ONLINE SUPPLEMENT TO: OPTIMIZING STRATEGIC SAFETY STOCK PLACEMENT IN GENERAL ACYCLIC NETWORKS

SALAL HUMAIR

School of Science and Engineering, Lahore University of Management Sciences

Lahore 54792, Pakistan, salal@lums.edu.pk

SEAN P. WILLEMS

School of Management, Boston University, Boston, MA 02215, willems@bu.edu

---

This online supplement to Humair & Willems (2008) contains the following. For description of acronyms, see Humair & Willems (2008).

(1) A literature review.

(2) An overview of the standard guaranteed service model.

(3) Some common examples of non-concave generalized cost functions.

(4) A counter-example to the existence of an optimal spanning tree for problem **P**.

(5) Implementation details for the GNA algorithm.

(6) A table with actual run times for the algorithm GNA,and the heuristics HGNA and TGNA.

(7) A discussion of problem **P**'s dual space, the structure of which motivated HGNA.

(8) Proofs of Theorem 3.1 and Proposition 3.3 in Humair & Willems (2008).

1. Literature Review

Literature on optimizing safety stocks in multi-echelon networks is extensive as literature reviews in Graves & Willems (2003b), Humair & Willems (2006), and Simchi-Levi & Zhao (2005) show. Instead of trying to be comprehensive here, we only contrast the GS model to other multi-echelon safety stock models, and then situate our algorithm within the work on network topologies within the GS model.

To contrast models, we note that no structural results are available for the form of the optimal policy in arbitrary acyclic networks, see for example Zipkin (2000), so literature focuses on approximate models and installation base stock ordering policies. These models can be classified based on how service between adjacent stages is modeled; the literature forming a dichotomy comprised of guaranteed service (GS) and stochastic service (SS) models.

GS models assume that each stage is quoted a service time from its upstream stages, after which it receives replenishment with certainty. Similarly each stage quotes a service time to its customers, after which it satisfies its demand with certainty. The argument is that the safety stock should be maintained to cover a reasonable amount of demand variability (corresponding to a desired service level), but should not be designed to respond to larger deviations. When large deviations happen, the system should employ countermeasures beyond safety stock, for example expediting, to guarantee service times. Graves & Willems (2003b) provide a check-out clerk analogy to illustrate the argument.

SS models on the other hand assume that each stage may suffer a stock-out and an attendant delay from one or more of its suppliers. This delay must be accounted for when computing a service level for the stage's own customers. The assumption is that holding inventory is the only countermeasure available to stages to offer the desired service level, in the event any of their own supply stages stock out. The principal challenge is in computing the delays in general networks. Therefore approximations such as in Ettl et al. (2000), Lee & Billington (1993) and Simchi-Levi & Zhao (2005) have been proposed.

Our paper is an extension of the GS framework that began with a 1955 manuscript by Kimball (reprinted Kimball (1988)). Simpson (1958) formulated the GS model for serial-line networks and established the existence of an extreme point property (or all-or-nothing policies) for concave demand. The works of Inderfurth (1991), Inderfurth & Minner (1998), and Graves & Willems (1996, 2000, 2003a) extend Simpson's work to supply chains modeled as assembly networks, distribution networks, and spanning trees. For each network topology, a single-state variable dynamic program is formulated to solve the resulting nonlinear-integer problem of minimizing total safety stock cost.

Humair & Willems (2006) consider a more general topology they define as *networks containing clusters of commonality* (CoC); informally described as networks of bi-partite sub-graphs that form spanning-trees when each bi-partite sub-graph is collapsed to a single node. Humair & Willems (2006) formulate a single-state dynamic program where the stage cost-to-go function depends on the stage's orientation in the network and the cluster.

The node ordering and cost-to-go definition still allow CoC networks to be optimized with a single pass through the stages.

Minner (2000), Magnanti et al. (2006), Karimi et al. (2004) and Lesnaia (2004) all consider concave costs for safety stock placement in general acyclic structures. Minner (2000) augments the state space of a node to consider all-or-nothing stocking policies at upstream or downstream nodes, depending on the node's relation to the subgraph containing commonality. Due to the combinatorial nature of the exact formulation, Minner (2000) investigates several heuristic approaches, finding that tabu search performs best. Magnanti et al. (2006) develop a piecewise linear approximation to concave stage cost functions that effectively solves sparse acyclic networks ranging from ten to 100 stages. Their solution method relies on adding redundant inequalities to the LP relaxation and refining the piecewise linear approximation in an iterative fashion. Karimi et al. (2004) develop two heuristic algorithms for dense acyclic networks ranging from ten to 8,000 stages. The heuristics employ a continuous and two-piece linear estimator, respectively, for the concave cost functions, successively iterating to refine the quality of the approximation. Lesnaia (2004) shows that the general network problem with concave costs is NP-hard. She uses the extreme point properties of the optimal solution within a branch and bound scheme to find the optimal solution, and considers the application of their algorithm for two-layer networks.

The thrust of algorithmic research for concave cost GS models has been to exploit the special structure of the problem to construct faster algorithms, rather than focus on general concave optimization techniques. For instance, in contrast to the complexity of general concave optimization techniques (c.f. Guisewite & Pardalos (1990)), the algorithm of Graves & Willems (2000, 2003a) is pseudo-polynomial for a spanning tree network, while the algorithms for all-or-nothing policies for only assembly or only distribution networks are polynomial (c.f. Magnanti et al. (2006)). Our work similarly exploits the network structure of the problem, since as discussed in Section 3, it is not easy to see how to adapt standard non-convex optimization approaches to deal with the full range of the generalized cost functions like those in Figures 2-5.

2. **The Standard GS Model**

In Graves & Willems (2000), hereafter referred to as GW, the supply chain is represented as a network with nodes representing stocking points (stages) and arcs representing precedence relationships between stages. Time is discrete. In each time period, external demand arises at stages with no outgoing arcs. All other stages in the chain

receive orders from their downstream stages. All stages place orders for their inputs on their upstream stages. Infinite supply is available to stages without incoming arcs.

Each stage has three times associated with it: a stage time, an outgoing service time, and an incoming service time. Stage time is the time from when all of the stage's inputs are available, to when its outputs are produced. Outgoing service time is the elapsed time from when an order is received (from an adjacent downstream stage) to when it is fulfilled. Incoming service time is the elapsed time from when orders are placed (on all adjacent upstream stages) to when *all* orders are fulfilled. Incoming and outgoing service times are the decision variables for the model.

Each stage follows a one-for-one base stock ordering policy, i.e. placing a replenishment order on its suppliers equal to the demand its observes in each time period. Each stage sets its base stock to guarantee that it can meet its outgoing service time with certainty. The interpretation and reasonableness of this assumption is discussed in detail in Graves & Willems (2000, 2003b).

2.1. **Single-stage inventory dynamics.** In GW's notation, T_i, S_i, SI_i denote stage i's stage time, outgoing service time and incoming service time. The base stock requirements are connected to the decision variables (service times) through the *net replenishment time* $\tau_i = SI_i + T_i - S_i$. If stage i starts at time 0 with initial inventory equaling the stage's base stock B_i, and $d_i(t)$ is the realized demand in period t, then guaranteed service implies realized inventory on hand at time t, $I_i(t) = B_i - \sum_{v=0}^{t-S_i} d_i(v) + \sum_{w=0}^{t-T_i-SI_i} d_i(w) = B_i - \sum_{v=t-T_i-SI_i+1}^{t-S_i} d_i(v)$. To satisfy the service time guarantee, the base stock B_i needs to be set so that $I_i(t)$ is always non-negative, i.e. $B_i \geq \sum_{v=t-T_i-SI_i+1}^{t-S_i} d_i(v)$. Since maximum demand over any time interval of length x is described by a function, denote it as $D_i(x)$, $I_i(t) \geq 0$ for all t can be assured by setting $B_i = D_i(SI_i + T_i - S_i) = D_i(\tau_i)$, which ties the base stock requirements to the net replenishment time.

