Management Science

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

Approximation Algorithms for Dynamic Inventory Management on Networks

Levi DeValve, Jabari Myles

To cite this article:
Levi DeValve, Jabari Myles (2025) Approximation Algorithms for Dynamic Inventory Management on Networks. Management Science 71(7):5893-5909. https://doi.org/10.1287/mnsc.2022.02965

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

MANAGEMENT SCIENCE
Vol. 71, No. 7, July 2025, pp. 5893–5909
ISSN 0025-1909 (print), ISSN 1526-5501 (online)

Approximation Algorithms for Dynamic Inventory Management on Networks

Levi DeValve,[a,*] **Jabari Myles**[b]

[a] Booth School of Business, University of Chicago, Chicago, Illinois 60637; [b] Operations Research, North Carolina State University, Raleigh, North Carolina 27695

*Corresponding author

Contact: levi.devalve@chicagobooth.edu, https://orcid.org/0000-0002-4730-4541 (LD); jabari_myles@ncsu.edu, https://orcid.org/0009-0001-8968-0756 (JM)

Received: September 22, 2022
Revised: October 30, 2023; March 22, 2024
Accepted: May 7, 2024
Published Online in Articles in Advance: October 18, 2024

https://doi.org/10.1287/mnsc.2022.02965

Abstract. We provide the first approximation algorithm for dynamic inventory management on a network with stochastic demand and backlogging. Specifically, under a mild cost condition, we prove the cost of a specially designed base-stock policy is less than 1.618 times the cost of an optimal policy. We develop a novel stochastic programming analysis to prove this result: We carefully calibrate two stochastic programs (providing upper and lower bounds on the optimal policy), and compare their objectives. The upper bound arises from a new class of base-stock policies we define to address the currently unresolved issue of how to assign and fulfill backlogs in a system with fulfillment flexibility. We show the optimal policy in this class takes a simple and intuitive form: backlogs are assigned to their lowest cost activities for replenishment, and the ordered resources for those activities are committed to the backlogs for fulfillment. Next, the lower bound stochastic program is derived through a novel cost accounting scheme that captures the tradeoff between current inventory decisions and future backlog decisions. We then exploit this tradeoff to bound the ratio of the two stochastic program objectives and prove our main result. We also demonstrate our policy's practicality with numerical simulations that show it performs within 1% of optimal on average across a wide range of problem instances. Finally, we show that our techniques and results extend to more general settings, demonstrating their potential for broader applicability. Importantly, our approach extends to problems with lead times, where we prove an approximation guarantee for base-stock policies that is independent of both the lead time and network structure. Thus, our work provides the new managerial insight that properly designed base-stock policies can be effective in network settings.

History: Accepted by Victor Martínez-de-Albéniz, operations management.

Supplemental Material: The online appendix and data files are available at https://doi.org/10.1287/mnsc.2022.02965.

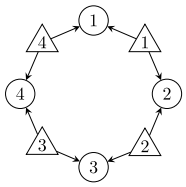
Keywords: inventory management • stochastic program • newsvendor network

1. Introduction

In today's complex global economy, inventory is rarely managed for individual products in isolated locations. Rather, multiple resources are often stored in several locations before being flexibly deployed to meet arriving customer demand. Examples include flexible manufacturing (e.g., automotive, consumer electronics), e-commerce fulfillment, and omni-channel retailing. In each of these settings, several *resources* (e.g., parts, raw materials, individual SKUs, etc.) need to be stocked in anticipation of uncertain demand; then, as demand is realized for multiple products, a number of *activities* (e.g., production, assembly, fulfillment, etc.) need to be executed in order to fulfill demand. An important feature of such systems is that they employ both *resource commonality* (i.e., each resource may be used by multiple activities) and *fulfillment flexibility* (i.e., each product may be fulfilled by multiple activities). The operations literature has called such models *newsvendor networks* (Van Mieghem and Rudi 2002), which we also adopt in this paper.

