Management Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

From Trees to Treewidth: Inventory Management in Complex Supply Chain Networks

Philippe Blaettchen, Andre P. Calmon, Georgina Hall, Mohit Tawarmalani

To cite this article:
Philippe Blaettchen, Andre P. Calmon, Georgina Hall, Mohit Tawarmalani (2026) From Trees to Treewidth: Inventory Management in Complex Supply Chain Networks. Management Science

Published online in Articles in Advance 10 Sep 2026

. https://doi.org/10.1287/mnsc.2025.03155

https://pubsonline.informs.org/journal/mnsc

MANAGEMENT SCIENCE
Articles in Advance, pp. 1–18
ISSN 0025-1909 (print), ISSN 1526-5501 (online)

From Trees to Treewidth: Inventory Management in Complex Supply Chain Networks

Philippe Blaettchen,[a,*] **Andre P. Calmon,**[b] **Georgina Hall,**[c] **Mohit Tawarmalani**[d]

[a] Lee Kong Chian School of Business, Singapore Management University, 178899 Singapore; [b] Scheller College of Business, Georgia Institute of Technology, Atlanta, Georgia 30308; [c] Decision Sciences, INSEAD, 77305 Fontainebleau, France; [d] Mitch Daniels School of Business, Purdue University, West Lafayette, Indiana 47907

*Corresponding author

pblaettchen@smu.edu.sg (P. Blaettchen), andre.calmon@gatech.edu (A. P. Calmon), georgina.hall@insead.edu (G. Hall), mtawarma@purdue.edu (M. Tawarmalani)

https://orcid.org/0000-0002-4253-6079 (P. Blaettchen), https://orcid.org/0000-0002-9017-1625 (A. P. Calmon), https://orcid.org/0000-0003-4580-3897 (G. Hall), https://orcid.org/0000-0003-3085-0084 (M. Tawarmalani)

Received: July 29, 2025
Revised: February 6, 2026; May 15, 2026
Accepted: May 31, 2026
Published Online in Articles in Advance: September 10, 2026

https://doi.org/10.1287/mnsc.2025.03155

Abstract. We propose an exact linear programming (LP)-based solution approach to the Guaranteed Service Model (GSM), one of the most widely applied models for optimizing safety stock placement in supply chain networks. Our approach handles any directed acyclic network and any cost function that depends on a stage's incoming and outgoing service times. It scales polynomially in the number of nodes n in the network, pseudo-polynomially with respect to the bit size of the maximum replenishment time M, and (for fixed M) exponentially in its *treewidth*, which quantifies how "tree-like" a network is and can be much smaller than n. This contrasts with existing approaches, which scale exponentially in n. The proof of exactness relies crucially on showing that the join of transportation-like polytopes remains integral and is more broadly applicable to other Operations Management problems. In addition to an exact formulation, our LP-based approach enables a practical solution strategy for the GSM built on a hierarchy of LP relaxations. These relaxations provide valid lower bounds, certify optimality when integral, and can strengthen existing exact methods. In our computational study, the smallest relaxation already recovers an optimal GSM solution on every real-world benchmark instance, leading to substantial speed-ups over the state-of-the-art exact algorithm and commercial general-purpose solvers. The framework also supports sensitivity analysis and accommodates additional operational constraints. Overall, our approach builds a new bridge between Operations Management and Computer Science, providing new theoretical foundations and practical tools for managing safety stocks in complex modern supply chain networks.

History: Accepted by Jeannette Song, operations management.
Funding: A. P. Calmon acknowledges financial support provided by the Ray C. Anderson Center for Sustainable Business.
Supplemental Material: The online appendix and data files are available at https://doi.org/10.1287/mnsc.2025.03155.

Keywords: supply chain management • inventory management • algorithms • computational complexity • linear programming

1. Introduction

The optimization of NP-hard problems over supply chain networks represents a cornerstone of Operations Management research and remains one of the most active areas of exchange between academia and industry. An example is the Guaranteed Service Model (GSM; Graves and Willems 2000), a celebrated framework for optimizing safety stock placement. It is popular among practitioners, with implementations at Hewlett-Packard, Procter & Gamble, and Intel recognized as Franz Edelman Award finalists (Eruguz et al. 2016, Schoenmeyr and Graves 2022). The GSM is NP-hard (Lesnaia et al. 2005) and can be formulated as an integer program with a nonconvex objective.

For decades, research on the GSM has largely progressed along three, sometimes overlapping, directions. The first involves designing custom optimization algorithms, typically based on dynamic programming (DP), which provably return optimal solutions. Authors either focus on specific network structures and/or objective functions, enabling them to obtain (pseudo-)polynomial-time algorithms (e.g., Graves and Willems 2000, Lesnaia 2004); or they allow for general network structures and functions, in which case the complexity of existing

optimization methods is exponential in the number of nodes in the network (Humair and Willems 2011). Although this has been a rich research avenue, many GSM optimization approaches feel artisanal, and the significant challenge of devising and implementing these bespoke algorithms limits their widespread adoption.

The second direction relies on heuristics, sometimes linked to formal theoretical analyses (e.g., Minner 2000, Shu and Karimi 2009, Li and Jiang 2012). These heuristics come with limited optimality guarantees, focusing instead on computational performance. They suffer from similar drawbacks to the exact solution approaches, being difficult to construct and sometimes to implement.

The third direction is to use powerful commercial optimizers, such as BARON (Sahinidis 1996), to solve the original integer program with a nonconvex objective (e.g., Moncayo-Martínez and Mastrocinque 2025). Although this is straightforward to implement, general-purpose tools frequently struggle to efficiently solve even relatively simple problem instances, as we discuss in Section 6.

We propose an exact solution approach to the GSM that combines the positives of all three previous research directions: (i) it applies to a general setting—that is, to any acyclic network and cost function—and improves, from a computational complexity perspective, on the current state-of-the-art; (ii) it recovers existing complexity guarantees when specific network structures and/or objective functions are assumed; (iii) it is easy to implement, allows for seamless integration of new constraints, and outperforms existing solution approaches in practice; and (iv) it allows for the easy construction of *principled* heuristics for the problem.

To derive our solution approach, we leverage a key concept from Graph Theory and Theoretical Computer Science—namely, the *treewidth* of a graph, which encodes how tree-like it is. Although it is well known in Computer Science and Combinatorial Optimization that the computational complexity of many NP-hard optimization problems over graphs (Bodlaender 1993) is driven by treewidth (e.g., vertex cover), such connections have remained unexplored for popular supply chain problems.[1] We bridge this divide by proposing the first treewidth-based optimization approach to the GSM. Our contributions are the following:

- We provide an exact linear programming (LP) reformulation of the GSM in the special case where the underlying acyclic supply chain network is a tree (Section 3). Through this formulation, we recover existing complexity results for the GSM on trees.
- We then provide an exact LP reformulation of the GSM that applies to any acyclic supply chain network and any objective function (Section 4). Its size scales polynomially in the number of nodes in the network, pseudo-polynomially with respect to (wrt) the bit size of the maximum replenishment time, but exponentially in the network's treewidth. The proof of exactness of our reformulation involves showing that "gluing" together transportation-like polytopes maintains integrality.
- We discuss how this technique contributes to the treewidth literature and may apply to other Operations Management problems (Section 5). We further demonstrate that our approach offers a unified framework that generalizes and improves upon all prior solution approaches to the GSM, including the state-of-the-art.
- We showcase several practical advantages of our approach (Section 6). In particular, we propose a framework for solving the GSM in practice which leverages the fact that the exact LP reformulation of the GSM yields a hierarchy of increasingly tight (and correspondingly expensive) LP relaxations. The lower bounds obtained are often sharp and allow certifying optimality when integral. The first level of the hierarchy is sufficient to recover the optimal GSM solution for every real-world supply chain in Willems (2008). Our approach thus outperforms, using standard LP solvers, the state-of-the-art algorithm of Humair and Willems (2011) and the commercial solver BARON by more than an order of magnitude, while modern graphics processing unit (GPU)-based LP solvers offer the potential for further speed-ups. Finally, we discuss other managerial advantages of our framework, including how it can support sensitivity analyses and seamlessly incorporate additional constraints.

We briefly recap the GSM in Section 2 before moving on to our central results.

2. The Guaranteed Service Model

We present the Guaranteed Service Model, a framework for discrete-time multiechelon inventory management without production capacity constraints, as introduced by Graves and Willems (2000). All notation used in the main draft is summarized in Table EC.1 in Online Appendix A. For a complete survey of the GSM, we refer the reader to Eruguz et al. (2016).

Consider an acyclic directed graph $G = (V, E)$ representing a supply chain network. Each node $j \in V$ is a supply chain stage that can hold inventory, and each arc $(i, j) \in E$ indicates that node i is an immediate supplier of node j. In other words, each unit produced at j requires one unit of input from i.[2] Thus, node j places orders at node i as needed, and i fulfills those orders. We denote the total number of nodes in the supply chain network by $n = |V|$.

Let $L \subseteq V$ be the set of *external demand nodes*—that is, nodes with no outgoing arcs. These nodes fulfill orders from outside customers, which arrive in discrete time periods $t \in \mathbb{N}$. We assume that demand at each node is random and independently and identically distributed (i.i.d.) across periods. Importantly, demand *across*

nodes need not be independent, as we discuss in Section 2.1.

Every node operates a discrete-time periodic review base stock policy[3] with a common review period normalized to length one. When a node receives an order, it immediately orders the necessary inputs from its supplier(s) to produce the relevant quantity. Hence, after an external demand node $j \in L$ receives customer orders in period $t \in \mathbb{N}$, it immediately induces orders in *all* nodes upstream of j on paths that include j. The propagation of orders follows from the base stock policy and does not require explicit sharing of any information beyond orders. In each period t, node j therefore faces stochastic demand D_j^t. This represents external demand if $j \in L$ or the sum of propagated demands from downstream nodes if $j \notin L$. Furthermore, node j has a deterministic processing time $T_j \in \mathbb{N}$, independent of the quantity being processed. Later, in Section 5.2, we discuss how our approach is general enough also to handle stochastic processing times.

2.1. Service Times and Demand Bounds

Each node $j \in V$ has two decision variables: the node's *guaranteed outbound service time* S_j and the node's *guaranteed inbound service time* SI_j.

The *guaranteed outbound service time* S_j is the maximum number of periods within which node j *guarantees* to fulfill any demand below a prespecified deterministic upper bound. For a time interval τ, denote this bound by $\bar{D}_j(\tau)$. A key assumption is that any demand exceeding the bound is dealt with through measures outside the model, enabling the GSM to "ignore" stochastic demand fluctuations below the bound and demand above. The assumption is supported by practice; many companies offer customers a guaranteed delivery time, provided demand is below a specific limit.

The choice of the functional form of $\bar{D}_j$ is somewhat arbitrary: it is generally assumed to be concave and increasing, reflecting pooling effects over time, but managers can tailor it to their specific application. In simulations, we use the demand bound from Graves and Willems (2000), who assume that an external demand node $j \in L$ faces i.i.d. normally distributed demand with mean μ_j and variance σ_j^2, and sets $\bar{D}_j(\tau) = \tau\mu_j + k\sigma_j\sqrt{\tau}$. The parameter k reflects how often managers are willing to resort to "extraordinary measures" to satisfy demand. For an internal node $j \notin L$, a common demand bound is simply the sum of the demand bounds of the successors of node j—that is, $\bar{D}_j(\tau) = \sum_{(j,l)\in E} \bar{D}_l(\tau)$. More complex bounds that account for demand-pooling effects and cross-node dependencies can also be used (Eruguz et al. 2016). We emphasize that our solution approach to the GSM does not require a specific functional form of $\bar{D}_j$, only that it be increasing. When all decision variables have upper bounds, we can further remove this assumption, if needed.

The GSM also assumes that there is an upper limit on external demand nodes' guaranteed service time. Namely, for $j \in L$, a manager prespecifies an upper bound $d_j \in \mathbb{N}$ and requires $S_j \le d_j$. This bound reflects the maximum time a customer is willing to wait for demand fulfillment.

The *guaranteed inbound service time* SI_j is the maximum number of periods required for node j to receive all the necessary input from its immediate predecessors. The guaranteed inbound service time must be at least as large as all predecessors' guaranteed outbound service times—that is, $SI_j \ge \max\{S_i : (i,j) \in E\}$. Without loss of generality, we assume that $S_j \le SI_j + T_j$. If S_j exceeded $SI_j + T_j$, production and processing at node j would always keep pace with demand.

2.2. Inventory Dynamics

Let $I_j(t)$ be the on-hand inventory of node j at time t. To ensure that its outbound service time is met when demand is within its upper bound, node j holds a certain quantity of its output in inventory: its *base stock* B_j. Clearly, the base stock required to fulfill demand within a guaranteed service time is a function of the time in which node j needs to provide outputs (i.e., fulfill demands), without being able to catch up on its production (including receiving inputs).