2.2. **Stage cost functions.** With the exception of Humair & Willems (2006), other GS literature including GW models stage i's stock cost as concave functions, multiplying the concave function $D_i(.)$ by a scalar per unit cost. More specifically, other GS literature assumes the stage cost has a functional form $C_i D_i(\tau_i) - C_i \mu_i \tau_i$, where C_i is the per unit inventory holding cost at stage i and μ_i the average demand per period.

2.3. **The mathematical programming formulation.** The network model is an acyclic connected graph $G = (V, \mathcal{A})$, where V is the set of all nodes and $\mathcal{A}$ the set of all arcs. Let $V_e \subset V$ be the set of external demand nodes

and $V_s \subset V$ be the set of supply nodes (with no incoming arcs). The maximum S_i quotable at a demand node is constrained by E_i. Then, letting $|V| = n$, the problem of interest $\mathbf{P}$ is as follows, where S_i and SI_i are integral decision variables.

$$\begin{aligned}
\mathbf{P}: \quad \min \quad & \sum_{i=1}^{n} C_i \Big\{ D_i(SI_i + T_i - S_i) \quad - \quad \mu_i(SI_i + T_i - S_i) \Big\} \\
\text{s.t.} \quad & S_i - SI_i \le T_i, && i = 1, \ldots, n, \\
& S_j - SI_i \le 0, && \forall (j,i) \in \mathcal{A}, \\
& S_i \le E_i, && \forall i \in V_e, \\
& SI_i = 0, && \forall i \in V_s, \\
& S_i,\ SI_i \ge 0,\ \text{integral}, && i = 1, \ldots, n.
\end{aligned}$$

The constraints of $\mathbf{P}$ simply ensure that each stage's outgoing service time does not exceed the sum of its stage time and incoming service time, the incoming service time for stage i is no less than the maximum outgoing service times quoted by the stages directly supplying i, and the outgoing service times to end-item customers are bounded by the E_i's.

When stage costs are concave, the optimal solution to $\mathbf{P}$ is an extreme point, or an all-or-nothing stocking policy, each stage either stocking all it can (setting its outgoing service time to 0) or nothing (setting its net replenishment time to 0 using $S_i = SI_i + T_i$). Further, the structure of costs assumed in other GS work implies the optimal solution is contained in a bounded region given by $SI_j \le \max_{i \in V}(M_i - T_i), \forall j$. M_i denotes the longest stage-time path through stage i, i.e. $M_i = T_i + \max_{k:(k,i)\in A}\{M_k\}$.

2.4. **Demand propagation.** GW proposed a demand propagation algorithm for spanning trees, which needs to be modified for general networks since demand at internal stages might be correlated even when the end-item demands are independent. In GW, at demand stages, the maximum demand function $D_i(t)$ and per-period mean demand μ_i are inputs to the model. At non-demand stages $D_i(t) = \mu_i t + \left(\sum_{j:(i,j)\in\mathcal{A}} \left[\phi_{ij}(D_j(t) - \mu_j t)\right]^p \right)^{1/p}$; where ϕ_{ij} is the units of stage i's product which are needed for one unit to be produced at stage j, and $\mu_i = \sum_{j:(i,j)\in\mathcal{A}} \mu_j$ is the mean demand seen at stage i. p is called a *pooling* factor which can be varied to model risk pooling demand streams from adjacent downstream stages, $p = 2$ modeling independent demand streams and $p = 1$ modeling perfectly correlated demand streams.

In general networks, even when end-item demand streams are independent, the demand at internal stages may be correlated. For example consider a network where $V = \{1, 2, 3, 4\}$ and $\mathcal{A} = \{(1, 2), (1, 3), (2, 4), (3, 4)\}$ where demand streams seen by stage 1 are perfectly correlated. Therefore the following expression should be used at non-demand stages (with $p = 2$) to compute demand bounds: $D_i(t) = \mu_i t + \left(\sum_{k \in V_e} \left[\tilde{\phi}_{ik}(D_k(t) - \mu_k t) \right]^p \right)^{1/p}$. Here $\tilde{\phi}_{ik} = \sum_{j:(i,j)\in\mathcal{A}} \phi_{ij} \tilde{\phi}_{jk}$ for each $k \in V_e$, with the initial conditions $\tilde{\phi}_{kk} = 1, \tilde{\phi}_{kj} = 0$ for each $k, j \in V_e$. Similar to GW, to model correlated end-item demands, $p \neq 2$ can still be used in the above formula, but p would then represent the correlation between the end-item demand streams, not the correlation between the demand from adjacent downstream stages.

2.5. **The Graves and Willems (2000) spanning tree algorithm.** The dynamic program in GW builds on the property that nodes in any spanning tree can be labeled with unique indices $k \in \{1, \ldots, n\}$, such that every node except one (called *root*) has at most one adjacent node with an index higher than its own (called *parent*). The *child sub-tree* of node k is the set of all nodes that remains connected to k after the link to its parent is removed from the tree. Once stages are labeled with index k, re-index all problem parameters in $\mathbf{P}$ in terms of k, i.e. stage-specific quantities such as T_k, S_k, M_k, etc. now refer to a stage with index k. Similarly, an arc $(i, j) \in \mathcal{A}$ now refers to an arc between stages with indices i and j. Let p_k be the index of stage k's parent.

There are two forms of the optimal cost-to-go function for stage k. If $(k, p_k) \in \mathcal{A}$, $f_k(S)$ is the optimal cost of the sub-network including k and its child sub-trees if k were to quote an outgoing service time S. If $(p_k, k) \in \mathcal{A}$, $g_k(SI)$ is the optimal cost of the sub-network including k and its child sub-trees when the incoming service time to k is SI. The cost-to-go function of the root ($k = n$) is defined as $g_n(SI)$ by convention.

The following dynamic program **SDP** is then a reformulation of $\mathbf{P}$, where $(S - T_k)^+$ denotes $\max(0, S - T_k)$.

If $(k, p_k) \in \mathcal{A}$, for $S = 0, \ldots, M_k$,

$$f_k(S) = \min_{SI:\ (S-T_k)^+ \le SI \le M_k - T_k} \left\{ c_k(SI, S) + \sum_{\substack{(i,k)\in\mathcal{A} \\ i<k}} \min_{x_i:\ x_i \le SI} f_i(x_i) + \sum_{\substack{(k,j)\in\mathcal{A} \\ j<k}} \min_{y_j:\ y_j \ge S} g_j(y_j) \right\}.$$

If $(p_k, k) \in \mathcal{A}$, for $SI = 0, \ldots, M_k - T_k$, where we let $E_k = \infty$ if $k \notin V_e$,

$$g_k(SI) = \min_{S:\ 0 \le S \le \min(SI + T_k, E_k)} \left\{ c_k(SI, S) + \sum_{\substack{(i,k)\in\mathcal{A} \\ i<k}} \min_{x_i:\ x_i \le SI} f_i(x_i) + \sum_{\substack{(k,j)\in\mathcal{A} \\ j<k}} \min_{y_j:\ y_j \ge S} g_j(y_j) \right\}.$$

3. Generalized Cost Functions

We allow stage costs of any generalized form $c_i(SI_i, S_i)$ in $\mathbf{P}$, instead of the concave form $C_i\{D_i(SI_i + T_i - S_i) - \mu_i(SI_i + T_i - S_i)\}$. There are no structural limitations on $c_i(SI_i, S_i)$ as long as a stage's cost depends on only its own service times, i.e. it can be any non-concave, non-monotone or non-closed form function.

