times.** Let SI_j be the inbound service time for the node j. We define inbound service time to be the time for the node j to get all of its inputs from nodes i: $(i, j) \in \mathbb{A}$ and to commence production. We require that $SI_j \geq S_i$ for all arcs $(i, j) \in \mathbb{A}$, since stage j cannot start production until all inputs have been received. We have shown in [6] that, if the objective is to minimize the cost of the safety stock held in the chain, there exists an optimal solution with:

$$SI_j = \max_{(i,j)\in\mathbb{A}} S_i.$$

All the parameters described here are known except for the outbound and inbound service times. These service times are decision variables for the optimization.

B. Formulation

Suppose B_j is the base stock level for a node j and $I_j(t)$ is inventory in j at time t. At time t, stage j observes demand $d_j(t)$ and starts replenishing the demand. It places an order for the input materials to the upstream nodes and replenishes the demand at the time $t+SI_j+T_j$. However, the node guarantees to satisfy the demand at time $t + S_j$. Therefore, if $t + S_j < t + SI_j + T_j$, the stage has to always store inventory to cover the time interval of $SI_j + T_j - S_j$. This interval is called the net replenishment time and we will see that the inventory that covers the interval is the base-stock level.

To provide 100% service level, we require $I_j(t) \geq 0$. To satisfy this requirement, we set the base stock B_j equal to the maximum demand over an interval of length $SI_j + T_j - S_j$, namely $B_j = D_j(SI_j + T_j - S_j)$. Hence, the expected inventory at the stage j is

$$D_j(SI_j + T_j - S_j) - (SI_j + T_j - S_j)\mu_j,$$

which represents safety stock held at the stage j.

Now, we formulate the problem $\mathcal{P}$ of finding the optimal guaranteed outbound service times S_j, $j = 1, \ldots, N$ and inbound service times SI_j, $j = 1, \ldots, N$ in order to minimize the total cost of safety stock in the chain.

$$\begin{array}{ll} \min & \sum_{j=1}^{N} h_j\{D_j(SI_j + T_j - S_j) - (SI_j + T_j - S_j)\mu_j\} \\ & SI_j + T_j - S_j \geq 0, \quad j = 1 \ldots, N \\ & S_i \leq SI_j, \quad (i, j) \in \mathbb{A} \\ & S_j \leq s_j, \quad j \in \mathbb{D} \\ & S_j, SI_j \geq 0, \text{ integer} \quad j = 1, \ldots, N \end{array}$$

where h_j denotes the per-unit holding cost for inventory at stage j.

This is a problem of minimizing a concave function over a polyhedron.

III. Complexity of the problem

Here we determine the complexity of the problem $\mathcal{P}$ stated in section II-B. We show that the problem is NP-hard by reducing a known NP-hard problem to the safety stock problem.

The idea of the proof appeared first in the unpublished note by [5]. In this note, the author reduces the Vertex Cover problem, which is known to be NP-complete, to a modification of the safety stock problem. The modified problem is essentially the same problem as problem $\mathcal{P}$, except that each node has an additional constraint on its outbound service time. The author assumes that the outbound service time for each node is bounded from above. That means, for each node i, there exists a service time s_i, such that the outbound service time is constrained as $S_i \leq s_i$. Note, that problem $\mathcal{P}$ also has similar service time constraints, but only for the demand nodes $\mathbb{D}$, while the outbound service times for the rest of the nodes are not constrained in this way. However, we have found a way to reduce an instance of the minimum-size Vertex Cover problem, which is NP-hard, to an instance of problem $\mathcal{P}$ such that a solution of problem $\mathcal{P}$ will imply a solution of the Vertex Cover instance.

We first describe the Vertex Cover problem. A vertex cover in graph G is a subset V of vertices of G such that every edge of G is incident to at least one vertex in V (see [7]). Then the optimization Vertex Cover problem for a graph is to find a vertex cover of minimum cardinality. This problem is NP-hard.