Assume, without loss of generality, that $I_j(0) = B_j$ and recall that D_j^m denotes the stochastic demand faced by node j in period m. Then, following Klosterhalfen et al. (2014),

$$I_j(t) = B_j - \sum_{m=1}^{t-S_j} D_j^m + \sum_{m=1}^{t-T_j-SI_j} D_j^m.$$

The first sum represents all demands j must fulfill up to S_j periods before the current period. The second sum represents the total output j can produce up to the current period, based on its inbound service time guarantees and its own processing time $(SI_j + T_j)$.

Meeting node j's service guarantee imposes $I_j(t) \ge 0$. Recalling $S_j \le SI_j + T_j$, this requires

$$B_j \ge \sum_{m=1}^{t-S_j} D_j^m - \sum_{m=1}^{t-T_j-SI_j} D_j^m = \sum_{m=t-T_j-SI_j+1}^{t-S_j} D_j^m \quad \forall t = 1,2,\ldots.$$

Because $\sum_{m=t-T_j-SI_j+1}^{t-S_j} D_j^m \le \bar{D}_j(SI_j + T_j - S_j)$ by assumption, $B_j := \bar{D}_j(SI_j + T_j - S_j)$ in the GSM to ensure that node j fulfills its service time guarantee in all periods.

To compute inventory holding costs, note that the long-term average inventory held by j is:

$$\lim_{t\to\infty}\frac{1}{t}\sum_{m=1}^{t} I_j(m) = \bar{D}_j(SI_j + T_j - S_j) - \lim_{t\to\infty}\frac{1}{t}\sum_{m=1}^{t-S_j} D_j^m + \lim_{t\to\infty}\frac{1}{t}\sum_{m=1}^{t-T_j-SI_j} D_j^m = \bar{D}_j(SI_j + T_j - S_j) - \mu_j\cdot(SI_j + T_j - S_j).$$

Multiplying the average safety stock by the holding cost per unit, h_j, at node j, and summing across all $j \in V$ gives the expected total inventory cost.

2.3. The GSM as an Optimization Problem

The GSM can then be formulated as the following optimization problem:

$$\begin{aligned}\min_{S_j, SI_j}\ & \sum_{j\in V} f_j(SI_j - S_j),\\ \text{s.t.}\ & S_j - SI_j \le T_j,\ j\in V,\\ & SI_j - S_i \ge 0,\ (i,j)\in E,\\ & S_j \le d_j,\ j\in L,\ S_j, SI_j \ge 0 \text{ and integer for } j\in V,\end{aligned} \tag{1}$$

where $f_j(SI_j - S_j) = h_j[\bar{D}_j(SI_j + T_j - S_j) - \mu_j\cdot(SI_j + T_j - S_j)]$ and $\bar{D}_j(\cdot)$ is generally assumed concave and increasing. In the rest of the draft, we consider a more general formulation:

$$\begin{aligned}\min_{S_j, SI_j}\ & \sum_{j\in V} f_j(SI_j - S_j), && (2)\\ \text{s.t.}\ & S_j - SI_j \le T_j,\ j\in V, && (2a_j)\\ & SI_j - S_i \ge 0,\ (i,j)\in E, && (2b_{ij})\\ & 0 \le S_j \le \Gamma_j \text{ and } S_j \text{ integer for } j\in V, && (2c_j)\\ & 0 \le SI_j \le M_j \text{ and } SI_j \text{ integer for } j\in V, && (2d_j)\end{aligned}$$

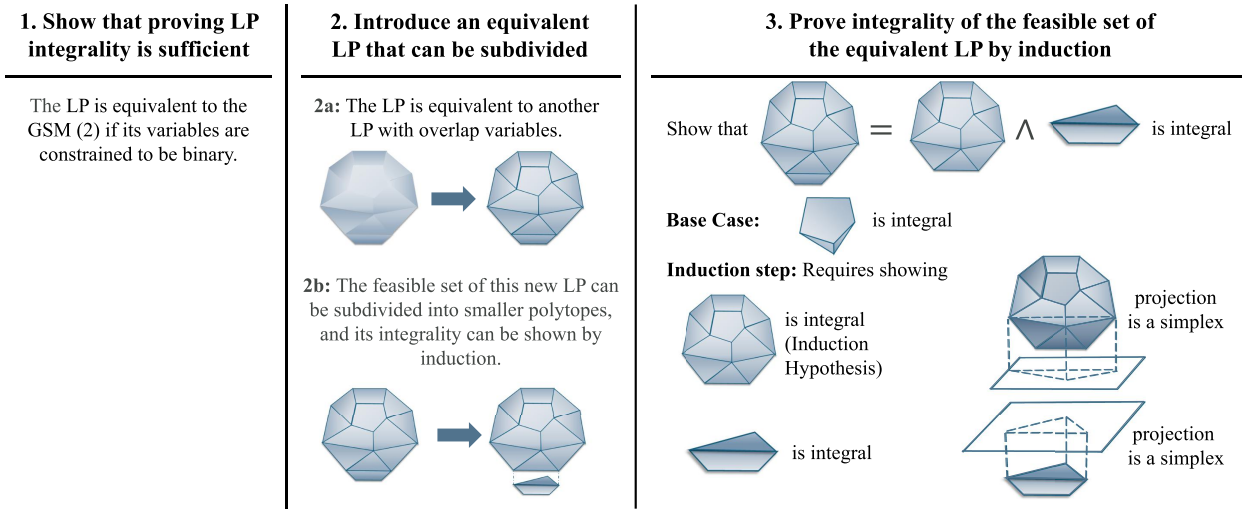
where Γ_j and M_j are upper bounds on S_j and SI_j, and f_j has any functional form—that is, it does not need to be concave, nor increasing, nor closed-form. Note that (2) is a generalization of (1) (Graves and Willems 2000). To see this, let $paths(G,j)$ be the set of directed paths in G that end at node j and define the maximum replenishment time $M_j = \max_{p\in paths(G,j)}\sum_{i\in p} T_i$ (with $M = \max_{j\in V} M_j$ for later purposes). Because M_j is the total processing time required to produce all the inputs needed at node j and the output at j, and f_j is increasing in (1), one can show $SI_j \le M_j - T_j$, which implies $S_j \le M_j$. Letting $\Gamma_j = \min\{d_j, M_j\}$ for $j\in L$ and $\Gamma_j = M_j$ for $j\notin L$, we obtain the conclusion. Crucially, the more general Formulation (2) with arbitrary f_j has the advantage of allowing the incorporation of important phenomena such as variable stage times and nonnested review periods into the model (Humair and Willems 2011; see also Section 5.2). Unsurprisingly, given an arbitrary acyclic graph structure, (2) is an NP-hard problem (Lesnaia et al. 2005).

3. Solving the GSM on a Tree via Linear Programming

We show that if $G = (V,E)$ is a tree, the GSM can be solved through an LP with $O(nM^2)$ variables and constraints, where n is the number of nodes in G and M is an upper bound on the total processing time, defined in Section 2.3. Our main result here is Theorem 1. A high-level overview of its proof is given after the theorem and in Figure 1, with a full proof provided in Online Appendix C.

We introduce two sets of decision variables to define the LP for the tree case. These should be interpreted as *indicator functions of the values taken on by pairs of decision variables in* (2). More formally, for each node $j\in V$, let $r^j_{k\ell} = 1_{S_j=k, SI_j=\ell}$—that is, $r^j_{k\ell}$ is one if $SI_j = \ell$ and $S_j = k$,

Figure 1. (Color online) Proof that the LP Solves the GSM (2)



Notes. For the proof of Theorem 1, we use LP (4). For the proof of Theorem 2, we use LP (12).

and zero otherwise. Furthermore, for each edge $(i,j) \in E$, let $q^{ij}_{k\ell} = 1_{S_i=k, SI_j=\ell}$. Because $|E| = n-1$ in a tree, and M_j, $\Gamma_j \le M\ \forall j \in V$, the resulting sets of decision variables, $\{r^j_{k\ell}\}_{k=0,\dots,\Gamma_j,\,\ell=0,\dots,M_j}$ for $j \in V$ and $\{q^{ij}_{k\ell}\}_{k=0,\dots,\Gamma_i,\,\ell=0,\dots,M_j}$ for $(i,j) \in E$, have a combined size in $O(nM^2)$. We further define the set

$$\mathcal{S}_{ij} = \{(k,\ell) \in \{0,\dots,\Gamma_i\} \times \{0,\dots,M_j\} \mid k > \ell\}, \text{ for } (i,j) \in E. \tag{3}$$

For $(i,j) \in E$, this can be interpreted as the set of values (k,ℓ) of (S_i, SI_j) with $k > \ell$, or equivalently, $S_i > SI_j$. In other words, *no solution* $(S_j, SI_j)_{j\in V}$ to (2) should take these values, given Constraint ($2b_{ij}$). Finally, we use $\Delta_n = \{(x_0, x_1, \dots, x_n) \mid \sum_{i=0}^n x_i = 1,\ x_i \ge 0, i = 0,\dots,n\}$ to denote the n-dimensional standard simplex, sometimes dropping the subscript n when the dimension is clear. We are now ready to introduce the GSM's LP formulation on trees:

Theorem 1. *When $G = (V,E)$ is a tree, the Guaranteed Service Model given in (2) can be solved via the following linear program with $O(nM^2)$ variables and $O(nM^2)$ constraints*:

$$\min \sum_{j\in V} \sum_{\ell=0}^{M_j} \sum_{k=0}^{\Gamma_j} f_j(\ell - k) r^j_{k\ell}, \tag{4}$$

$$\text{s.t. } r^j_{k\ell} \le \left\lfloor \frac{T_j}{k-\ell} \right\rfloor,\ \forall k = 0,\dots,\Gamma_j,\ \ell = 0,\dots,M_j,\ k > \ell,\ j \in V, \tag{4a_j}$$

$$q^{ij}_{k\ell} = 0,\ \forall (k,\ell) \in \mathcal{S}_{ij},\ \forall (i,j) \in E, \tag{4b_{ij}}$$

$$\sum_{k=0}^{\Gamma_j} r^j_{k\ell} = \sum_{k=0}^{\Gamma_i} q^{ij}_{k\ell},\ \forall \ell = 0,\dots,M_j,\ \forall j \in V \text{ s.t. } (i,j) \in E, \tag{4c_j}$$

$$\sum_{\ell=0}^{M_i} r^i_{k\ell} = \sum_{\ell=0}^{M_j} q^{ij}_{k\ell},\ \forall k = 0,\dots,\Gamma_i,\ \forall i \in V \text{ s.t. } (i,j) \in E, \tag{4d_i}$$

$$\{r^j_{k\ell}\}_{k,\ell} \in \Delta,\ \forall j \in V, \tag{4e_j}$$

$$\{q^{ij}_{k\ell}\}_{k,\ell} \in \Delta,\ \forall (i,j) \in E, \tag{4f_{ij}}$$

where the decision variables are

$$\{r^j_{k\ell}\}_{k=0,\dots,\Gamma_j,\ \ell=0,\dots,M_j,\ j\in V} \text{ and } \{q^{ij}_{k\ell}\}_{k=0,\dots,\Gamma_i,\ \ell=0,\dots,M_j,\ (i,j)\in E}.$$

For each node $j \in V$, let SI^*_j and S^*_j denote the optimal inbound and outbound service times in (2), and let $r^{j*}_{k\ell}$ denote the optimal value of $r^j_{k\ell}$ in (4). Then,

$$SI^*_j = \sum_{\ell=0}^{M_j} \left(\ell \cdot \sum_{k=0}^{\Gamma_j} r^{j*}_{k\ell} \right) \text{ and } S^*_j = \sum_{k=0}^{\Gamma_j} \left(k \cdot \sum_{\ell=0}^{M_j} r^{j*}_{k\ell} \right), \text{ for } j \in V. \tag{5}$$

It is straightforward to see that (4) is an LP, because the constraints are linear in the decision variables and the objective is linear in $\{r^j_{k\ell}\}_{k,\ell}$. In particular, $f_j(\ell - k)$ is simply f_j evaluated at $\ell - k$, so no specific functional form of f_j needs to be assumed.

Remark 1. Theorem 1 can also be derived from known results on Markov random fields.[4] Specifically, Wainwright et al. (2005, corollary 1) implies that the local marginal polytope $LOCAL(H)$ is integral whenever H is a tree. If we drop constraints ($4a_j$) and ($4b_{ij}$) from (4), then, it can be shown, after eliminating some variables, that the resulting feasible region is $LOCAL(G')$, where G' is the intersection graph associated with (2) over G (see Definition 2). Proposition 2 shows that G' is a tree whenever G is a tree. Hence, $LOCAL(G')$ is integral by Wainwright et al. (2005, corollary 1). Adding back Constraints ($4a_j$) and ($4b_{ij}$) sets to zero exactly those marginal coordinates corresponding to locally infeasible GSM configurations. Because these coordinates are nonnegative, this restriction is the intersection of $LOCAL(G')$ with coordinate faces and is therefore itself a face of $LOCAL(G')$. Because every face of an integral polytope is integral, the feasible region of (4) is integral. Theorem 1 then follows from Wainwright et al. (2005, corollary 1), this facial argument, and Proposition EC.1 in the Online Appendix. We retain our self-contained proof, summarized in Figure 1, because its structural insights allow us to extend Theorem 1 beyond trees to arbitrary acyclic graphs.