With generalized stage costs, the GS model can capture commonly occurring and important real-world phenomena such as variable stage times, non-nested review periods and non-nested batch-ordering. Variable stage times are the most common, as the 38 industrial supply chains in Willems (2008) show, where 26 chains from 22 different companies have variable stage times. Expected inventory levels under variable stage times can be computed using a stage-level model, or can be simulated under the guaranteed service assumption and base stock ordering. For the chain topology of Figure 1, Figure 2 shows the cost functions obtained from a simulation, where μ_D, σ_D and μ_T, σ_T represent the mean and standard deviation of demand and stage times respectively.

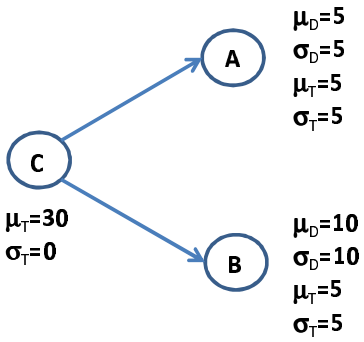

FIGURE 1. Chain topology for the generalized cost functions graphs.

Another common and important practical phenomenon is non-nested review periods. Bossert & Willems (2007) detail the importance of non-nested review periods in the chemical industry. They present a model from Celanese, a $6 billion chemical company, in detail. They also document that their review periods research has been applied at more than a dozen Fortune 500 companies including Black and Decker, Boston Scientific, Hewlett Packard, Honeywell, Intel, and Procter & Gamble. Figure 3 shows results for non-nested review periods from a simulation although it is also possible to analytically compute base stocks as in Bossert & Willems (2007).

Non-nested batch-ordering policies are another common feature which produce nonlinear effects as the simulated functions in Figure 4 show. Finally, integrating two or more of these phenomena compounds the effects as in Figure 5 where variable stage times and non-nested review periods occur simultaneously.

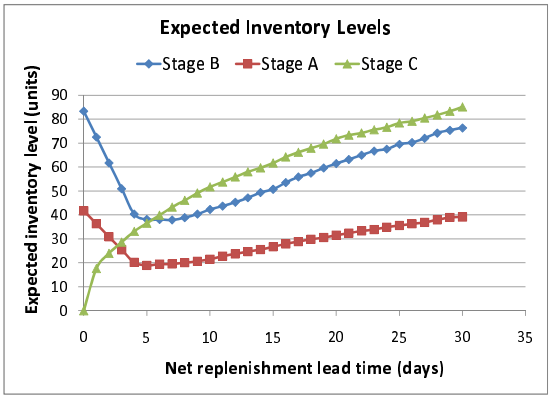

Figure 2: Simulated cost functions under variable stage times.

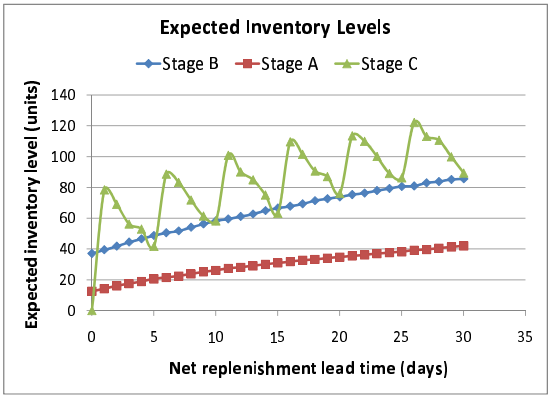

Figure 3: Simulated cost functions under non-nested review periods.

The generality of the costs suggests that, viewing $c_i(SI_i, S_i)$ as an arbitrary mathematical construct, problem **P** may be viewed in a context wider than inventory optimization. For instance, it is easy to see **P** as a flow problem where stage i directs a flow S_i to its successors, each successor j chooses a flow from one of its suppliers, $\max_{i:(i,j)\in\mathcal{A}} S_i$, and incurs some arbitrary flow related cost $c_i(SI_i, S_i)$, which could have an interpretation as some flow-related physical quantity.

Figures 2-5 also help explain why standard GS optimization approaches cannot be used for generalized cost functions. For instance, existing GS approaches either assume the optimal solution is an extreme point of **P** (Minner (2000), Lesnaia (2004)), which cannot handle functions of the type in Figures 2 and 5 where the optimal solution may be in the interior of the feasible region; or approximate the objective function with a piecewise linear function (Magnanti et al. (2006), Karimi et al. (2004)), which makes functions of Figures 3-5 difficult to handle, where easy linear approximations are difficult to construct.

Figures 3-5 also help explain the difficulty in using standard non-convex optimization approaches. Our functions do not have a well-known structure such as sub-modularity or quasi-concavity (Nemhauser & Woolsey (1988)), so efficient approximation algorithms available for these classes of functions cannot be used. That primarily leaves search algorithms, which fall in two classes. The first class of search algorithms, e.g conjugate gradient methods etc. (Bazaraa et al. (1993)), depend on continuity and differentiability. Even if service times were continuous, the objective function is not differentiable as the figures show most prominently in the case of review periods. More acutely, even tests for local optimality for these generalized cost functions, i.e. termination criteria for use in such search algorithms are difficult to construct.

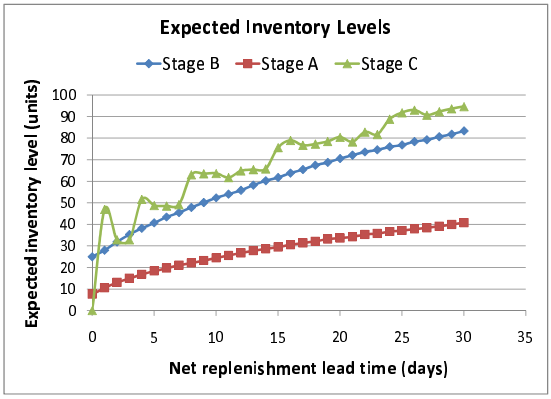

Figure 4: Simulated cost functions under non-nested batch-ordering.

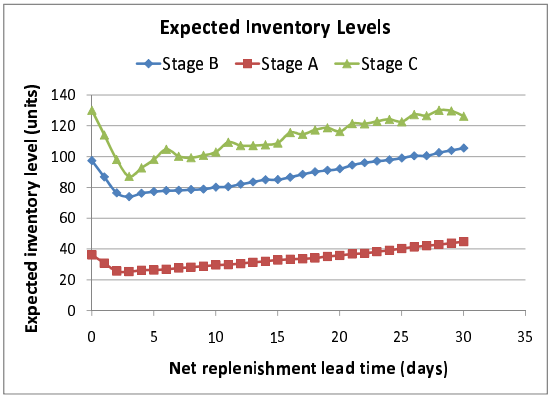

Figure 5: Simulated cost functions under non-nested review periods and variable stage times.