Now, we show how to reduce an instance VC of the Vertex Cover problem to an instance P_i of problem $\mathcal{P}$. Suppose, instance VC is characterized by an undirected graph G with N nodes and M edges and we want to find a minimum vertex cover. Then we perform the following steps:

1) **Make a directed graph from G.** We can arbitrarily assign directions to the edges of G. The only condition that has to be satisfied while doing so is that the directed graph has to have no directed cycles. One way to satisfy this condition is by following a simple algorithm. We first create set U with all of the nodes and an empty set L. We then choose node $i \in U$ and assign the direction to each arc (i, j), $j \in U$ from node i to node j. After that we move node i to the set L and remove all the edges (i, j). Then we pick another node from U and

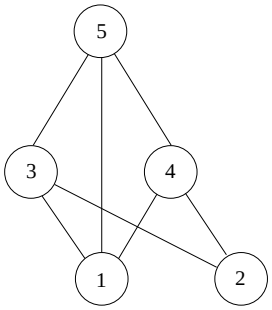

Fig. 1. Graph G for the problem VC.

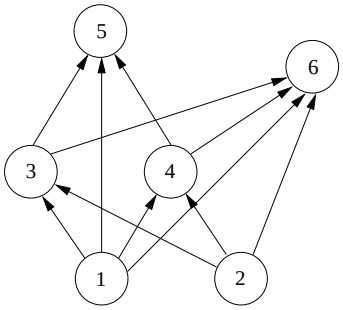

Fig. 2. Graph G' for the problem P_i.

repeat the procedure until the set U is empty. This simple algorithm will produce a directed graph with M directed edges in polynomial time, since at each step we set the direction of each edge such that the nodes in the set L are predecessors of the nodes in the set U.

As in the supply chain network, we will call the nodes with zero outdegree as demand nodes.

2) **Create a new node.** We create a new node $N+1$ such that every non demand node j has an edge $(j, N+1)$ directed from j to $N+1$. Let us denote the directed graph with the new node as G'. Figure 1 shows an example of an undirected graph G, while Figure 2 shows one way to transform the graph into graph G'.
3) **Assign parameters.** We define the safety stock function for each node of the graph G'. For simplicity, we call the safety stock at each node i as

$$SS_i(\tau_i) = D_i(\tau_i) - \mu_i(\tau_i),$$

where $\tau_i = SI_i + T_i - S_i$.

Then we require that $SS_i(\tau_i)$ is continuous, concave and satisfies

$$SS_i(\tau_i) = \begin{cases} 0, & \tau_i = 0 \\ 1, & \tau_i \geq 1 \\ \sqrt{\tau}_i & 0 < \tau_i < 1 \end{cases} \quad (1)$$

Without loss of generality, we can assume that the function is $\sqrt{\tau}_i$ on the interval $(0, 1)$, since τ_i takes only integer values. The most important properties of the function are that it is equal to 0 when $\tau_i = 0$ and is equal to 1 for all the other integer $\tau_i \geq 1$. Figure 3 shows an example of the safety stock function.

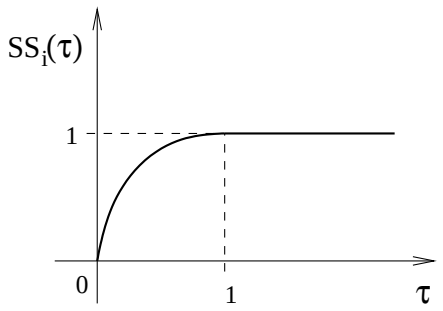

Fig. 3. An example of the safety stock function.

Next, we assign the per-unit cost of a safety stock:

$$h_i = \begin{cases} 1, & i = 1, \ldots, N \\ N+1, & i = N+1 \end{cases}$$

The lead times for the nodes are:

$$T_i = \begin{cases} 1, & i = 1, \ldots, N \\ 0, & i = N+1 \end{cases}$$

Finally, the service times promised to the end customers are 1, i.e., $s_i = 1$ for all demand nodes i. As in the formulation of problem $\mathcal{P}$, by demand nodes we mean the nodes with zero outdegree, including node $N+1$.