We now provide a sketch of the proof of Theorem 1, which follows the steps outlined in Figure 1. First, notice that (2) is equivalent to (4) if the variables of (4) are binary: in this case, $r^j_{k\ell}$ and $q^{ij}_{k\ell}$ can be set equal to the indicator functions $1_{S_j=k, SI_j=\ell}$ and $1_{S_i=k, SI_j=\ell}$, respectively. Constraints ($4a_j$) enforce that, if $T_j < k - \ell$ for some $k > \ell$ and $j \in V$, then $r^j_{k\ell} = 1_{S_j=k, SI_j=\ell} = 0$. In other words, (S_j, SI_j) are precluded from taking on values (k,ℓ) with $T_j < k - \ell$, or, equivalently $T_j < S_j - SI_j$—that is, (S_j, SI_j) need to satisfy ($2a_j$). Similarly, for $(i,j) \in E$, the Constraint ($4b_{ij}$) forces $q^{ij}_{k\ell} = 1_{S_i=k, SI_j=\ell} = 0$ when $k > \ell$. This precludes $S_i > SI_j$, or equivalently, (S_i, SI_j) need to satisfy $S_i \le SI_j$—that is, ($2b_{ij}$). Constraints ($4c_j$) ensure consistent values of $SI_j, j \in V$, whether they are computed through $r^j_{k\ell}$ or $q^{ij}_{k\ell}$: we have $\sum_{k=0}^{\Gamma_j} r^j_{k\ell} = 1_{SI_j=\ell} = \sum_{k=0}^{\Gamma_i} q^{ij}_{k\ell}$ for all ℓ. Constraints ($4d_i$) play a similar role for $S_i, i \in V$. Constraints ($4e_j$)–($4f_{ij}$) are necessary for $\{r^j_{k\ell}\}_{k,\ell}$ and $\{q^{ij}_{k\ell}\}_{k,\ell}$ to be indicator functions. The second step of the proof is to introduce a new linear program, with additional decision variables ("overlap" variables), which is equivalent to (4), and then show integrality of its feasible set. This is easier with the new linear program, because its feasible set naturally decomposes into the *join* of smaller polytopes (see

Definition EC.2 in the Online Appendix). We can then use induction to show its integrality, as done in Step 3 of the proof. Showing that the induction step holds is the main difficulty here. Online Appendix C is organized to mirror this outline—see Figure EC.1, which mirrors Figure 1, but summarizes the dependency structure among the propositions and lemmas employed.

4. Optimizing the GSM on a Network with Bounded Treewidth

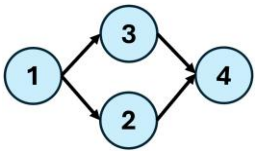
We consider the general case where G has *bounded treewidth*. We show that, when the treewidth of G is fixed, (2) can be recast as an LP of size polynomial in the number of nodes in G and pseudo-polynomial wrt the bit size of the maximum replenishment time. We thus identify G's treewidth as one of the main drivers of the GSM's solving complexity. Section 4.1 highlights why (4) does not work for nontree networks; Section 4.2 introduces key concepts; Section 4.3 reinterprets (4) using these concepts. This approach is then generalized in Section 4.4 to arbitrary networks.

4.1. From Trees to Graphs with Bounded Treewidth: An Illustrative Example

The LP formulation for trees in (4) is powerful due to its simplicity and efficiency: it only creates interaction variables $\{q^{ij}_{kl}\}$ for pairs of service times linked by an edge $(i,j) \in E$. This is sufficient for a tree, where all dependencies can be expressed along simple paths. On general networks, however, different paths can rejoin and create higher-order interactions between nodes that are not adjacent. In those cases, (4) only "sees" edge-wise interactions and may therefore miss critical cross-path couplings, leading to an LP with nonbinary vertices that is a strict relaxation of (2).

This limitation of (4) is best illustrated through a concrete example. Consider the "diamond" network in Figure 2, one of the simplest nontree network structures. The GSM (2) requires that $S_1 \le \min\{SI_2, SI_3\}$ and $SI_4 \ge \max\{S_2, S_3\}$—that is, S_1 and SI_4 are coupled *jointly* through the service times at both 2 and 3. In contrast, the tree-based LP (4) only creates interaction variables along edges: it tracks (S_1, SI_2) and (S_2, SI_4) along path $1 \to 2 \to 4$, and (S_1, SI_3) and (S_3, SI_4) along path $1 \to 3 \to 4$, but it never jointly tracks (SI_2, SI_3, S_2, S_3). As a result, we can obtain optimal solutions to the LP (4) which are infeasible for (2) and provide strict lower bounds on the optimal value.

Figure 2. (Color online) A Diamond Network G



For instance, set the processing times $T_1 = T_2 = T_4 = 1$, $T_3 = 0$, the upper bound on the customer-facing node $d_4 = 1$, and the objective $f_j(SI_j - S_j) = h_j\sigma_j\sqrt{SI_j + T_j - S_j}$. If $h_1\sigma_1 = h_3\sigma_3 = h_4\sigma_4 = 2$ and $h_2\sigma_2 = 1$, then $\{SI_j\}_{j=1}^4 = \{0,1,1,2\}$, $\{S_j\}_{j=1}^4 = \{1,2,1,1\}$ is an optimal solution to (2), with total cost $2\sqrt{0+1-1} + \sqrt{1+1-2} + 2\sqrt{1+0-1} + 2\sqrt{2+1-1} = 2\sqrt{2}$. Indeed, the maximum replenishment time at node 4 is $M_4 = 3$, but the node's outgoing service time cannot exceed $d_4 = 1$, so at least two periods of safety stock must be stored in the network. Placing it all at node 2 is impossible without increasing stock elsewhere (due to node 3's requirements), leading to higher costs. Storing everything at node 4 has cost $2\sqrt{2}$, whereas splitting units across nodes costs at least $\sqrt{1} + 2\sqrt{1} = 3 > 2\sqrt{2}$. Finally, storing everything at 1 or 3, even if feasible, would never be cheaper than $2\sqrt{2}$.

Now, consider LP (4). A standard solver returns an optimal fractional solution with $r^1_{10} = r^1_{00} = r^2_{10} = r^2_{01} = r^3_{11} = r^3_{00} = r^4_{11} = r^4_{10} = 1/2$, $q^{ij}_{11} = q^{ij}_{00} = 1/2$ for all $(i,j) \in E$, and all remaining $r^i_{k\ell} = q^{ij}_{k\ell} = 0$, with objective value $2 + \sqrt{2}/2 < 2\sqrt{2}$. Intuitively, the LP does not include sufficient consistency constraints. In particular, looking at path $1 \to 2 \to 4$, the LP interpolates between an integral solution with one period of inventory at nodes 1 and 4 each and another solution with one period at node 2. From the perspective of path $1 \to 3 \to 4$, the LP interpolates between an integral solution with one period of inventory at node 1 and another solution with one period at node 4. Although the total inventory at nodes 1 and 4 is consistent if the solutions are averaged across paths (half a period each), the individual components that make up these solutions are inconsistent, because they distribute inventory in different ways, depending on which path we look at. Online Appendix E.1 provides more details on this LP solution. It also makes explicit an inequality valid for (2), but violated by the previous solution to (4), showing that (4) admits solutions that do not belong to the convex hull of the feasible set of (2).

This example highlights the need for a principled and scalable framework that systematically identifies which sets of *variables* must be considered jointly for a given network G. Working directly with G is insufficient: it only captures interactions between *nodes* rather than interactions between *variables* in (2). To address this limitation, we introduce a new graph, called the *intersection graph*, which explicitly encodes interactions among variables. Applying the notion of a *tree decomposition* to this intersection graph then provides a formal mechanism for determining which sets of variables must be treated jointly. We give formal definitions of both concepts in the next section and show how they can be used to construct a general LP formulation of (2) for any acyclic graph G.

4.2. Introducing Tree Decompositions, Treewidth, and the Intersection Graph

4.2.1. Tree Decompositions and Treewidth. For any graph G, we can define the notion of a *tree decomposition* and its associated treewidth.

Definition 1 (Bodlaender 1993). Let $G=(V,E)$ be a graph. A *tree decomposition* of G is a pair $\mathcal{T}=(T,\{X_z\}_{z\in T})$ with tree T and a *bag* (or set) X_z for each node $z\in T$ such that:

a. $\cup_{z\in T} X_z = V$

b. If $(i,j)\in E$, then there must be a bag X_z containing both i and j.

c. If a node $i\in V$ appears in two distinct bags X_x and X_y, then i also appears in *every* bag on the unique path between x and y in T.

The *width* of the tree decomposition is $\max_{z\in T}|X_z|-1$, and the treewidth $tw(G)$ is the minimum possible width over all tree decompositions of G.

In other words, we build a tree where each node is a bag containing vertices of G such that properties (a)–(c) hold. As an example, Figure 3 shows a tree decomposition of the diamond network in Figure 2. The tree T is simple, containing two nodes. Each node is a bag of vertices of G. The upper bag contains $X_1=\{1,2,3\}$. The lower bag contains $X_2=\{2,3,4\}$. The corresponding sets are also plotted in their respective colors in Figure 3(a). Property (b) holds, because bag X_1 contains the nodes of edges $(1,3)$ and $(1,2)$, and bag X_2 contains the nodes of edges $(3,4)$ and $(2,4)$. Property (c) holds trivially, because there are only two nodes in the tree. Note that the tree decomposition is not unique: for example, we could also have the bags $X_1=\{1,3,4\}$ and $X_2=\{1,2,4\}$. However, there does not exist any tree decomposition with all bags containing two vertices or fewer. Any such decomposition would have to violate either (b) or (c). Thus, the treewidth of G is equal to the maximum size of the bags, 3, minus 1. Hence, $tw(G)=2$.

We provide a second example of a tree decomposition in Figure 4. Here, one may wish to remove node 2 from the bag {1,2,5}, as edges $(1,2)$, $(2,4)$, and $(2,6)$ are covered by other bags. However, doing so would violate (c) because 2 must appear in the bags $\{2,3,4\}$ and $\{2,5,6\}$, so it also has to appear in all bags on the path between these two. The treewidth of G is also 2.

Figure 3. (Color online) The Diamond Network G (a) and its Tree Decomposition $\mathcal{T}$ (b)

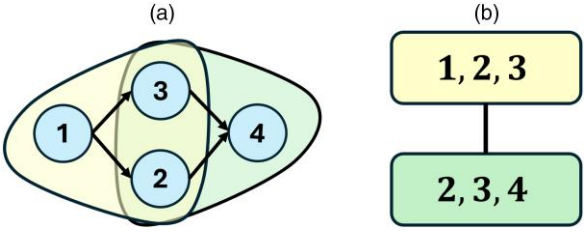
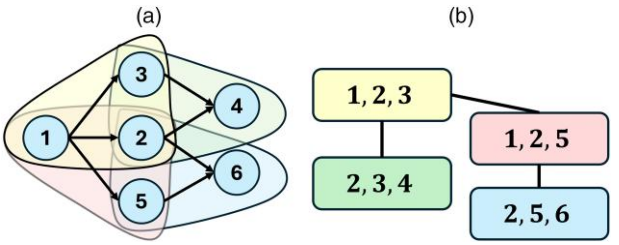


Figure 4. (Color online) A Network G (a) and its Tree Decomposition $\mathcal{T}$ (b)



The treewidth of a graph varies between 1 (for trees) and $n-1$ (for, e.g., complete graphs), where n is the number of vertices in G. Cycles have treewidth 2. In general, it is NP-complete to determine whether a given graph G has treewidth k (Arnborg et al. 1987). However, when k is any fixed constant, the graphs with treewidth k can be recognized and width-k tree decompositions can be constructed in linear time (Bodlaender 1996). In practice, there are several high-quality algorithms to compute tree decompositions and (upper bounds on) treewidths (Dell et al. 2018).

4.2.2. Intersection Graphs. We do not use the tree decomposition of a supply chain network G itself, but the tree decomposition of an auxiliary network G', which we refer to as the *intersection graph* of (2). An intersection graph, at a high level, encodes the interactions between decision variables within an optimization problem. Because (2) is based on the structure of G, in particular through constraints ($2b_{ij}$), there is a relationship between G and G', which we discuss in detail below. First, we formally define the intersection graph for (2). Recall that a function $f:\mathbb{R}^{n_1}\times\cdots\times\mathbb{R}^{n_p}\to\mathbb{R}$ is *separable* if

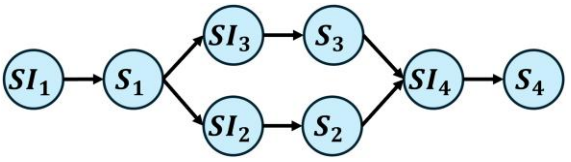
$$f(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_p)=g_1(\boldsymbol{x}_1)+\cdots+g_p(\boldsymbol{x}_p),$$

where $\boldsymbol{x}_i\in\mathbb{R}^{n_i}$ and each $g_i:\mathbb{R}^{n_i}\to\mathbb{R}$. Note that the objective function of (2) is separable.