The other class of search algorithms are probabilistic, such as genetic algorithms (Goldberg (1989)). If the structure of the function to the optimized is known *a priori*, some strategy may be devised to perform well for that class of functions. In the diverse structures permitted by generalized cost functions, finding a search strategy which performs well over the entire range of possible function structures is quite hard. Moreover, evaluation times for large populations of solutions such as in genetic algorithms are prohibitive. Further, even probabilistic guarantees on the optimal value are difficult to obtain.

4. Counter-Example to the Existence of an Optimal Spanning Tree

It is easy to create an example to demonstrate that the general networks problem cannot be reduced to the problem of finding an optimal spanning tree. In the seven-stage example of Figure 6, T_i denotes the stage time and C_i denotes the per unit inventory holding cost at stage i. Demand is propagated from the end-item demand stages E, F, and G as described in Humair & Willems (2008), $\phi_{ij} = 1$ if $(i, j) \in \mathcal{A}$ and $p = 2$. We assume the standard stage cost function used in other GS work, i.e. $c_i(SI_i, S_i) = kC_i\sigma_i\sqrt{SI_i + T_i - S_i}$; we choose $k = 1.645$ for all stages to reflect a service level target of 95%.

Figure 7 shows that the optimal solution to every spanning tree associated with Figure 6 violates at least one constraint in the full network. The example is more complicated than just a diamond network (stages B, C, D, and F) because it is difficult to make all spanning trees non-optimal in a diamond for realistic holding cost and demand propagation as in Humair & Willems (2008).

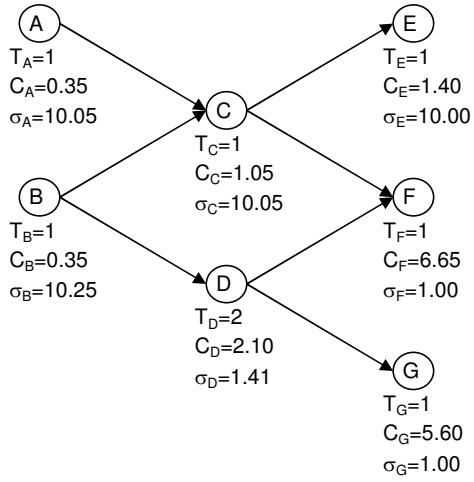

FIGURE 6. Seven-stage acyclic network to demonstrate that the general networks problem can not be reduced to the problem of finding an optimal spanning tree.

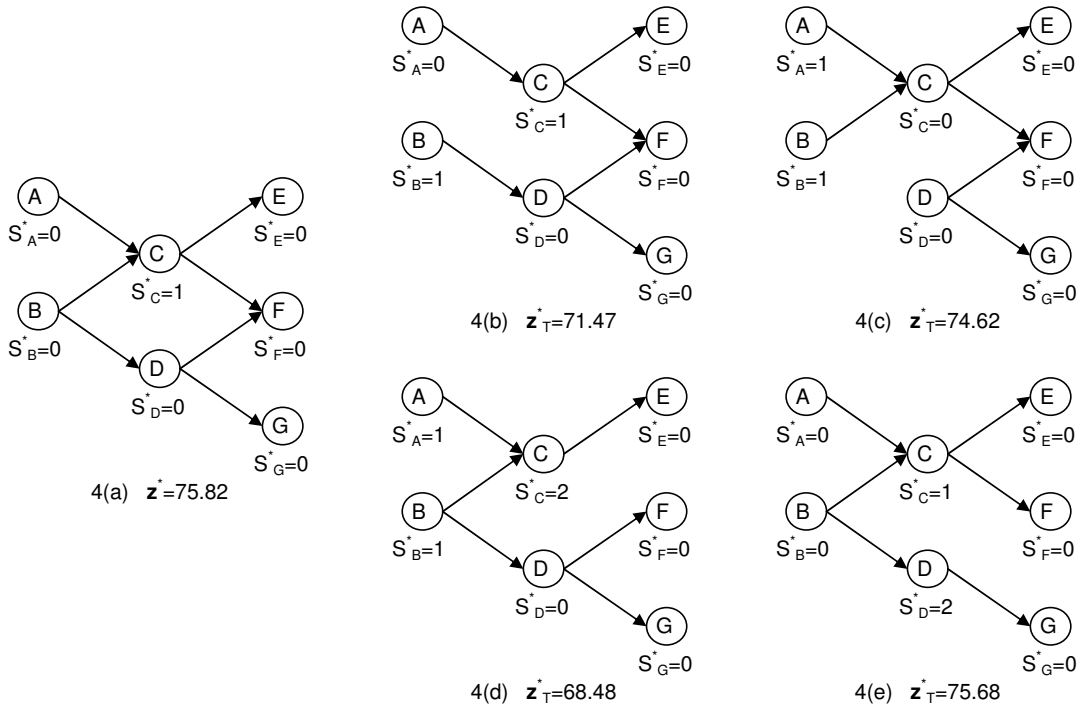

FIGURE 7. Counter-example to the existence of an optimal spanning tree solution for the general network in Figure 6. Figure **(a)** illustrates the optimal outgoing service times and total stage cost z^* for the general network in Figure 6. Figures **(b)**, **(c)**, **(d)**, and **(e)** illustrate the optimal outgoing service times, optimal tree costs z^*_T, and the violated constraint in $\mathbf{P}$ (corresponding to the missing link) for each of the general network's four spanning tree relaxations.

5. **Implementation**

5.1. **Selecting a spanning tree.** In theory one can use any spanning tree in GNA since the proof for GNA's correctness does not rely on any particular spanning tree. In practice, we know the efficiency of any branch-n-bound is strongly dependent on the quality of the relaxations. We therefore attempt to select a tree which we

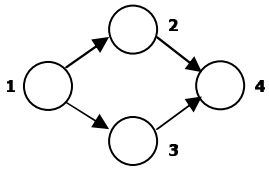

FIGURE 8. Diamond network for discussion of **P**'s dual space.

believe will result in a small number of infeasible links once the spanning tree is optimized. We use a heuristic to rank links.

To rank links, we note that realistic cost functions usually depend on the per unit inventory holding cost and the standard deviation of demand at a stage. We therefore associate a cost with each link $(j,i) \in \mathcal{A}$, of the form $-(C_j + C_i)\sigma_i$. Once link costs are assigned, we pick a spanning tree that minimizes the link costs. This is the classical minimum cost spanning tree problem which can be solved in $O(n^2)$ or $O(\mid \mathcal{A} \mid \log n)$ time (c.f. Nemhauser & Woolsey (1988) for instance). Incidentally, the proof of correctness for the branch-n-bound does not rely on using the same spanning tree for each call to routine $\mathcal{R}$. One could therefore choose a different spanning tree or choose from a small set of spanning trees for each call to $\mathcal{R}$ and still find the optimal solution.

5.2. **Initial bounds $\underline{z}$.** We generically initialized the cost of the best solution in the algorithm $\underline{z} = \infty$, but a much better bound is *a priori* available in practice. For instance, one can set $\underline{z}$ to be the cost of the following feasible solution: $SI_i = \max_{j:(j,i)\in\mathcal{A}} S_j$ for all i with $SI_i = 0$ for all stages with no incoming arcs, and $S_i = SI_i + T_i$ if $i \notin V_e$, $S_i = \min(E_i, SI_i + T_i)$ if $i \in V_e$. Alternatively one could let $\underline{z}$ be the cost of the following feasible solution: $SI_i = 0$ for all i, and $S_i = 0$ if $i \notin V_e$, $S_i = \min(E_i, T_i)$ if $i \in V_e$. We use the second method in our implementation, but clearly the minimum of the two costs is a better bound.