The procedure described above polynomially transforms graph G to an instance P_i of the safety stock placement problem $\mathcal{P}$. Now, we show that an optimal solution of problem P_i determines an optimal solution of the Vertex Cover problem VC for graph G.

Lemma 1. *Suppose we have instance P_i of the safety stock problem $\mathcal{P}$. Then it is optimal to hold no stock in node $N+1$.*

Proof:

We first show that the feasible region is not empty by constructing a solution with the cost N. Such solution is

- $S_{N+1} = 1$, $SI_{N+1} = 1$;
- $S_i = 0$, $SI_i = 1$ for $i \leq N$.

The solution is feasible. Indeed, the nonnegativity of net replenishment time $SI_i + T_i - S_i$ is satisfied for each node. Each demand node i, including node $N+1$, has $S_i \leq 1$. Also, for each arc (i, j), $S_i = 0$ and $SI_j = 1$, hence, $S_i \leq SI_j$ is also true. Therefore, the solution is feasible and the feasible region is not empty. We note also, that the solution gives cost N, since each node $i \leq N$ contributes cost 1 and node $N+1$ contributes 0 to the overall cost.

Now, we prove the statement of the lemma. Indeed, we notice that the total value of the safety stock cost in all the nodes other than $N+1$ is at most N. This is due to the fact that the maximum of the safety stock function $SS_i(\tau)$ is 1 and holding cost $h_i = 1$ for all $i \neq N+1$. All the unknown variables in the optimal solution take discrete values $0, 1, 2, \ldots$. Therefore, if a node $i \neq N+1$ holds non zero stock, then the holding cost in the node is 1.

On the other hand, if node $N+1$ holds any stock, then $\tau_{N+1} = 1$ or 2 or .... Thus, in this case the value of the safety stock function $SS_{N+1}(\tau_{N+1}) = 1$ and it contributes $N+1$ to the total holding cost, since $h_{N+1} = N+1$.

From this we conclude that if node $N+1$ holds stock, the total cost is at least $N+1$. If the node does not hold any stock, then the cost of holding inventory is at most N. Therefore, we conclude that node $N+1$ holds no stock in an optimal solution. □

Lemma 2. *Suppose we have instance P_i of the safety stock problem $\mathcal{P}$. Then in an optimal solution for each node i in the network G' we have $S_i \leq 1$.*

Proof:

The statement of the lemma follows from lemma 1, where we showed that for node $N+1$, $\tau_{N+1} = SI_{N+1} + T_{N+1} - S_{N+1} = 0$. Then, since $T_{N+1} = 0$ and $S_{N+1} \leq s_{N+1} \leq 1$, we have $SI_{N+1} \leq 1$. From the constraints of the problem P_i, $S_j \leq SI_i$ for all the arcs (i,j) in G'. Therefore, since each non-demand node is connected to node $N+1$, we have, $S_i \leq 1$ for all non-demand nodes. Also, we know that for each demand node j, $S_j \leq 1$, which we imposed by construction of problem P_i. Therefore, we can conclude that for each node i, $S_i \leq 1$. □

Lemma 3. *Suppose we have instance P_i of the safety stock problem $\mathcal{P}$. Then, in an optimal solution, for each arc (i,j), $i,j \neq N+1$ it is impossible to have values for τ_i and τ_j such that $SS_i(\tau_i) = 0$ and $SS_j(\tau_j) = 0$.*

Proof:

In lemma 2, we showed that each optimal solution of problem P_i satisfies $S_i \leq 1$. Suppose now there is an arc (i,j), $i \neq N+1$ such that $SS_i(\tau_i) = 0$ and $SS_j(\tau_j) = 0$. That is, we suppose $\tau_i = SI_i + T_i - S_i = 0$ and $\tau_j = SI_j + T_j - S_j = 0$. Then $SI_j = S_j - T_j = S_j - 1 \leq 1 - 1 = 0$. Since $SI_j \geq 0$,we have $SI_j = 0$. Because $S_i \leq SI_j = 0$, $S_i = 0$. However, $SI_i + T_i - S_i = SI_i + 1 = 0$ by assumption and $SI_i \geq 0$. Therefore, we found a contradiction, which proves the lemma □

From the lemma, we conclude that for each arc (i,j) : $i,j \neq N+1$, in an optimal solution, at least one node i or j holds safety stock. Therefore, the nodes with positive safety stock form a vertex cover V for the graph G. Moreover, by construction of the cost function, each node that holds safety stock contributes cost 1 to the objective function of the safety stock problem. Therefore, the objective function value is equal to the cardinality of the vertex cover V. By solving the safety stock problem, we find the minimum cost of safety stock which equals the cardinality of a vertex cover of graph G.

To prove that we find a minimum vertex cover by solving the safety stock problem P_i, we only need to prove that the minimum of problem P_i does not depend on the orientation of the graph. Step 1 assigns the orientation to graph G arbitrarily, which determines the demand nodes and the relationship between the variables. If arc (i,j) is directed from node i to node j, then the corresponding constraint is $S_i \leq SI_j$. If, however, the orientation were reversed, the constraint is $S_j \leq SI_i$. Therefore, the problems are different and can in theory give different solutions. Consequently, we have to show that they indeed give the same solutions independent of the orientation.

For the purposes of the next lemma, we define the *vertex cover assignment* or *v.c. assignment* of the safety stock problem. A v.c. assignment is a distribution of the safety stock in the nodes of a graph such that for each arc (i,j) the assignment implies holding stock in i or in j or in both. We notice, that for the problem P_i, a v.c. assignment of the safety stock creates a vertex cover of graph G. This is because the cost of holding stock in a node is always one, which is equivalent to putting the node into the vertex cover set. However, in the safety stock problem setting, we refer to the v.c. assignment and in the VC problem setting - to the vertex cover.

Lemma 4. *Suppose we have a directed graph G and safety stock problem P_i on $G' = G \cup \{N+1\}$ as described above. Then for each v.c. assignment of safety stock on G, there exists a feasible solution of the safety stock problem P_i.*

Proof:

To prove the lemma we consider a v.c. assignment and explicitly construct a feasible solution of the safety stock problem. We consider any node i of graph G. Depending on the v.c. assignment, the node holds or does not hold safety stock in i.

- If there is zero stock in i, we set $SI_i = 0$, $S_i = 1$.
- If there is nonzero stock in i, we set $SI_i = 1$, $S_i = 0$.

To complete the solution for all the nodes of problem P_i, we set $SI_{N+1} = 1$ and $S_{N+1} = 1$.

The solution is feasible. First, we see that for every demand node i, $S_i \leq s_i = 1$. It is also obvious that $SI_i + T_i - S_i \geq 0$ is satisfied for this solution. In the zero stock case, the solution gives $SI_i + T_i - S_i = 0 + 1 - 1 = 0$ and indeed implies zero stock. In the nonzero stock case, it satisfies $SI_i + T_i - S_i = 1 + 1 - 0 = 2$ and implies stock $SS_i(2) = 1$.

Next, we have to check that the constraint $S_i \leq SI_j$ for any arc (i,j) is satisfied. Indeed, SI_{N+1} imposes a constraint on all the outbound service times $S_i \leq 1, i \notin \mathbb{D}$. The proposed solution clearly satisfies the constraint.

Now, consider node $i \in G$. Because the solution is a v.c. assignment, if node i has zero stock, all the nodes connected to i have to have nonzero stock. That means, if $j \in G$ is downstream of i, $SI_j = 1$ and constraint $1 = S_i \leq SI_j = 1$ is satisfied. If node j is upstream of i, $S_j = 0$ and the constraint $0 = S_j \leq SI_i = 0$ is satisfied again.