Definition 2. The *intersection graph* G' for (2) is an undirected graph where each node represents one decision variable S_j or SI_j for $j\in V$. Two nodes share an edge if and only if the corresponding variables in (2) appear together in a constraint or in any term of the separable objective function.

For convenience, we sometimes refer to the intersection graph for (2) over a graph G as the intersection graph of G. In Figure 5, we provide the intersection graph for (2) over the diamond network from Figure 2. Because the diamond network has four nodes, there are eight nodes in the intersection graph corresponding to S_j and SI_j for $j=1,\ldots,4$. There is an edge between SI_j and S_j for $j=1,\ldots,4$ in light of Constraints ($2a_j$) and an edge between S_i and SI_j for all $(i,j)\in E$ in light of

Figure 5. (Color online) Intersection Graph G' for (2) over the Graph G from Figure 2



Constraints ($2b_{ij}$). It is straightforward to construct the intersection graph G' for (2):

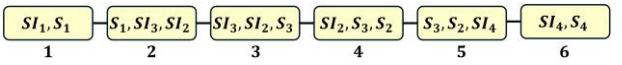
Proposition 1. *Let $G=(V,E)$ be a directed graph. The intersection graph G' for (2) over G is obtained in two steps. First, replace each node $i \in V$ with two nodes SI_i and S_i in G', linked by an edge. Then, for each $(i,j) \in E$, create an (undirected) edge between S_i and SI_j in G'.*

The proof is given in Online Appendix B.2.

4.2.3. Tree Decomposition of the Intersection Graph.

Note that G' is a graph, with decision variables as nodes. As a consequence, the notion of a tree decomposition of G' applies as before, with bags containing decision variables of (2) rather than vertices of G. The tree decomposition of G' from Figure 5 appears in Figure 6. A tree decomposition of the intersection graph can be understood as a blueprint for building our linear programming formulation. Each bag in the decomposition corresponds to a set of variables that must be kept track of simultaneously, because they appear either in the same constraint or in a term of the objective. For instance, the variables (SI_4, S_4) appear in node 6 in the tree decomposition $\mathcal{T}$ in Figure 6, because they are coupled by the constraint $S_4 - SI_4 \le T_4$ and the cost term $f_4(SI_4 - S_4)$. Node 5, in turn, contains (S_2, S_3, SI_4) because the variables are coupled by the constraints $SI_4 - S_2 \ge 0$ and $SI_4 - S_3 \ge 0$. Note that SI_4 is shared between the two bags: the value it takes is determined by both the constraints from node 5 and those from node 6. As we move from node 6 to node 1

Figure 6. (Color online) Tree Decomposition $\mathcal{T}$ of the Intersection Graph G' from Figure 5



in the tree decomposition, the variables shared between adjacent bags ensure global consistency of the solution. For nonserial tree decompositions, such as the one we see later in Figure 7, there is an additional operation—namely, a "merge," where two nodes in the tree are merged into one. There, the common variables across all three nodes perform the same role.

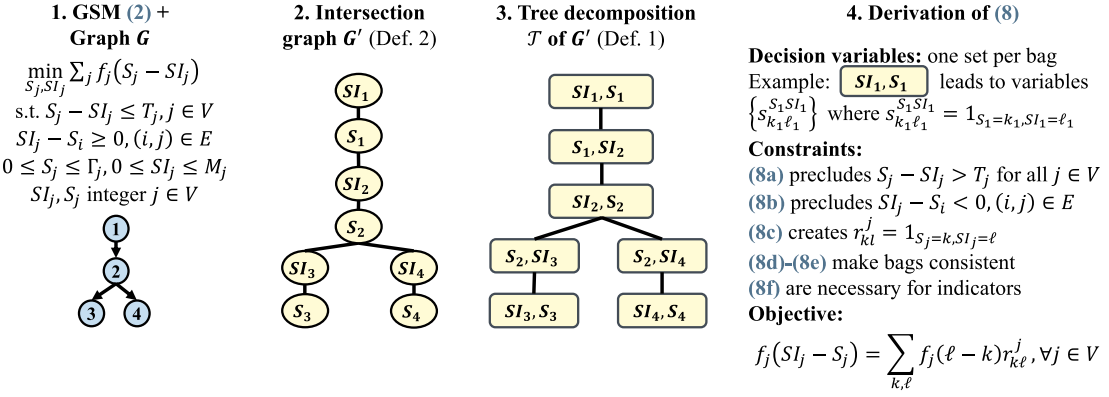
4.3. Reinterpreting Section 3 Using Treewidth and Tree Decompositions

The example in Section 4.1 demonstrates that the LP (4) is not suitable for solving the GSM on graphs that are not trees. Is there a way to generalize (4) so that it can solve the GSM on a broader class of graphs? This is not immediately obvious when using (4), but becomes much clearer if we revisit (4) through the lens of intersection graphs and tree decompositions. Our next proposition thus studies the tree decomposition of the intersection graph G' for (2) when G is a tree. This decomposition plays a crucial role in building the LP that solves the GSM, as illustrated in Figure 7.

Proposition 2. *Let $G=(V,E)$ be a directed tree and let G' be the intersection graph for (2) over G, obtained as in Proposition 1. G' is also a tree. Consider the tree decomposition $\mathcal{T}=(T,\{X_z\}_{z\in T})$ built as follows. Root G' at a leaf. For each edge of G', introduce a node z in T and let X_z contain the two variables joined by that edge. Connect two nodes of T whenever the corresponding edges in G' share a vertex and one is the parent edge of the other. Then, $\mathcal{T}$ is a tree decomposition of G' with the following properties*: (i) *$\mathcal{T}$ has width one and* (ii) *every bag contains either a pair (S_i, SI_j) for $(i,j) \in E$ or a pair (S_j, SI_j) for $j \in V$.*

It is easy to check that $\mathcal{T}$ is a tree decomposition of G' and that its width is one. Point (ii) follows because

Figure 7. (Color online) How to Obtain LP (8) from the Tree Decomposition $\mathcal{T}$



Notes. G' is the intersection graph of (2) on G. This approach is not specific to trees and applies to *any* acyclic graph (see Section 4.4).

there are only edges in G' between pairs (S_j, SI_j) for $j \in V$ and pairs (S_i, SI_j) for $(i,j) \in E$. In Figure 7, we provide an illustrative example of the construction of G' and $\mathcal{T}$ for a tree graph G.

We now reinterpret LP (4) using the tree decomposition $\mathcal{T}$ of G' and its structural properties. We can see that the variables $\{r^j_{k\ell}\}_{k,\ell}$ for $j \in V$ and $\{q^{ij}_{k\ell}\}_{k,\ell}$ for $(i,j) \in E$ map to the bags in the tree decomposition $\mathcal{T}$. In the example in Figure 7, $q^{23}_{k\ell}$ is associated with the bag (S_2, SI_3) in $\mathcal{T}$, whereas $r^2_{k\ell}$ is associated with the bag (S_2, SI_2) in $\mathcal{T}$. Constraints $(4c_j)$ and $(4d_i)$ ensure that the value taken on by the variable in the overlap between a parent bag and a child bag is consistent: For instance, the LP associated with the example contains the equality constraint $\sum_\ell q^{23}_{k\ell} = \sum_\ell r^2_{k\ell}$, $\forall k$. Because $q^{23}_{k\ell}$ corresponds to (S_2, SI_3) and $r^2_{k\ell}$ corresponds to (S_2, SI_2) in $\mathcal{T}$, this constraint simply ensures that S_2, which is the variable in the overlap between the two bags, has the same values in both bags.

We can thus rewrite (4) using tree decomposition notation. For a node $z \in T$, let i_z, respectively j_z be the indices of the variables S_i, respectively SI_j, that are present in the corresponding bag X_z. In Figure 7, the top node of T has $i_z = 1$ and $j_z = 1$. Given Proposition 2, we can then partition the nodes in T into two:

$$T_1 = \{z \in T \mid (S_j, SI_j) \in X_z \text{ for } j \in V\} \text{ and}$$
$$T_2 = \{z \in T \mid (S_i, SI_j) \in X_z \text{ for } (i,j) \in E\}. \tag{6}$$

We now replace the two sets of variables $\{r^j_{k\ell}\}_{k,\ell}$ for $j \in V$ and $\{q^{ij}_{k\ell}\}_{k,\ell}$ for $(i,j) \in E$ by one set only, $\{s^z_{k\ell}\}_{k,\ell}$, indexed by the nodes $z \in T$. Note that

$$s^z_{k\ell} = r^{j_z}_{k\ell},\ \forall k,\ell \text{ when } z \in T_1 \text{ and } s^z_{k\ell} = q^{i_z j_z}_{k\ell},\ \forall k,\ell \text{ when } z \in T_2. \tag{7}$$

Using the shorthand $p(z)$ to mean the parent of node $z \in T$, LP (4) is then equivalent to:

$$\min \quad \sum_{j \in V} \sum_{\ell=0}^{M_j} \sum_{k=0}^{\Gamma_j} f_j(\ell - k) r^j_{k\ell}, \tag{8}$$

$$\text{s.t.} \quad r^j_{k\ell} \le \left\lfloor \frac{T_j}{k-\ell} \right\rfloor, \quad \forall k = 0,\dots,\Gamma_j,\ \ell = 0,\dots,M_j,\ k > \ell,\ j \in V, \tag{8a$_j$}$$

$$s^z_{k\ell} = 0, \quad \forall (k,\ell) \in \mathcal{S}_{i_z j_z},\ \forall z \in T_2, \tag{8b^z}$$

$$r^{j_z}_{k\ell} = s^z_{k\ell}, \quad \forall k = 0,\dots,\Gamma_{j_z}, \ell = 0,\dots,M_{j_z}, \quad \forall z \in T_1, \tag{8c^z}$$

$$\sum_{k=0}^{\Gamma_{j_z}} s^z_{k\ell} = \sum_{k=0}^{\Gamma_{j_{p(z)}}} s^{p(z)}_{k\ell}, \ \forall \ell = 0,\dots,M_{j_z}, \quad \forall z \text{ s.t. } i_{p(z)} \neq i_z, \tag{8d^z}$$

$$\sum_{\ell=0}^{M_{j_z}} s^z_{k\ell} = \sum_{\ell=0}^{M_{j_{p(z)}}} s^{p(z)}_{k\ell}, \ \forall k = 0,\dots,\Gamma_{j_z}, \quad \forall z \text{ s.t. } j_{p(z)} \neq j_z, \tag{8e^z}$$

$$\{s^z_{k\ell}\}_{k,\ell} \in \Delta, \quad \forall z \in T. \tag{8f^z}$$

It is straightforward to see that LP (8) is equivalent to (4), using Proposition 2 and (7). This is the LP formulation we generalize to the case of bounded treewidth.

4.4. An LP Formulation of the GSM for Acyclic Networks with Bounded Treewidth

The process we followed to obtain LP (8), illustrated in Figure 7, is not specific to trees and can be generalized to the case where G is any arbitrary graph of bounded treewidth. As seen in Figure 7, the tree decomposition of the intersection graph of (2) plays a crucial role in deriving the LP. This motivates the next proposition, which is an analog of Proposition 2 for arbitrary G.

Proposition 3. *Let G be a directed graph of treewidth ω and let G' be the corresponding intersection graph obtained as described in Proposition 1. Suppose that G has a tree decomposition $\mathcal{T}^* = (T^*, \{X^*_z\}_{z \in T})$. Build a new tree decomposition $\mathcal{T} = (T, \{X_z\}_{z \in T})$ in the following way:*

1. *Set $T = T^*$.*
2. *Set $X_z = \cup_{i \in X^*_z} \{S_i, SI_i\}$ for all $z \in T$.*

Then, $\mathcal{T}$ is a tree decomposition of G' with width at most $2\omega + 1$.

The proof is in Online Appendix B.2. We use the diamond network G from Figure 2, its intersection graph G' from Figure 5, and the tree decomposition $\mathcal{T} = (T, \{X_z\}_{z \in T})$ of G' from Figure 6 as our running example. We number the nodes in T by 1 through 6 from left to right, as indicated in the figure—for example, $X_4 = \{SI_2, S_2, S_3\}$. We first introduce some notation relating to $\mathcal{T}$ and then state our main result. Parallels between this notation and that in Section 4.3 should be easy to see.

Recall that the bags of $\mathcal{T}$ contain decision variables of (2). Thus, for each node $z \in T$, we keep track of the indices of the variables S_i (resp., SI_i) present in the bag X_z—that is, we define

$$I_z = \{i \in V \mid S_i \in X_z\} \text{ and } J_z = \{j \in V \mid SI_j \in X_z\}.$$

In the example in Figure 6, we have $J_4 = \{2\}$ and $I_4 = \{2,3\}$. It will be useful to partition the nodes of T into the nodes that contain the same-index pairs (S_j, SI_j) and those that do not. We thus define

$$T_1 = \{z \in T \mid I_z \cap J_z \neq \emptyset\} \text{ and } T_2 = \{z \in T \mid I_z \cap J_z = \emptyset\}. \tag{9}$$

In the running example, $T_1 = \{1,3,4,6\}$ and $T_2 = \{2,5\}$.