5.3. **Choosing an infeasible link.** While the correctness of the algorithm does not depend on which infeasible link is chosen to partition the feasible region, we actually use the least cost (rank) infeasible link, using the same link cost as used to select the spanning tree.

5.4. **Partitioning the feasible region.** We have used one rule for partitioning the feasible region, but others are readily available as well. For instance, in *Step 5* of GNA, given an infeasible link $(j,i) \in \mathcal{A}$ one could use

$\Omega \cup \{S_k \leq \bar{S}, \forall k : (k,i) \in \mathcal{A}\}$ for the *Upperbound* step, and use $\Omega \cup \{SI_i > \bar{S}\}$ for the *Lowerbound* step. The rule we propose in *Step 5* of GNA's routine $\mathcal{R}$ can be visualized as a bias towards first trying solutions that push inventory downstream in the chain; whereas the rule we have mentioned in this section has a bias towards first trying solutions that push inventory upstream in the chain. The heuristic HGNA, motivated by the structure of the dual space, in fact uses this second rule when pruning the branch-n-bound tree for performance.

6. **Run Times for GNA, HGNA, TGNA**

Table 1 contains the actual run times of GNA, HGNA and TGNA.

7. **Problem P's Dual Space**

HGNA was motivated by the mechanics of the GNA algorithm in the dual space for **P**. Although we have not characterized the exact effect of the dual space structure on HGNA's performance, we present the motivating intuition as it could suggest fruitful research directions to others. The development below closely follows section 6.1 of Bazaraa et al. (1993). For ease of exposition, we consider the simple diamond network example of Figure 8(a). Suppose that link $(1,2)$ is dropped from the network to form a spanning tree. Let $h(\mathbf{S})$ denote the objective function value of **P** for any value of the service times **S**. **P** for the example network is to minimize $h(\mathbf{S})$ subject to $S_1 - SI_2 \leq 0$ and $\mathbf{S} \in F_{\Omega^T}$, in the notation of Humair & Willems (2008). In a two dimensional (y,z) plane, the set $H = \{(y,z) : y = S_1 - SI_2, z = h(\mathbf{S})\}$ is the image of the set of all spanning tree solutions F_{Ω^T} under the functions $S_1 - SI_2$, $h(\mathbf{S})$. The solution to **P** finds a point in H with $y \leq 0$ which has the minimum ordinate. For simple cost functions of the type $c_i(SI_i, S_i) = kC_i\sigma_i\sqrt{SI_i + T_i - S_i}$ (see Humair & Willems (2008) for notation), Figure 9 shows the upper and lower envelopes of H for some different cost and time parameters. A spanning tree minimization in GNA corresponds to moving the horizontal line $z = h(\mathbf{S})$ as far down as possible while it remains in contact with H. If the point at which contact with H remains has $y \leq 0$, the optimal solution to the spanning tree is optimal for the network, otherwise if the contact point is y^*, GNA first optimizes with the bound $S_1 \leq \lfloor y^*/2 \rfloor$ and then with $SI_2 > \lfloor y^*/2 \rfloor$.

Figure 9 shows that the lower envelope of H is not sharply varying, and that its qualitative form, for GNA's purpose, may fall into a few distinct categories. For instance, Figures 9(a) and 9(c) might produce similar bounding behavior (bounding S_1 all the way down to 0 for instance, as Figure 10 shows), and Figures 9(b) and 9(g) might

				GNA Algorithm				HGNA Heuristic				TGNA Heuristic			
Chain	**Stages**	**Links**	**MaxChain Length**	**Time (secs)**	**Optimal Cost ($)**	**Total Iters**	**Optimal Iter**	**Time (secs)**	**Optimal Cost ($)**	**Total Iters**	**Optimal Iter**	**Time (secs)**	**Optimal Cost ($)**	**Total Iters**	**Optimal Iter**
01	8	10	38	0.00	3.35E+04	9	1	0.00	3.35E+04	9	1	0.015	3.35E+04	9	1
02	13	13	64	0.03	9.51E+06	9	1	0.03	9.51E+06	9	1	0.032	9.51E+06	9	1
03	17	18	79.8	0.11	6.94E+06	19	17	0.06	6.94E+06	11	9	0.062	6.94E+06	11	9
04	22	39	204	0.63	4.90E+04	19	9	0.53	4.90E+04	15	3	0.64	4.90E+04	19	9
05	27	31	47.35	0.00	2.01E+06	1	1	0.00	2.01E+06	1	1	0.015	2.01E+06	1	1
06	28	28	96	0.00	7.78E+04	1	1	0.02	7.78E+04	1	1	0.016	7.78E+04	1	1
07	38	78	85	1.47	5.22E+06	101	15	0.28	5.22E+06	20	6	1.453	5.22E+06	100	15
08	40	48	91.04	0.28	1.63E+06	11	5	0.27	1.63E+06	11	5	0.265	1.63E+06	11	5
09	49	52	47.38	0.02	1.12E+06	1	1	0.02	1.12E+06	1	1	0	1.12E+06	1	1
10	58	176	162	0.08	1.21E+06	15	11	0.06	1.21E+06	11	9	0.078	1.21E+06	15	11
11	68	108	60	9.50	1.12E+07	267	226	0.66	1.12E+07	18	8	2.547	1.12E+07	73	32
12	88	107	108.6	10.67	7.35E+06	391	359	0.98	7.35E+06	34	21	2.187	7.35E+06	79	47
13	108	452	26	22.59	6.09E+06	25,695	132	0.02	6.09E+06	16	3	0.094	8.34E+06	100	1
14	116	119	128.32	0.13	7.16E+04	1	1	0.14	7.16E+04	1	1	0.14	7.16E+04	1	1
15	133	164	26	0.02	1.16E+06	1	1	0.02	1.16E+06	1	1	0.015	1.16E+06	1	1
16	145	224	163	30.03	4.64E+06	79	63	24.92	4.64E+06	65	41	26.375	4.64E+06	69	53
17	152	211	57	0.02	1.09E+06	1	1	0.02	1.09E+06	1	1	0.031	1.09E+06	1	1
18	154	224	100	2.09	9.75E+04	33	26	2.47	9.75E+04	36	21	1.547	9.75E+04	25	18
19	156	263	125	3,103.63	3.15E+05	59,061	58,681	13.84	3.18E+05	264	91	5.031	3.19E+05	100	25
20	156	169	160.9	804.39	2.91E+05	16,699	14,514	2.30	3.02E+05	46	4	4.797	3.01E+05	100	26
21	186	359	96	140.34	1.08E+06	1,983	59	6.61	1.08E+06	88	33	7	1.08E+06	100	59
22	253	253	691	12.19	1.94E+06	1	1	11.69	1.94E+06	1	1	12.218	1.94E+06	1	1
23	271	524	77	176,722.45	4.26E+05	10,000,000	108	56.53	4.26E+05	3,153	55	1.844	4.26E+05	100	100
24	334	1245	68.53	10,435.67	1.41E+07	127,521	5,107	35.47	1.44E+07	434	263	8.453	1.45E+07	100	87
25	409	853	82	883,312.55	1.14E+06	10,000,000	3,138,047	143.52	1.14E+06	1,552	233	9.016	1.15E+06	100	70
26	468	605	394.07	8.38	4.53E+06	21	11	3.58	4.53E+06	9	5	8.375	4.53E+06	21	11
27	482	941	105	705,636.17	8.32E+05	10,000,000	8,428,498	221.83	7.53E+05	2,850	2,461	7.375	8.51E+05	100	58
28	577	2262	123	103.02	4.63E+06	389	180	18.03	4.63E+06	68	27	26.656	4.73E+06	100	99
29	617	753	43	0.75	3.46E+05	11	6	0.75	3.46E+05	11	6	0.766	3.46E+05	11	6
30	626	632	71.05	0.69	9.60E+05	5	2	1.28	9.60E+05	9	2	0.688	9.60E+05	5	2
31	706	908	17.92	116.22	3.40E+07	4,065	3,894	5.99	3.41E+07	233	7	2.828	3.49E+07	100	13
32	844	1685	112.2	3,089,722.84	7.09E+05	10,000,000	5,535,857	1,708.02	7.09E+05	5,393	108	31.093	7.09E+05	100	56
33	976	1009	72.36	0.28	4.22E+05	1	1	0.30	4.22E+05	1	1	0.281	4.22E+05	1	1
34	1206	4063	89	12.66	8.64E+05	147	71	9.42	8.64E+05	98	41	8.594	8.64E+05	100	71
35	1386	1857	81	37.88	1.79E+06	235	192	7.11	1.80E+06	43	5	16.203	1.81E+06	100	77
36	1451	4812	49.55	126,583.03	1.24E+06	781,737	11,235	1,028.47	1.43E+06	7,502	594	13.765	1.43E+06	100	98
37	1479	2069	27.85	17.28	9.99E+05	253	180	2.63	9.99E+05	38	26	6.875	9.99E+05	100	40
38	2025	16225	26.03	577,190.78	1.10E+06	2,839,444	65,507	39.31	1.29E+06	256	115	16.938	1.53E+06	100	1