If node $i \in G$ has nonzero stock, then the solution again does not violate the constraint. Indeed, as we showed before, $S_j \leq 1$ for any node in graph G, therefore, $SI_i = 1$ does not violate constraints $SI_i \geq S_j$ for all arcs (j,i) . Because $SI_j \geq 0$, $S_i = 0$ does not violate constraints $S_i \leq SI_j$ for all arcs (i,j).

Therefore, we conclude that the solution is feasible and this proves the lemma. □

Lemma 4 shows, that for every v.c. assignment, there is a feasible solution of the safety stock problem. Since every v.c. assignment of the safety stock problem P_i is equivalent to a vertex cover on graph G, we conclude that for every vertex cover of graph G we can always find a feasible solution of problem P_i.

Corollary 1. *Suppose we want to find a minimum vertex cover on an undirected graph G. Then, we can transform the problem*

into problem P_i and solve the problem optimally to obtain a minimum vertex cover, independent of the orientation assigned during the transformation.

Proof:

By lemma 3, an optimal solution of problem P_i is a v.c. assignment on G with cost K. Suppose, there exists a transformation of VC problem into problem P_i' with different orientation and with strictly smaller cost $K' < K$. But the solution of problem P_i' is a v.c. assignment on G as well. Therefore, by lemma 4 there exists a solution of problem P_i with the same cost K', which contradicts optimality of K.

We conclude that for any orientation of graph G, problem P_i gives an optimal solution to the VC problem. □

Corollary 1 shows that by solving the safety stock problem optimally, we solve the Vertex Cover problem for the graph G. We can conclude now that problem $\mathcal{P}$ is NP-hard.

IV. Conclusion

Previous work has developed efficient algorithms for restricted versions of problem P. For instance, Graves and Willems [1] give a dynamic programming algorithm for spanning tree networks which runs in $O(NM^2)$. However, it has not been known as to whether or not such algorithms exist for problem P defined on a general network. In this paper, we provide a proof that problem P is NP-hard. Thus, we cannot expect to develop a polynomial-time algorithm for problem P for general-network supply chains. As a consequence one would want to consider approximate solution procedures, and/or enumerative algorithms such as branch and bound. Indeed, there has been recent encouraging research on both counts, e.g., [6], [8], and [9].

Acknowledgment

The authors are grateful for support from the Singapore-MIT Alliance (SMA), an engineering research and education collaboration among the National University of Singapore, Nanyang Technological University, and MIT.

References

[1] S. C. Graves and S. Willems, "Optimizing strategic safety stock placement in supply chains," *Manufacturing & Service Operations Management*, vol. 1, no. 2, pp. 68–83, Winter 2000.

[2] S. Sahni, "Computationally related problems," *SIAM Journal on Computing*, vol. 3, pp. 262–279, 1974.

[3] S. Vavasis, "Quadratic programming is in NP," *Information Processing Letters*, vol. 36, pp. 73–77, 1990.

[4] S. Graves, "Safety stocks in manufacturing systems," *Journal of Manufacturing and Operations Management*, vol. 1, no. 1, pp. 67–101, 1988.

[5] Z.-J. Shen, "Note on the safety stock placement," 2003, unpublished.

[6] E. Lesnaia, "Optimizing safety stock placement in general network supply chains," Ph.D. dissertation, MIT, 2004.

[7] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 2nd ed., ser. Algorithms and Combinatorics. Springer Verlag, 2002, vol. 21.

[8] S. Humair and S. Willems, "Optimal inventory placement in networks with clusters of commonality," January 2003, working paper.

[9] T. Magnanti, Z.-J. M. Shen, J. Shu, D. Simchi-Levi, and C.-P. Teo, "Inventory placement in acyclic supply chain networks," 2004.