We are now ready to introduce the decision variables of our problem. These are bag-specific, as seen in Figure 7, and keep track of all the values assigned to the variables in the bag. We let

$$\boldsymbol{K}_z = (k_{i_1}, \dots, k_{i_{|I_z|}}) \text{ for } i_1, \dots, i_{|I_z|} \in I_z \text{ and}$$
$$\boldsymbol{L}_z = (\ell_{j_1}, \dots, \ell_{j_{|J_z|}}) \text{ for } j_1, \dots, j_{|J_z|} \in J_z, \text{ with}$$
$$\boldsymbol{K}_z \in \mathcal{D}_{\boldsymbol{K}_z} := \{0, \dots, \Gamma_{i_1}\} \times \cdots \times \{0, \dots, \Gamma_{i_{|I_z|}}\} \text{ and}$$
$$\boldsymbol{L}_z \in \mathcal{D}_{\boldsymbol{L}_z} := \{0, \dots, M_{j_1}\} \times \cdots \times \{0, \dots, M_{j_{|J_z|}}\}.$$

As $\Gamma_{i_1},\dots,\Gamma_{i_{|I_z|}},M_{j_1},\dots,M_{j_{|J_z|}} \le M$, and $|I_z|+|J_z| \le 2\omega+2$, $(\boldsymbol{K}_z,\boldsymbol{L}_z)$ take on at most $O(M^{2\omega+2})$ values. We define, for $z \in T$, the variables $\{s^z_{\boldsymbol{K}_z\boldsymbol{L}_z}\}_{(\boldsymbol{K}_z,\boldsymbol{L}_z)\in\mathcal{D}_{\boldsymbol{K}_z}\times\mathcal{D}_{\boldsymbol{L}_z}}$, of which there are at most $O(M^{2\omega+2})$. Because G' contains $2n$ nodes, it follows from Kloks (1994) that $|T| \le 8n$. Thus, there are a total of $O(nM^{2\omega+2})$ variables $\{s^z_{\boldsymbol{K}_z\boldsymbol{L}_z}\}_{(\boldsymbol{K}_z,\boldsymbol{L}_z),z\in T}$. Consider, for example, the diamond network: for node 4 in T, with $X_4=\{SI_2,S_2,S_3\}$ and, thus, $I_4=\{2,3\}$ and $J_4=\{2\}$, we introduce the variables $\{s^4_{k_2k_3\ell_2}\}$ for $k_2=0,\dots,\Gamma_2$, $k_3=0,\dots,\Gamma_3$, and $\ell_2=0,\dots,M_2$. The variable s^4_{221}, for example, should then be interpreted as an indicator that $S_2=2$, $S_3=2$, and $SI_2=1$.

As in (8), we also use variables $\{r^{j_z}_{k\ell}\}$ for nodes $z \in T_1$—that is, bags in which same-index pairs (S_{j_z},SI_{j_z}) appear. This is important for the objective function. There are a total of $O(nM^2)$ such variables, bringing the total number of variables to $O(nM^{2\omega+2})+O(nM^2)=O(nM^{2\omega+2})$.

To generalize Constraint ($8b^z$), we require an analog to the set $\mathcal{S}_{ij}$. For each node $z \in T$ and for every directed edge $(i_z,j_z) \in (I_z\times J_z)\cap E$, define

$$\mathcal{S}^z_{i_zj_z}=\{(\boldsymbol{K}_z,\boldsymbol{L}_z)\mid \boldsymbol{K}_z\in\mathcal{D}_{\boldsymbol{K}_z},\ \boldsymbol{L}_z\in\mathcal{D}_{\boldsymbol{L}_z},\ k_{i_z}>\ell_{j_z}\}. \tag{10}$$

In other words, for each node $z \in T$ and any $(i_z,j_z) \in (I_z\times J_z)\cap E$, the set $\mathcal{S}^z_{i_zj_z}$ keeps track of all the values of $(S_{i_1},\dots,S_{i_r},SI_{j_1},\dots,SI_{j_p})$, where $S_{i_z}>SI_{j_z}$. In Figure 6, for instance, $\mathcal{S}^2_{12}$ is needed because the edge $(1,2)$ appears in E. The set would include, for example, the vector $(k_1,\ell_2,\ell_3)=(1,0,1)$.

To generalize constraints ($8d^z$) and ($8e^z$) in a succinct way, we require a new variable set. As before, for $z \in T$ (excluding the root), let $p(z)$ be its unique parent. Note that $X_z\cap X_{p(z)}$ contains the overlapping variables between those in node z and those in node $p(z)$. In our running example, we have $X_3\cap X_{p(3)}=X_3\cap X_2=\{SI_2,SI_3\}$ and $X_4\cap X_{p(4)}=X_4\cap X_3=\{SI_2,S_3\}$. Define

$$\hat{I}_z=I_z\cap I_{p(z)} \quad\text{and}\quad \hat{J}_z=J_z\cap J_{p(z)}, \tag{11}$$

so $\hat{I}_z$ (resp., $\hat{J}_z$) consists of the S_i (resp., SI_j) variables that appear in both X_z and $X_{p(z)}$. In our running example, $\hat{I}_4=\{3\}$, whereas $\hat{J}_3=\{2,3\}$. Note that if $\hat{I}_z$ or $\hat{J}_z$ are empty, then any indexing using these sets or summations over them can be dropped. We now state our main result:

Theorem 2. *When $G=(V,E)$ is a graph of treewidth ω, the Guaranteed Service Model given in (2) can be solved via a linear program in $O(nM^{2\omega+2})$ variables and $O(nM^{2\omega+2})$ constraints*:

$$\min \sum_{j\in V}\sum_{\ell=0}^{M_j}\sum_{k=0}^{\Gamma_j} f_j(\ell-k)r^j_{k\ell}, \tag{12}$$

$$\text{s.t. } r^j_{k\ell}\le\left\lfloor\frac{T_j}{k-\ell}\right\rfloor,\ \forall k=0,\dots,\Gamma_j,\ \ell=0,\dots,M_j,\ k>\ell,\ j\in V, \tag{12a_j}$$

$$s^z_{\boldsymbol{K}_z\boldsymbol{L}_z}=0,\ \forall(\boldsymbol{K}_z,\boldsymbol{L}_z)\in\mathcal{S}^z_{i_zj_z},\ \forall(i_z,j_z)\in(I_z\times J_z)\cap E,\ z\in T, \tag{12$b^z_{i_zj_z}$}$$

$$r^{j_z}_{k_{j_z}\ell_{j_z}}=\sum_{\boldsymbol{K}_z\setminus k_{j_z}}\sum_{\boldsymbol{L}_z\setminus\ell_{j_z}} s^z_{\boldsymbol{K}_z\boldsymbol{L}_z},\ \forall k_{j_z},\ell_{j_z},\ \forall j_z\in I_z\cap J_z,\ \forall z\in T_1, \tag{12$c^z_{j_z}$}$$

$$\sum_{\boldsymbol{K}_z\setminus\{k_i\}_{i\in\hat{I}_z}}\sum_{\boldsymbol{L}_z\setminus\{\ell_j\}_{j\in\hat{J}_z}} s^z_{\boldsymbol{K}_z\boldsymbol{L}_z}=\sum_{\boldsymbol{K}_{p(z)}\setminus\{k_i\}_{i\in\hat{I}_z}}\sum_{\boldsymbol{L}_{p(z)}\setminus\{\ell_j\}_{j\in\hat{J}_z}} s^{p(z)}_{\boldsymbol{K}_{p(z)}\boldsymbol{L}_{p(z)}}\quad \forall\{k_i\}_{i\in\hat{I}_z},\ \{\ell_j\}_{j\in\hat{J}_z},\ z\in T, \tag{12d^z}$$

$$\{s^z_{\boldsymbol{K}_z\boldsymbol{L}_z}\}_{(\boldsymbol{K}_z,\boldsymbol{L}_z)\in\mathcal{D}_{\boldsymbol{K}_z}\times\mathcal{D}_{\boldsymbol{L}_z}}\in\Delta,\ \forall z\in T, \tag{12e^z}$$

where the decision variables are

$$\{r^j_{k\ell}\}_{k=0,\dots,\Gamma_j,\ \ell=0,\dots,M_j} \text{ for } j\in V \text{ and } \{s^z_{\boldsymbol{K}_z\boldsymbol{L}_z}\}_{\boldsymbol{K}_z\in\mathcal{D}_{\boldsymbol{K}_z},\boldsymbol{L}_z\in\mathcal{D}_{\boldsymbol{L}_z}} \text{ for } z\in T.$$

For each node $j\in V$, let SI^*_j and S^*_j denote the optimal inbound and outbound service times in (2), and let $r^{j*}_{k\ell}$ denote the optimal value of $r^j_{k\ell}$ in (12). Then,

$$SI^*_j=\sum_{\ell=0}^{M_j}\left(\ell\cdot\sum_{k=0}^{\Gamma_j} r^{j*}_{k\ell}\right) \text{ and } S^*_j=\sum_{k=0}^{\Gamma_j}\left(k\cdot\sum_{\ell=0}^{M_j} r^{j*}_{k\ell}\right), \text{ for } j\in V. \tag{13}$$

Remark 2. Given its scaling, this approach is particularly appropriate when G is large but tree-like (i.e., ω is small) and M, the maximum replenishment time, is small.

Remark 3. The LP in (12) mirrors (8): we can match ($8a_j$) to ($12a_j$), ($8b^z$) to ($12b^z_{i_zj_z}$), ($8c^z$) to ($12c^z_{j_z}$), ($8d^z$)–($8e^z$) to ($12d^z$), and, finally, ($8f^z$) to ($12e^z$). Each constraint in (12) plays the same role as its counterpart in (8); see Figure 7 for a high-level explanation of what this is. Moreover, the size of (12) can be slightly reduced, as explained in Remark EC.1 in the Online Appendix.

We refer the reader to Online Appendix D for the complete proof of Theorem 2. The proof follows the same outline as the one depicted in Figure 1. Figure EC.2 repeats it, adding the dependency structure among the propositions and lemmas used to establish the result. We wrap up by writing out the LP (12) for the diamond network G in Figure 2. As seen in Figure 7, this requires the intersection graph for (2) over G in Figure 5, as well as its tree decomposition in Figure 6. We then introduce one set of variables per bag: for node 2, this is $s^2_{k_1\ell_2\ell_3}$; for node 3, this is $s^3_{k_3\ell_2\ell_3}$; for node 4, this is $s^4_{k_2k_3\ell_2}$; and for node 5, this is $s^5_{k_2k_3\ell_4}$. For nodes 1 and 6, which contain same-index pairs (SI_i,S_i), these variables are redundant with $r^1_{k_1\ell_1}$ and $r^4_{k_4\ell_4}$, so they are excluded. We now write the LP necessary to solve the GSM over

the diamond:

$$\min \sum_{j=1}^{4}\sum_{\ell=0}^{M_j}\sum_{k=0}^{\Gamma_j} f_j(\ell-k) r^j_{k\ell}$$

$$\text{s.t. } r^j_{k\ell} \le \left\lfloor \frac{T_j}{k-\ell} \right\rfloor,\ \forall k=0,\dots,\Gamma_j, \ell=0,\dots,M_j, k>\ell, j=1,\dots,4,$$

$$s^2_{k_1\ell_2\ell_3}=0,\ \forall k_1>\ell_2,\ \forall \ell_3, s^2_{k_1\ell_2\ell_3}=0,\ \forall k_1>\ell_3,\ \forall \ell_2,$$

$$s^4_{k_2k_3\ell_4}=0,\ \forall k_2>\ell_4,\ \forall k_3, s^5_{k_2k_3\ell_4}=0,\ \forall k_3>\ell_4,\ \forall k_2,$$

$$r^2_{k_2\ell_2}=\sum_{k_3} s^4_{k_2k_3\ell_2},\ \forall k_2,\ell_2, r^3_{k_3\ell_3}=\sum_{\ell_2} s^3_{k_3\ell_2\ell_3},\ \forall k_3,\ell_3,$$

$$\sum_{\ell_1} r^1_{k_1\ell_1}=\sum_{\ell_2}\sum_{\ell_3} s^2_{k_1\ell_2\ell_3},\ \forall k_1, \sum_{k_2}\sum_{k_3} s^5_{k_2k_3\ell_4}=\sum_{k_4} r^4_{k_4\ell_4},\ \forall \ell_4,$$

$$\sum_{k_1} s^2_{k_1\ell_2\ell_3}=\sum_{k_3} s^3_{k_3\ell_2\ell_3},\ \forall \ell_2,\ell_3, \sum_{\ell_3} s^3_{k_3\ell_2\ell_3}=\sum_{k_2} s^4_{k_2k_3\ell_2},\ \forall k_3,\ell_2,$$

$$\sum_{\ell_2} s^4_{k_2k_3\ell_2}=\sum_{\ell_4} s^5_{k_2k_3\ell_4},\ \forall k_2,k_3,$$

$$\{r^1_{k_1\ell_1}\},\{s^2_{k_1\ell_2\ell_3}\},\{s^3_{k_3\ell_2\ell_3}\},\{s^4_{k_2k_3\ell_2}\},\{s^5_{k_2k_3\ell_4}\},\{r^4_{k_4\ell_4}\}\in\Delta.$$

Online Appendix E.2 provides an additional example.