TABLE 1. Performance of GNA, HGNA, and TGNA.

produce similar bounds. Figures 9(e) and 9(f) could have similar bounding behavior but are more interesting because they show the effect of link selection on GNA's performance. In 9(e) link (1,2) is removed to form the spanning tree and in 9(f) link (1,3) is removed. In 9(f), the optimal solution is found in the first iteration, whereas 9(e) will require multiple optimizations. Finally, the figures also hint that the duality gap may not be very large for inventory cost functions for an optimal value of a scalar multiple $u \geq 0$ for the constraint $S_1 - SI_2$. It is certainly possible to use the penalty function $u(S_1 - SI_2)$ in **SDP** since the cost functions remain separable, i.e. of the form $c_i(SI_i, S_i)$. However, the derivation of the correct multiplier to use is not immediately obvious.

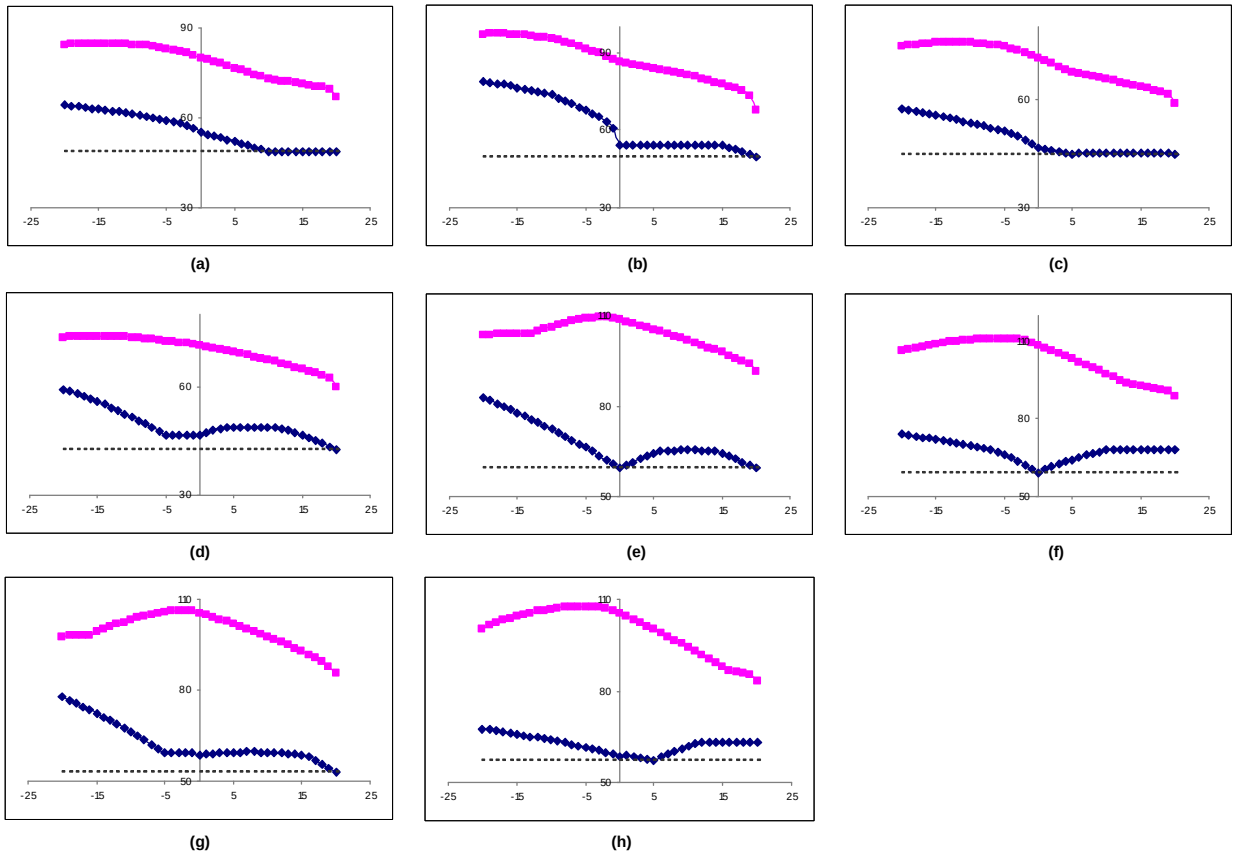

FIGURE 9. Example images of spanning tree solutions in the dual space for some network parameters. y, z are the horizontal and vertical axes respectively. Non-linear lines are the upper and lower envelopes of set H. The horizontal line is the cost function minimized when solving a spanning tree relaxation of **P**.

Figure 10 shows why HGNA lower-bounds the SI_i's only once. Each row of sub-figures in Figure 10 shows the GNA bounding behavior for one set of parameters. Each row shows that before, or after the first time S_2 is bounded, a solution close to the optimal has been found. This of course is not a proof, but the behavior is consistent across a larger set of examples than shown here. The tests in Tables 1 lend more weight to the observation.

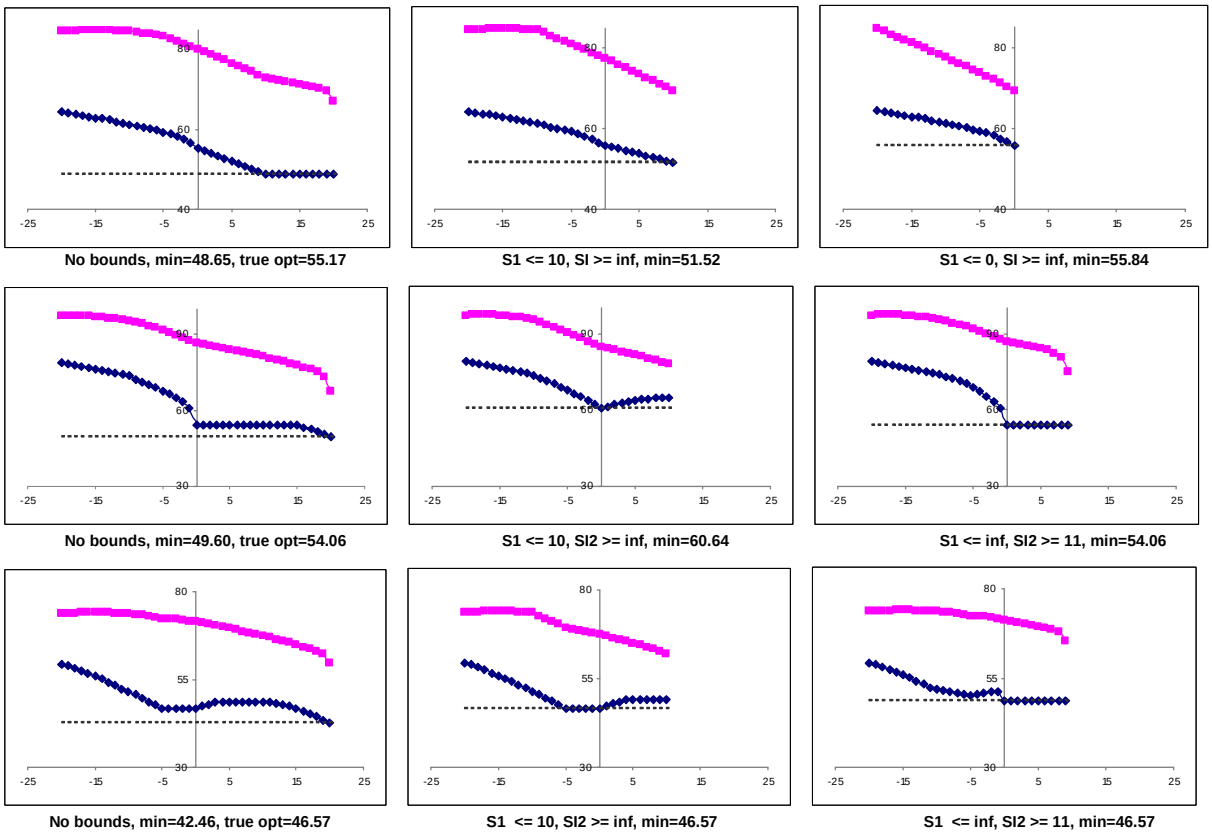

FIGURE 10. Examples of how a solution close to the optimal is usually found by the time the first SI_2 is bounded. Each row of sub-figures in above shows the GNA bounding behavior for one set of parameters; demonstrating that before or after the first time S_2 is bounded, a solution close to the optimal has been found.

8. **Proofs**

Theorem 3.1. SDP′ *correctly solves any instance of* $\mathbf{P}'$*, i.e.* $\mathbf{P}$ *with additional bounds of the form* $l_i \leq S_i \leq u_i$*,* $L_i \leq SI_i \leq U_i$*,* $i = 1, \ldots, n$*, if the underlying network for* $\mathbf{P}'$ *is a spanning tree.*

Proof. As in Section 2.5, we use stage k to refer to a stage with index k. Index k has the property that most one stage adjacent to stage k has a depth $p_k > k$. We call p_k k's *parent*. In fact, every stage in the tree except one, which we call the *root* stage, has exactly one *parent*. The *root* has no *parent*.

Define the stage sets N_k for each k as follows,

$$N_k = \{k\} \bigcup_{\substack{i<k \\ (i,k)\in\mathcal{A}}} N_i \bigcup_{\substack{j<k \\ (k,j)\in\mathcal{A}}} N_j.$$

And associate the sub-graph $G_k = (N_k, \mathcal{A}_k)$ with each k, where $\mathcal{A}_k = \{(i,j) \in \mathcal{A} : i \in N_k, j \in N_k\}$. Intuitively, G_k is the connected sub-tree that contains k after removing the link (k, p_k) or (p_k, k) from the original tree.

Also note the correspondence of the constraints of $\mathbf{P}'$ to the network structure, which makes this proof possible. The following three constraints are associated with each stage k: $S_k - SI_k \leq T_k$, $l_k \leq S_k \leq u_k$ and $L_k \leq SI_k \leq U_k$. The following constraint is associated with each link $(j,i) \in \mathcal{A}$: $S_j - SI_i \leq 0$. There are no other constraints in $\mathbf{P}'$ apart from integrality.

Define the family of optimization problems $\mathbf{P}_k$ for each stage k as follows:

$$
\begin{aligned}
\mathbf{P}_k: \quad \min \;\; & c_k(S_k, SI_k) \; + \sum_{i<k:\; i \in N_k} c_i(SI_i, S_i) \\
\text{s.t.} \;\; & S_i - SI_i \leq T_i, && \forall i \in N_k, \\
& l_i \leq S_i \leq u_i, && \forall i \in N_k, \\
& L_i \leq SI_i \leq U_i, && \forall i \in N_k, \\
& S_j - SI_i \leq 0, && \forall (j,i) \in \mathcal{A}_k, \\
& S_i, \; SI_i, \text{ integral}, && \forall i \in N_k.
\end{aligned}
$$

Let $\mathcal{C}_k$ denote the cost function and Ω_k denote the set of constraints of $\mathbf{P}_k$, and note that $\mathbf{P}_n = \mathbf{P}$.

If $(k, p_k) \in \mathcal{A}$, let $f_k^*(S)$ be the true optimal value of $\mathbf{P}_k$ if the service time for stage k is fixed to S. Similarly, if $(p_k, k) \in \mathcal{A}$, let $g_k^*(SI)$ be the true optimal value of $\mathbf{P}_k$ if the inbound service time for k is fixed to SI. Note that the optimal value to $\mathbf{P}$ can then be obtained as the $\min_{0 \leq SI \leq U_n} g_n^*(SI)$, if $k = n$ is the *root* stage for the entire tree.

We will show by induction that the functions $f_k(S)$ and $g_k(SI)$ generated by the dynamic program $\mathbf{SDP}'$ equal the optimal functions $f_k^*(S)$ and $g_k^*(SI)$ for every k.

The base case is to verify that for all stages k such that $N_k = \{k\}$, the functions $g_k(SI) = g_k^*(SI)$ and $f_k(S) = f_k^*(S)$, i.e. the functions produced by $\mathbf{SDP}'$ are optimal. This is straightforward to see.