5. Theoretical Advantages of Our Approach

We discuss how our solution approach contributes to the optimization literature on treewidth (Section 5.1) and improves upon the state-of-the-art for optimizing the GSM (Section 5.2).

5.1. Novelty and Broader Applicability of the Treewidth-Informed LP

Theorem 2 presents a linear programming reformulation of the GSM of size $O(nM^{2\omega+2})$ on a network of treewidth ω. Thus, our algorithm is *fixed-parameter tractable* (FPT) (Downey and Fellows 2012) in M and ω. We first discuss how our reformulation is new to the literature and then explain how it directly applies to other problems in supply chain management.

5.1.1. Our Treewidth-Informed LP Reformulation as a New Contribution to the Literature on Algorithms FPT in Treewidth. The current optimization literature focuses on FPT algorithms in the treewidth for two sets of problems: Integer Linear Programs (ILPs), which are optimization problems of the form $\min_{x\in\mathbb{Z}^n, Ax\le b} c^T x$, and Binary Linear Programs (BiLPs), which are optimization problems of the form $\min_{x\in\{0,1\}^n, Ax\le b} c^T x$ for $c\in\mathbb{R}^n$, $A\in\mathbb{Z}^{m\times n}$ and $b\in\mathbb{Z}^m$.

Jansen and Kratsch (2015) show that ILPs are FPT in the treewidth of the intersection graph, which is, at most, $2\omega+1$ in our case, and the infinity-norm of the difference between the variables' upper and lower bounds, which is M in our case. However, this result does not apply to our case because (2) is *not* an ILP due to its nonlinear objective function. Furthermore, the result in Jansen and Kratsch (2015) relies on kernelization, a dynamic-programming-based preprocessing routine that yields smaller ILPs, which are then optimized. They do not propose a linear programming reformulation.

For BiLPs, Laurent (2009), Muñoz (2017), and Bienstock and Muñoz (2018) propose LP reformulations that are FPT in the treewidth of the intersection graph of the BiLP. Although (2) is not a BiLP, because its variables are natural numbers, it is somewhat straightforward to find a BiLP reformulation. There are naive approaches to do so (e.g., replacing S_i by $\sum_{k=1}^{M} k\cdot\mathbf{1}_{\{S_i=k\}}$, resp., SI_i by $\sum_{k=1}^{M} k\cdot\mathbf{1}_{\{SI_i=k\}}$) and more complex approaches, such as the binarizations in the proofs of Theorems 1 and 2. Applying the results of prior literature to any of these binarizations leads to LPs that are much larger than those we provide, because the treewidth of the intersection graph of these problems is larger than 1 (in the tree case) or $2\omega+1$ (in the general case). Thus, linking the treewidth of the supply chain network to the complexity of the resulting LP requires a new reformulation procedure.

To illustrate, consider the tree LP with overlap variables (EC.2) in the Online Appendix and add binary constraints to the decision variables. From Propositions EC.1 and EC.2 in the Online Appendix, this is a binarization of (2). From equation (EC.2f$_j$), for every fixed $j\in V$ and $k\in\{0,\dots,\Gamma_j\}$, there is an edge between γ^j_k and $r^j_{k\ell}$ for all $\ell\in\{0,\dots,M_j\}$ in the intersection graph. This implies that there is a clique of size M_j+2 (equal to $M+2$ for some $j\in V$), which, in turn, implies that the treewidth is lower-bounded by $M+1$.

If we were to use the results in the literature, such as those in Bienstock and Muñoz (2018), we would then have to reformulate (EC.2) as another LP, this time of size $O(nM^{M+1})$, which is orders of magnitude larger than the LP we use, of size $O(nM^2)$. Thus, the fact that (4) and (12) are the "right" LPs—exact reformulations of (2) with the appropriate complexity guarantees—is not straightforward. Our reformulation may also be of broader interest, as we discuss next.

5.1.2. Applying Our Approach to Other Problems in the Literature. Our treewidth-informed LP reformulation applies to any optimization problem whose structure shares two crucial characteristics with that of the GSM in (2): a separable objective function with one term per node; and precedence constraints, where decision variables interact only if they are associated with the same node or with nodes connected by an edge. These characteristics are present in other Operations Management problems widely used in practice, most notably the *Discrete Time-Cost Tradeoff Problem* (DTCT).

The DTCT is a classic, strongly NP-hard scheduling problem, defined over a directed acyclic graph (De et al. 1997). Nodes represent activities and edges their dependencies. For each activity j, a decision maker

chooses from a discrete set of possible durations, where shorter ones incur higher costs. The goal is to find a schedule that minimizes the total cost while fulfilling external deadlines.

The structural equivalence between the GSM and the DTCT is straightforward: Let an activity's start time be analogous to a stage's inbound service time (SI_j) and its completion time be analogous to its outbound service time (S_j). Then, the duration of an activity is bounded similarly to the nonnegativity constraint of inventory ($S_j - SI_j \leq T_j$), and the duration $S_j - SI_j$ also drives the cost of the activity. Moreover, the GSM's precedence constraints ($SI_j \geq S_i$ for any edge (i,j)) map directly to the temporal logic of a project network, where j cannot start before i's completion. Finally, leaf nodes $j \in L$ may have externally imposed deadlines $S_j \leq d_j$. This connection has been recognized before. For instance, Li and Jiang (2012) explicitly model the GSM as a project scheduling problem to develop constraint programming and heuristic solution methods.

The DTCT has been extensively studied, and, similar to the GSM, optimization approaches typically consider specific network structures and consist of bespoke DPs, heuristics, approximation algorithms, and nonlinear mixed integer programs. For a review of early research, see De et al. (1995) and Skutella (1998). Our treewidth-based LP framework, therefore, represents a significant contribution to this domain as well. It provides the first exact algorithm for the general case of the DTCT that is fixed-parameter tractable in the treewidth of the project network. This allows for the exact and efficient optimization of a broad class of scheduling problems that are typically considered intractable, provided their underlying network structure is sufficiently tree-like.

5.2. Improvements on Previous Solution Approaches to the GSM

Table 1 presents a summary of how our approach improves upon existing GSM exact solution approaches by (i) applying to any directed acyclic graph, (ii) allowing for any objective function, and (iii) having computational complexity that does not scale exponentially in the number of nodes n. We discuss each of these improvements below.

5.2.1. Applies to Any Directed Acyclic Graph. This contrasts with a large number of prior GSM papers, which develop exact solution approaches for networks with specific structures. Existing results are subsumed or extended by Theorems 1 and 2.

More specifically, Theorem 1 subsumes existing results for optimizing the GSM on trees by introducing a linear programming reformulation of the GSM of size $O(nM^2)$, where M is as defined in Section 2.3. Solving the GSM on trees dates back to the first theoretical breakthrough in Simpson (1958), who shows that for serial networks, an "all-or-nothing" strategy is optimal—stages either hold maximal inventory or none at all. Inderfurth (1991) extends this approach to distribution networks, and Graves and Willems (2000) further generalize it, with a pseudo-polynomial dynamic programming algorithm for any tree network, running in time $O(nM^2)$.

Lesnaia (2004) leverages extreme point properties to strengthen the result of Graves and Willems (2000) to a polynomial-time DP algorithm, running in time $O(n^3)$ for tree networks. The crux of their proof is to show that the variables S_i and SI_i can be chosen among $\min\{n^2, M^2\} \leq n^2$ possible values (rather than M^2). In Lesnaia (2004), algorithm 4 finds such values efficiently and then runs the dynamic program developed by Graves and Willems (2000). We can recover the polynomial-time complexity of Lesnaia (2004) through Theorem 1, by leveraging their algorithm 4 to restrict the values taken on by k and ℓ in (4), before solving the LP.

Finally, Theorem 2 generalizes the approach by Humair and Willems (2006), who use a DP to solve the GSM on "clusters of commonality"—overarching tree structures with nodes replaced by bipartite subgraphs.

Table 1. A Summary of How Our Approach Compares to Other Exact Solution Approaches to the GSM

Paper	Description	Limitation	Relevant theorem
Simpson (1958)	Serial graphs + DP	Specific networks	Subcase Theorem 1
Inderfurth (1991)	Distrib. graphs + DP	Specific networks	Subcase Theorem 1
Graves and Willems (2000)	Trees + DP (pseudo-polynomial)	Specific networks	Theorem 1 (recover complexity)
Lesnaia (2004)	Trees + DP (polynomial)	Specific networks	Modif. of Theorem 1 (recover complexity)
Humair and Willems (2006)	CoC + DP	Specific networks	Subcase Theorem 2
Lesnaia (2004)	B&B + DP	Needs concave obj.	Theorem 2
Magnanti et al. (2006)	Piecewise lin. approx.	Needs concave obj.	Theorem 2
Humair and Willems (2011)	B&B + DP	Exponential in n; DP rather than LP; expensive to run	Theorem 2

Note. Approx., approximation; B&B, branch-and-bound; CoC, network with clusters of commonality; distrib., distribution; DP, dynamic programming; lin., linear; LP, linear programming; modif., modification; obj., objective.

Theorem 2 allows for embedding *any graph of bounded treewidth* into a tree, through a tree decomposition of G. The algorithm by Humair and Willems (2006) runs in time $O(n_T M^c)$, where c is the maximum size of the smallest partition across all bipartite subgraphs and n_T is the number of nodes in the tree. Interestingly, the construction of these networks immediately provides a (minimal) tree decomposition of the graph, and its treewidth is of order c.

5.2.2. Allows for General Objective Functions. Our approach places no structural restriction on a node's cost function, which can be any function of the node's service times, provided that the incoming and outgoing service times can be bounded. In other words, our approach allows for general objective functions, which need not be concave, increasing, or even closed-form functions of the service times. This enables modeling stochastic service times, as explained in Humair et al. (2013), and nonnested review periods (Humair and Willems 2011).

It also contrasts with prior work in the literature, leveraging structural properties of the objective to build solution approaches. For example, Lesnaia (2004) develops a branch-and-bound scheme that makes use of the extreme point properties of the optimal solution. Magnanti et al. (2006) iteratively refine a piecewise-linear approximation of the objective by adding redundant constraints. Both of these approaches crucially rely on the concavity of the objective function.

5.2.3. Does Not Scale Exponentially in *n*. To the best of our knowledge, the only other paper in the literature developing an exact solution approach for directed acyclic graphs and general objective functions is Humair and Willems (2011). We thus provide an in-depth comparison with their results.

The approach proposed in Humair and Willems (2011) involves a branch-and-bound algorithm. At the root node, it optimizes the GSM on a tree using the dynamic program in Graves and Willems (2000). It then branches on edges, based on the Constraints ($2b_{ij}$) violated by the current solution, and resolves the previous dynamic program with improved bounds on the variables. Thus, in the worst case, the approach can be exponential in the number of nodes n.

In contrast, our approach scales polynomially in n, pseudo-polynomially wrt the bit size of the maximum replenishment time M, and, for fixed M, exponentially in the treewidth, which can be much smaller than n. Online Appendix E.3 constructs an example of a network where Humair and Willems (2011) need to solve order 2^n DPs to obtain an optimal solution, whereas our LP remains polynomial in n, showcasing the dramatic differences in terms of computational complexity.

6. Practical Advantages of Our Approach

We describe how the results from Sections 3 and 4 can be used in practice to solve the GSM (Section 6.1) and illustrate their computational performance on real-world examples (Section 6.2). We then explain how our approach further supports managerial decision making (Section 6.3).

6.1. Solving the GSM in Practice

The LP in (12) can be expensive to solve, especially when M and ω are large. The formulation is nevertheless useful beyond its role as an exact method: it yields a hierarchy of increasingly tight LP relaxations, whose optimal values are lower bounds on the GSM. This motivates the following GSM optimization strategy:

1. Solve (4), the cheapest relaxation in the hierarchy. If its optimal solution is binary, the lower bound is tight, and the optimal service times to the GSM follow from (5). In practice, we find that for large instances, adding a very small linear tie-breaking perturbation term to the objective can help recover a binary solution when (4) is tight. If, however, the solution to (4) is fractional, we proceed with the options below.

2(a). Move up the aforementioned hierarchy of LP relaxations of (2), which are increasingly tight, but correspondingly more expensive. This hierarchy culminates in the exact formulation (12) and is thus guaranteed to recover the optimal solution to (2). The procedure can stop early if one of these relaxations has a binary optimal solution or if its lower bound matches the value of a feasible GSM solution.

2(b). Run an enhanced version of the algorithm in Humair and Willems (2011) that leverages our LP lower bounds.

We describe each step in more detail below. Online Appendix F.2 also presents a principled approach for obtaining upper bounds on the optimal value of (2). These upper bounds can be paired with the LP lower bounds to certify optimality or bound the remaining gap. We highlight that in all our tests on real-world data, Step 1 alone suffices to recover an optimal GSM solution (see Section 6.2). The same is true for 99.2% of the synthetic examples used in Online Appendix F.2.