Now consider any k and suppose that for all $i < k$ such that $i \in N_k$, $g_i(SI) = g_i^*(SI)$ or $f_i(S) = f_i^*(S)$, whichever is applicable for i. Then, if $(p_k, k) \in \mathcal{A}$, $g_k^*(SI) = \infty$ for all $SI < L_k$ or $SI > U_k$, and otherwise, for every SI satisfying $L_k \leq SI \leq U_k$, we can write

$$
\begin{aligned}
g_k^*(SI) &= \min_{\Omega_k} \left\{ c_k(SI, S_k) + \sum_{\substack{i<k: \\ i \in N_k}} c_i(SI_i, S_i) \right\}, \\
&= \min_{\Omega_k} \left\{ c_k(SI, S_k) + \sum_{\substack{i<k: \\ (i,k)\in\mathcal{A}_k}} \mathcal{C}_i + \sum_{\substack{j<k: \\ (k,j)\in\mathcal{A}_k}} \mathcal{C}_j \right\}, \\
&= \min_{\substack{SI - S_k + T_k \geq 0 \\ l_k \leq S_k \leq u_k}} \left\{ c_k(SI, S_k) + \sum_{\substack{i<k: \\ (i,k)\in\mathcal{A}_k}} \min_{\Omega_i \cup \{S_i \leq SI\}} \mathcal{C}_i + \sum_{\substack{j<k: \\ (k,j)\in\mathcal{A}_k}} \min_{\Omega_j \cup \{SI_j \geq S_k\}} \mathcal{C}_j \right\}, \\
&= \min_{l_k \leq S_k \leq \min(SI+T_k, u_k)} \left\{ c_k(SI, S_k) + \sum_{\substack{i<k: \\ (i,k)\in\mathcal{A}_k}} \min_{S_i \leq SI} f_i^*(S_i) + \sum_{\substack{j<k: \\ (k,j)\in\mathcal{A}_k}} \min_{SI_j \geq S_k} g_j^*(SI_j) \right\}, \\
&= \min_{l_k \leq S_k \leq \min(SI+T_k, u_k)} \left\{ c_k(SI, S) + \sum_{\substack{i<k: \\ (i,k)\in\mathcal{A}_k}} \min_{x \leq SI} f_i(x) + \sum_{\substack{j<k: \\ (k,j)\in\mathcal{A}_k}} \min_{y \geq S} g_j(y) \right\}, \\
&= g_k(SI).
\end{aligned}
$$

The first step above is simply the definition of $g_k^*(SI)$. In the second step, we use the fact that the stages of the children sub-trees of k are disjoint to separate the cost terms. The third step uses the separability of the constraints among the children sub-trees to decompose the constraint set Ω_k. Step four uses definitions of f_i^* and g_j^* and step five uses the inductive hypothesis. In step five, we also rewrite $S_k = S$ and use x, y for the decision variables S_i, SI_j, to better differentiate the variables with respect to which the minimization is being carried out.

The proof $f_k(S) = f_k^*(S)$ when $(k, p_k) \in \mathcal{A}$ follows exactly the same operations as above. □

Proposition 3.3. *In any iteration of $\mathcal{R}$, for any infeasible link $(j,i) \in \mathcal{A}$, $F_{\Omega \cup \{S_j \leq \bar{S}\}} \cup F_{\Omega \cup \{SI_k > \bar{S}, \forall k:(j,k)\in\mathcal{A}\}} \supset F_\Omega$.*

Proof. $F_\Omega = F_{\Omega \cup \{S_j \leq \bar{S}\}} \cup F_{\Omega \cup \{S_j > \bar{S}\}}$. But note that if $S_j > \bar{S}$, then $SI_k \leq \bar{S}$ for any $(j,k) \in \mathcal{A}$ cannot be in F_Ω. Therefore $F_{\Omega \cup \{S_j > \bar{S}\}} \subset F_{\Omega \cup \{SI_k > \bar{S}, \forall k:(j,k)\in\mathcal{A}\}}$. □

References

Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (1993). *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, 2 ed. ISBN 0-471-55793-5.

Bossert, J. M. & Willems, S. (2007). 'A Periodic Review Modeling Approach for Guaranteed Service Supply Chains'. *Interfaces*, **37**(5), pp. 420–435.

Ettl, M., Feigin, G. E., Lin, G. Y., & Yao, D. D. (2000). 'A Supply Network Model with Base-Stock Control and Service Requirements'. *Operations Research*, **49**(2), pp. 216–232.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. ISBN 0201157675.

Graves, S. C. & Willems, S. P. (1996). 'Strategic Safety Stock Placement in Supply Chains'. Proceedings of the 1996 MSOM Conference. Hanover, NH.

Graves, S. C. & Willems, S. P. (2000). 'Optimizing Strategic Safety Stock Placement in Supply Chains'. *Manufacturing & Service Operations Management*, **2**(1), pp. 68–83.

Graves, S. C. & Willems, S. P. (2003a). 'Erratum: Optimizing Strategic Safety Stock Placement in Supply Chains'. *Manufacturing & Service Operations Management*, **5**(2), pp. 176–177.

Graves, S. C. & Willems, S. P. (2003b). 'Supply Chain Design: Safety Stock Placement and Supply Chain Configuration'. Handbooks in Oper. Res. And Management Sci. Vol. 11, Supply Chain Management: Design, Coordination and Operation. A. G. de Kok and S. C. Graves (eds.), North-Holland Publishing Company, Amsterdam, The Netherlands, chap. 3.

Guisewite, G. M. & Pardalos, P. M. (1990). 'Minimum Concave-Cost Network Flow Problems: Applications, Complexity and Algorithms'. *Annals of Operations Research*, **25**, pp. 75–100.

Humair, S. & Willems, S. P. (2006). 'Optimizing Strategic Safety Stock Placement in Supply Chains with Clusters of Commonality'. *Operations Research*, **54**(4), pp. 725–742.

Humair, S. & Willems, S. P. (2008). 'Optimizing Strategic Safety Stock Placement in General Acyclic Networks'. *Working Paper*.

Inderfurth, K. (1991). 'Safety Stock Optimization in Multi-Stage Inventory Systems'. *International Journal of Production Economics*, **24**, pp. 103–113.

Inderfurth, K. & Minner, S. (1998). 'Safety Stocks in Multi-Stage Inventory Systems under Different Service Levels'. *European Journal of Operations Research*, **106**, pp. 57–73.

Karimi, I. A., Shu, J., & Song, M. (2004). 'Heuristic Methods for Inventory Placement in General Acyclic Supply Chains'. *Working Paper*, p. 25.

Kimball, G. E. (1988). 'General Principles of Inventory Control'. *Journal of Manufacturing and Operations Management*, **1**, pp. 119–130.

Lee, H. L. & Billington, C. (1993). 'Material Management in Decentralized Supply Chains'. *Operations Research*, **41**, pp. 835–847.

Lesnaia, E. (2004). 'Optimizing Safety Stock Placement in General Network Supply Chains'. Ph.D. thesis, Massachusetts Insitute of Technology.

Magnanti, T. L., Shen, Z.-J. M., Shu, J., Simchi-Levi, D., & Teo, C.-P. (2006). 'Inventory Placement in Acyclic Supply Chain Networks'. *Operations Research Letters*, **34**(2), pp. 228–238.

Minner, S. (2000). *Strategic Safety Stocks in Supply Chains*. Lecture Notes in Economics and Mathematical Systems; 490. Springer-Verlag, Berlin.

Nemhauser, G. L. & Woolsey, L. A. (1988). *Integer and Combinatorial Optimization*. John Wiley & Sons. ISBN 0-471-82819-X.

Simchi-Levi, D. & Zhao, Y. (2005). 'Safety Stock Positioning in Supply Chains with Stochastic Lead Times'. *Manufacturing & Service Operations Management*, **7**(4), pp. 295–318.

Simpson, K. F. (1958). 'In-Process Inventories'. *Operations Research*, **6**, pp. 863–873.

Willems, S. (2008). 'Real-World Multiechelon Supply Chains Used for Inventory Optimization'. *Manufacturing & Service Operations Management*. 5 pages.

Zipkin, P. (2000). *Foundations of Inventory Management*. McGraw-Hill.