6.1.1. Step 1—Solve (4), the Cheapest Relaxation. By Theorem 2, the optimal value of (12) equals that of the GSM (2). It therefore suffices to show that (4) provides a lower bound on (12), which is the content of Proposition 4.

Proposition 4. *The optimal value of* (4) *is a lower bound on the optimal value of* (12). *This lower bound is tight if the optimal solution to* (4) *is binary.*

A formal proof is in Online Appendix F.1. We provide a high-level outline here. Recall that the decision

variables in (12) encode joint assignments of *all* variables $\{S_i, SI_j\}_{i\in I_z, j\in J_z}$ appearing together in the bags z of the tree decomposition $\mathcal{T} = (T, \{X_z\}_{z\in T})$ of G' (see, e.g., Figure 7). In contrast, the decision variables in (4) keep track of a much smaller set of variables—namely, the pairs (S_i, SI_i) for $i \in V$ and (S_i, SI_j) for $(i,j) \in E$. Each pair is guaranteed to appear in at least one bag of the tree, by definition of $\mathcal{T}$. Given a solution to (12), one can then obtain a solution to (4) by identifying which bag z a given pair belongs to and "summing out" all values taken on by the variables $\{S_i, SI_j\}_{i\in I_z, j\in J_z}$ that do not feature in this pair. In other words, the decision variables of (4) can be viewed as low-dimensional marginals of the decision variables in (12), where all but two variables have been summed out. Because the constraints in (12) are consistency constraints on the joint assignments of $\{S_i, SI_j\}_{i\in I_z, j\in J_z}$ across overlapping bags, they automatically imply the corresponding consistency conditions in (4) for these marginals. Because the objective function of (12) only depends on the decision variables through the marginal variables corresponding to the subsets $(S_i, SI_i)_{i\in V}$, the optimal value of (4) is a lower bound on the optimal value of (12).

Proposition 4 shows that one may replace the exact LP (12) with the significantly smaller LP (4) when a lower bound suffices. It also provides an easy-to-check condition under which the bound is exact. As mentioned above, this lower bound is often remarkably tight in practice: in all instances in Willems (2008), (4) provides a tight lower bound on the true optimal value (see Section 6.2). We also found that for some large instances in Willems (2008), adding a small linear tie-breaking perturbation to the objective helps recover the optimal solution.

6.1.2. Step 2(a): Move up the Hierarchy of LP Relaxations. The LP (4) can be viewed as the first step in a hierarchy of lower bounds obtained by retaining increasingly higher-dimensional marginals of the joint assignment variables in (12). Tracking larger subsets of variables—as opposed to the pairs considered in (4)—yields progressively tighter lower bounds at the cost of larger linear programs. The original formulation (12) is recovered when all variables in each bag are tracked jointly. This induces a natural hierarchy that interpolates between the inexpensive pairwise relaxation (4) and the exact formulation in (12). We illustrate this approach on a concrete example in Online Appendix F.1. Formalizing the hierarchy in general is left to future work.

6.1.3. Step 2(b): Run Enhanced Versions of Existing Algorithms. The algorithm of Humair and Willems (2011) is a powerful exact method for the GSM. However, as the authors note, the method often identifies high-quality solutions quickly but may require substantially more time to certify optimality. Our LP relaxations can be used to strengthen the bounding step in their branch-and-bound scheme. In particular, one could use the lower bounds obtained from (4), or from higher levels of the hierarchy, to prune subproblems more aggressively. This suggests a hybrid approach in which the algorithm of Humair and Willems (2011) supplies feasible solutions and upper bounds, whereas our LP relaxations supply stronger lower bounds.

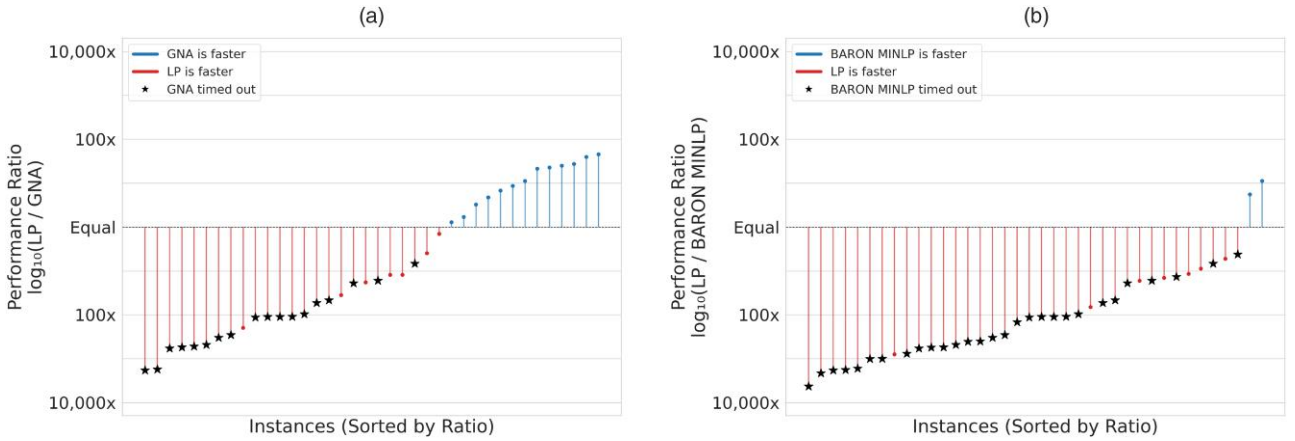
6.2. Computational Performance of Our Approach on Real-World Networks

We argue that our approach significantly outperforms all previous approaches (including commercial solvers) on practical instances, when the same hardware is used. Based on the prior GSM literature, we choose the real-world instances in Willems (2008) for benchmarking. Table EC.3 in Online Appendix F.3 reports structural characteristics for all 38 instances. The median treewidth of the intersection graphs is 6, confirming that real-world supply chains exhibit manageable structural complexity. The table also reports service-time domain bounds ($\max\Gamma_j$, $\max M_j$) and detailed solve times for each method.

We first contrast the DP in Humair and Willems (2011), called General Networks Algorithm (GNA), against our LP approach. We implement the DP based on the electronic companion to Humair and Willems (2011) and cap the solving time at two hours. For our LP approach, we follow the solution steps described in Section 6.1. As it turns out, Step 1 returns the optimal solution for all instances. As a result, we can solve even the largest instances in Willems (2008) in reasonable time. If the LP in (4) was not tight, we would have proceeded with Step 2(a) or 2(b). Additional implementation details can also be found in Online Appendix F.3.

The log-ratios of the total time to solve the LP (which includes building and solving times) to the time taken for the DP are provided in Figure 8(a). As shown, the LP approach yields the optimal solution for all instances within the two-hour time limit (in fact, within 30 minutes), whereas the DP times out for 18 out of 38 instances. Furthermore, the LP enables an improvement in the solving time over the DP for 25 out of the 38 instances, as indicated by the bars below the horizontal axis. The instances where the DP has a better performance than the LP are typically the smaller ones: the average (resp., median) number n of nodes in the graphs where the DP is faster is 177 (resp., 58), whereas the average number (resp., median) n of nodes in the graphs for all data sets is 252 (resp., 152). We highlight that for all instances with 1,000 or more nodes, the LP is dozens of times faster, even using a two-hour time cap when no optimal solution is found. The average (resp., median) time for building and solving the LP is 132 s (resp., 14 s), whereas the corresponding solving time of

Figure 8. (Color online) Comparison of our LP Framework with Two Alternative Solution Approaches on the Real-World Networks in Willems (2008)



Notes. (a) The DP Algorithm GNA in Humair and Willems (2011). (b) The Commercial Solver BARON.

GNA is 3,462 s (resp., 568 s), a speed-up of over an order of magnitude. The implementation of the LP is very straightforward, because it uses the standard LP solver Gurobi.

We also compare our LP approach with a mixed-integer nonlinear programming (MINLP) implementation in BARON, one of the few commercial solvers capable of solving nonconvex integer optimization problems such as (2) (Sahinidis 1996). The log-ratios of the solving times are given in Figure 8(b). Here, we use the same time cap of two hours, which BARON hits 29 times out of 38. Furthermore, the LP improves over the runtime of BARON in all but two instances (small instances 9 and 10). Overall, the average (resp., median) time for solving (2) with BARON is 5,590 s (resp., 7,200 s).

To ensure that the relatively poor performance of BARON is not due to the presence of integrality constraints in (2), we further compare our LP approach against a nonlinear programming (NLP) formulation of (2), where we drop the integrality constraints. When T_j and d_j are integers for $j \in V$ and $f_j(\cdot)$ is concave in (2)—assumptions which are met for our experiments—this NLP formulation can be shown to be equivalent to the MINLP (2) via, for example, total unimodularity. Solving times may differ, however, because different heuristics are used for each formulation. We solve the NLP using BARON on the same data with the same two-hour time cap. The speed-ups obtained are modest: the average solving time is 5,312 s, the median solve time remains at the timeout, and the LP is faster than the NLP in 30 of the 38 instances (see Figure EC.13b in Online Appendix F.3). This confirms that the computational difficulties in solving (2) are largely due to the nonconvex objective.

6.3. Other Managerial Considerations

Finally, we consider two important questions that supply chain managers may face: How does a change in a specific parameter impact the total cost of safety stock placement? How can they modify the problem—for example, by incorporating additional constraints—and how does this affect its solvability?

6.3.1. Leveraging LP Duality for Sensitivity Analyses.

We demonstrate how to utilize LP duality to assess the sensitivity of the GSM's optimal objective to certain problem parameters, such as the service-level constraints $\{d_j\}_{j\in L}$. This can be valuable for a manager who does not necessarily want to rerun all simulations, but wants to understand the impact that different parameters can have on the optimal solution. Note that this would be challenging to do using (2), because it is nonconvex and integer, and equally challenging for DP formulations. For ease of exposition, we focus on the case where the network is a tree (see Section 3), though our procedure is more generally applicable.

Recall the following well-known proposition on bounding changes in the right-hand side of linear programs (a short proof is in Online Appendix F.4).

Proposition 5. *Consider the LP*

$$\min_{\boldsymbol{x}\in\mathbb{R}^n,\ A\boldsymbol{x}\le\boldsymbol{b},\ G\boldsymbol{x}=\boldsymbol{h},\ \boldsymbol{x}\ge 0} \boldsymbol{c}^T\boldsymbol{x}, \qquad (LP_{\boldsymbol{b}})$$

where $A \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, $G \in \mathbb{R}^{p\times n}$, *and* $\boldsymbol{h} \in \mathbb{R}^p$. *Let* $D = \{\boldsymbol{b} \in \mathbb{R}^m \mid (LP_{\boldsymbol{b}}) \text{ is feasible}\}$, *and for each* $\boldsymbol{b} \in D$, *denote the optimal value by* $\Phi(\boldsymbol{b})$. *Suppose that we solved* $(LP_{\boldsymbol{b}})$ *for* r *distinct right-hand-side vectors* $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_r$, *yielding dual solutions* $(\boldsymbol{y}_{\boldsymbol{b}_1}, \boldsymbol{\nu}_{\boldsymbol{b}_1}), \ldots, (\boldsymbol{y}_{\boldsymbol{b}_r}, \boldsymbol{\nu}_{\boldsymbol{b}_r})$. *Then, for any* $\bar{\boldsymbol{b}} \in D$,

$$\Phi(\bar{\boldsymbol{b}}) \ge \max_{i\in\{1,\ldots,r\}} \{\bar{\boldsymbol{b}}^T\boldsymbol{y}_{\boldsymbol{b}_i} + \boldsymbol{h}^T\boldsymbol{\nu}_{\boldsymbol{b}_i}\}.$$

Note that the right-hand-side parameters $\{d_j\}_{j\in L}$ in (2) do not appear as the right-hand-side parameters in (4), making the use of Proposition 5 difficult. However,

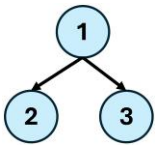
one can reformulate (4) as another LP, given in (EC.10) in the Online Appendix, in such a way that changing the parameters $\{d_j\}_{j\in L}$ only impacts the right-hand side of this LP, allowing the above dual-based bound to be applied directly. We refer the reader to Online Appendix F.4 and Proposition EC.10 for details on this reformulation.

We show how to use this result in practice. Consider the three-node tree in Figure 9, with fixed parameters $T_1 = 1,\ T_2 = 3,\ T_3 = 2,\ h_1 = 1,\ h_2 = 2.5,\ h_3 = 3,\ k = 3,$ $\sigma = 1$, and $d_3 = 1$. We let $M_2 = T_1 + T_2 = 4$ and $M_3 = 3$. Suppose that we solve the GSM reformulated as an LP in (EC.10) for $d_2 = 1$ and $d_2 = 3$ (see Proposition EC.10 in the Online Appendix). Our goal is to *avoid* resolving the LP when $d_2 = 2$. From the dual solutions obtained at $d_2 = 1$ and $d_2 = 3$, we can directly compute a lower bound of 13.39, using Proposition 5 (the naive lower bound obtained from $d_2 = 3$ alone is 12). Meanwhile, a simple feasible solution[5] for $d_2 = 2$ provides an easy upper bound of 19.5. Hence, we know that the optimal value lies between 13.39 and 19.5, *without* solving the LP with $d_2 = 2$. In fact, solving directly for $d_2 = 2$ yields 19.5, so these bounds are valid and nontrivial.

6.3.2. Integrating Additional Constraints. Although the GSM provides a robust framework for optimizing inventory placement in a supply chain network, and the ability to account for arbitrary cost functions allows for many extensions, real-world considerations might require additional constraints in (1). We now discuss how these can be integrated within our LP framework. For illustrative purposes, we consider the tree case in Section 3 and the concrete example of base stock constraints. Similar ideas can be developed for the general treewidth setting and different constraints.

Companies often impose upper limits on inventory at specific nodes due to risk exposure in volatile regions, limited storage capacity, tariffs, or operational limitations. For example, firms with parts of their supply chains in areas prone to natural disasters or political instability may limit the maximum inventory held in these areas to mitigate potential losses. Similarly, storage capacity constraints or working capital considerations might generate restrictions on the amount of inventory held at certain locations. This can be modeled using a total base stock constraint for a subset of nodes $V' \subseteq V$. For nodes $j \in V'$, let $B^{\max}$ denote the maximum aggregate base stock level allowable at these nodes. The total inventory collectively held by the nodes in V' can never exceed $B^{\max}$.

Figure 9. (Color online) Toy Example for Sensitivity Analysis



Recalling that the base stock at node j is given by $B_j = \bar{D}_j(SI_j + T_j - S_j)$ (see Section 2), we impose the following additional constraint in the GSM:

$$\sum_{j\in V'} \bar{D}_j(SI_j + T_j - S_j) \le B^{\max}, \tag{14}$$

which ensures that total base stock in V' does not exceed $B^{\max}$. In other words, we wish to solve (1) with the additional Constraint (14), which we refer to as (1)+(14). This can be done via an LP, as discussed: derive the intersection graph of (1)+(14), obtain its tree decomposition, associate variables to each bag in the tree and to each set of overlapping variables in neighboring bags, and rewrite (1)+(14) as an LP using these variables and appropriate constraints.

The challenge of this approach is that (14) can significantly increase the treewidth of the intersection graph of the GSM. For example, assume that a supply chain is a tree, so the treewidth of the intersection graph is one. Adding (14) introduces new edges in the intersection graph between any two variables that appear in (14). Hence, the treewidth of the intersection graph of (1)+(14) is at least $2|V'| - 1$. If $|V'|$ is large, this can drastically lengthen the solving time of (1)+(14).

Instead, we propose a *relaxation* of (1)+(14) providing a lower bound on the optimal value of (1)+(14). The key idea is to incorporate (14) into LP (4) using the fact that $\{r^j_{k_j\ell_j}\}$ can be interpreted as the indicator functions of $(S_j = k_j, SI_j = \ell_j)$ for $j \in V$. Hence, rewrite (14) as

$$\sum_{j\in V'} \sum_{k_j=0}^{\Gamma_j} \sum_{\ell_j=0}^{M_j} \overline{D}_j(\ell_j + T_j - k_j) r^j_{k_j\ell_j} \le B^{\max}. \tag{15}$$

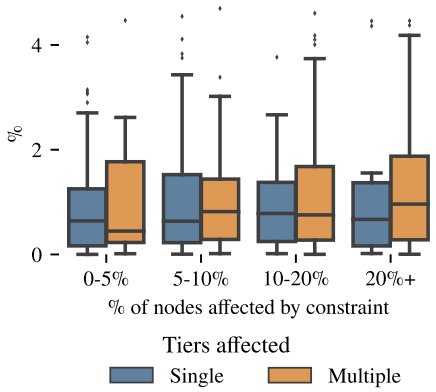
This yields a relaxation of the original GSM with Constraint (14), because every feasible solution to (1)+(14) can be mapped to a feasible solution of (4)+(15), but the converse is not true. Consequently, the objective obtained from (4)+(15) is a valid *lower* bound on the optimal cost for (1)+(14).

Our numerical experiments, detailed in Online Appendix F.5 and illustrated in Figure 10, demonstrate that the lower bound obtained is very tight across a considerable set of underlying graphs and base stock constraints. Even when V' is large or spans multiple tiers, the gap remains small.

7. Conclusion

We introduce a novel approach for optimizing the Guaranteed Service Model on arbitrary supply chain networks with a general objective function. This approach involves an exact reformulation of the GSM as a linear program, whose size is exponential in the network's

Figure 10. (Color online) Relative Optimality Gap of (4)+(15) to (1)+(14)



Notes. The construction of networks and the solving of instances are explained in detail in Online Appendix F.5. We differentiate between a set V' that affects only a single tier of nodes and a set V' that affects nodes at multiple tiers.

treewidth but polynomial in the number of nodes in the network and pseudo-polynomial wrt the bit size of the maximum replenishment time. The techniques we use to provide this reformulation are novel and may be of independent interest in other operations management problems.

Through our results, we unify and improve on the computational complexity of the GSM from prior literature. In addition to its complexity advantages, our formulation enables us to leverage the theoretical properties of LPs to derive principled bounds and establish sensitivity analyses. It also permits practitioners to use "off-the-shelf" LP software—which has undergone dramatic improvements in recent years (Applegate et al. 2021)—to solve the GSM, rather than having to implement bespoke dynamic programs. Using our approach, we achieve orders-of-magnitude improvements in GSM optimization time for the real data sets in Willems (2008).

Looking forward, our findings open several research avenues. First, new LP optimizers are being developed for GPUs, such as Nvidia's CuOPT. It would be worthwhile to explore the speed-up enabled by our approach when employing such infrastructure. Second, further developing the hierarchy of lower-bound LPs introduced in Section 6.1 could yield highly effective computational approaches for the GSM. Third, one may extend the GSM framework to closed-loop supply chain networks and apply our solution approach there. Fourth, the approach of reformulating an optimization problem over a supply chain network as an LP whose complexity depends on the underlying network's structure (in particular, the treewidth) likely applies to several other Operations Management problems, beyond the Discrete Time-Cost Tradeoff Problem discussed in Section 5.1.

Acknowledgments

The authors thank the department editor, Prof. Jeannette Song, the associate editor, and the reviewers for their constructive comments and support throughout the revision process. The authors also express their gratitude to seminar participants at HEC Paris, Johns Hopkins University, Singapore Management University, and the University of Virginia for their helpful feedback. Finally, Andre Calmon acknowledges the research cyberinfrastructure resources and services provided by the Partnership for an Advanced Computing Environment at the Georgia Institute of Technology.

Endnotes

[1] An exception is Blaettchen et al. (2025), who employ treewidth-based techniques to examine the optimization of technology diffusion on supply chains.

[2] The assumption of a one-to-one input-output ratio is made for ease of exposition; the model readily extends to the more general case in which one unit at j requires $\phi_{ij} \in \mathbb{N}_+$ units from i.

[3] Under a base stock policy, whenever the inventory position falls below a target (the base stock level), an order is immediately issued to replenish the inventory position to that target.

[4] We thank the Associate Editor for pointing out this nontrivial connection.

[5] Set $S_j = d_j$ for all $j \in L$, and then $SI_j = \max\{0, S_j - T_j\}$ and $S_i = \min_{\{j|(i,j)\in E\}}\{SI_j\}$ iteratively from the leaves upward.

References

Applegate D, Díaz M, Hinder O, Lu H, Lubin M, O'Donoghue B, Schudy W (2021) Practical large-scale linear programming using primal-dual hybrid gradient. *Adv. Neural Inform. Processing Systems* 34:20243–20257.

Arnborg S, Corneil DG, Proskurowski A (1987) Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods* 8(2):277–284.

Bienstock D, Muñoz G (2018) LP formulations for polynomial optimization problems. *SIAM J. Optim.* 28(2):1121–1150.

Blaettchen P, Calmon AP, Hall G (2025) Traceability technology adoption in supply chain networks. *Management Sci.* 71(1):83–102.

Bodlaender HL (1993) A tourist guide through treewidth. *Acta Cybernetica* 11(1–2):1–21.

Bodlaender HL (1996) A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* 25(6): 1305–1317.

De P, Dunne EJ, Ghosh JB, Wells CE (1995) The discrete time-cost tradeoff problem revisited. *Eur. J. Oper. Res.* 81(2):225–238.

De P, Dunne EJ, Ghosh JB, Wells CE (1997) Complexity of the discrete time-cost tradeoff problem for project networks. *Oper. Res.* 45(2):302–306.

Dell H, Komusiewicz C, Talmon N, Weller M (2018) The PACE 2017 parameterized algorithms and computational experiments challenge: The second iteration. Lokshtanov D, Nishimura N, eds. *12th Internat. Sympos. Parameterized Exact Comput. IPEC 2017*, Leibniz International Proceedings in Informatics (LIPIcs), vol. 89 (Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Wadern, Germany), 30:1–30:12.

Downey RG, Fellows MR (2012) *Parameterized Complexity* (Springer Science & Business Media, New York).

Eruguz AS, Sahin E, Jemai Z, Dallery Y (2016) A comprehensive survey of guaranteed-service models for multi-echelon inventory optimization. *Internat. J. Production Econom.* 172:110–125.

Graves SC, Willems SP (2000) Optimizing strategic safety stock placement in supply chains. *Manufacturing Service Oper. Management* 2(1):68–83.

Humair S, Willems SP (2006) Optimizing strategic safety stock placement in supply chains with clusters of commonality. *Oper. Res.* 54(4):725–742.

Humair S, Willems SP (2011) Optimizing strategic safety stock placement in general acyclic networks. *Oper. Res.* 59(3):781–787.

Humair S, Ruark JD, Tomlin B, Willems SP (2013) Incorporating stochastic lead times into the guaranteed service model of safety stock optimization. *Interfaces* 43(5):421–434.

Inderfurth K (1991) Safety stock optimization in multi-stage inventory systems. *Internat. J. Production Econom.* 24(1–2):103–113.

Jansen BM, Kratsch S (2015) A structural approach to kernels for ILPs: Treewidth and total unimodularity. *Algorithms ESA 2015 Proc. 23rd Annual Eur. Sympos.* (Springer, Berlin, Heidelberg), 779–791.

Kloks T (1994) *Treewidth: Computations and Approximations* (Springer, Berlin, Heidelberg).

Klosterhalfen ST, Minner S, Willems SP (2014) Strategic safety stock placement in supply networks with static dual supply. *Manufacturing Service Oper. Management* 16(2):204–219.

Laurent M (2009) Sums of squares, moment matrices and optimization over polynomials. Putinar M, Sullivant S, eds. *Emerging Applications of Algebraic Geometry*, The IMA Volumes in Mathematics and its Applications, vol. 149 (Springer, New York), 157–270.

Lesnaia E (2004) Optimizing safety stock placement in general network supply chains. PhD thesis, Massachusetts Institute of Technology, Cambridge.

Lesnaia E, Vasilescu I, Graves S (2005) The complexity of safety stock placement in general-network supply chains. *Proc. 2005 SMA Internat. Conf.* Accessed January 30, 2025, https://dspace.mit.edu/bitstream/handle/1721.1/7537/IMST033.pdf.

Li H, Jiang D (2012) New model and heuristics for safety stock placement in general acyclic supply chain networks. *Comput. Oper. Res.* 39(7):1333–1344.

Magnanti TL, Shen ZJM, Shu J, Simchi-Levi D, Teo CP (2006) Inventory placement in acyclic supply chain networks. *Oper. Res. Lett.* 34(2):228–238.

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

Moncayo-Martínez LA, Mastrocinque E (2025) Solving the guaranteed-service time inventory model for real-world supply chains by implementing it in commercial optimisers. *J. Engrg. Res.* 13(3):2559–2571.

Muñoz G (2017) Integer programming techniques for polynomial optimization. PhD thesis, Columbia University, New York.

Sahinidis NV (1996) BARON: A general purpose global optimization software package. *J. Global Optim.* 8(2):201–205.

Schoenmeyr T, Graves SC (2022) Coordination of multiechelon supply chains using the guaranteed service framework. *Manufacturing Service Oper. Management* 24(3):1859–1871.

Shu J, Karimi I (2009) Efficient heuristics for inventory placement in acyclic networks. *Comput. Oper. Res.* 36(11):2899–2904.

Simpson KF Jr (1958) In-process inventories. *Oper. Res.* 6(6):863–873.

Skutella M (1998) Approximation algorithms for the discrete time-cost tradeoff problem. *Math. Oper. Res.* 23(4):909–929.

Wainwright MJ, Jaakkola TS, Willsky AS (2005) MAP estimation via agreement on trees: Message-passing and linear programming. *IEEE Trans. Inform. Theory* 51(11):3697–3717.

Willems SP (2008) Real-world multiechelon supply chains used for inventory optimization. *Manufacturing Service Oper. Management* 10(1):19–23.