For a simple illustrative example, consider the fulfillment network depicted in Figure 1, where triangles represent warehouses and circles represent demand regions (i.e., cities or zip codes where demand for products originates). In this simple network, each warehouse is capable of fulfilling demand from two regions, illustrating fulfillment flexibility. The warehouses may also stock multiple items, and be capable of fulfilling

Figure 1. Fulfillment Network



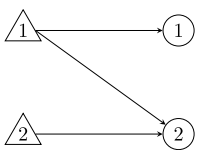
Note. Triangles are warehouses, and circles are demand regions.

orders for multiple items, illustrating resource commonality. This type of fulfillment network is typical of many e-commerce (Jasin and Sinha 2015), omni-channel retailing (Govindarajan et al. 2021b), flexible production (Jordan and Graves 1995), and spare-parts inventory management (Kranenburg and Van Houtum 2009) settings.

The proliferation of newsvendor networks in modern commerce has motivated much academic interest in the topic, but dynamic inventory control with demand backlogging has remained an open question. The key challenge, first identified by Van Mieghem and Rudi (2002), is that fulfillment flexibility complicates classic methods for replenishing inventory of resources to clear product backlogs. The classic approach, known since Scarf (1959), is to map product backlogs to their required resources and replenish inventory according to a base-stock (or order-up-to) policy accounting for these backlogs. The practical issue with applying this approach to newsvendor networks is that fulfillment flexibility makes it unclear how to map backlogs to resources, leaving managers with little insight for implementing classic inventory control. The underlying theoretical issue is that the lack of a clear backlog mapping eliminates the state-space reduction (i.e., the well-known "inventory position") needed to establish optimality of a base-stock policy.

This issue can be illustrated with the network in Figure 2, where product 1 can be fulfilled by resource 1, and product 2 can be filled by either resource 1 or 2. If this system has a backlog of product 2, a policy guided by classic inventory theory would order a resource to fulfill the backlog plus enough to get up to a base-stock level. However, the flexible system faces a choice of which resource to order to fulfill the backlog. As described in Van Mieghem and Rudi (2002), it may appear cost effective ex ante (i.e., before demand is realized) to assign the backlog to resource 1, but ex post (i.e., after demand is realized) resource 1 may be diverted to fulfill higher priority product 1 demand, in which case it would have been better to assign the backlog to resource 2. This difference between optimal ex ante and ex post decisions prevents collapsing the inventory and backlog states into the unified "inventory position" of classic inventory theory, leaving scant understanding of how to design good inventory control policies in this setting.

Figure 2. Network with Two Products (Circles) and Two Resources (Triangles)



Motivated by these challenges, two open questions considered in this research are (i) how to assign product backlogs to resources for replenishment and (ii) how to prioritize product fulfillment once replenishment is received. Further, as base-stock policies (or their variants) are often used in practice, we also consider the question of how well a base-stock policy can perform relative to optimal. We provide answers to each of these questions, which we describe in our summary of major contributions next.

1.1. Novel Family of Lower Bounds

We develop a novel family of stochastic program lower bounds on the cost of an optimal policy. The key novelty in our derivation is taking convex combinations of costs across different periods, allowing us to balance current and future costs to obtain tighter bounds. In particular, our approach allows us to characterize a tradeoff between the cost of current inventory decisions and future backlog fulfillment decisions. This allows us to prove new approximation guarantees (described below) in a range of settings, facilitated by the flexibility of the lower bounds.

1.2. Intuitive Class of Base-Stock Policies

Next, we construct a class of base-stock policies that address the questions of how to assign and fulfill backlogs. The policy is straightforward: it fixes a backlog assignment rule that maps backlogs to resources in fixed proportions, orders according to a base-stock policy accounting for this backlog assignment, and then commits to fulfilling backlogs with their assigned resources when a replenishment arrives. This rule allows us to characterize the evolution of the system and evaluate the expected cost of a policy within this class, which we call *assigned backlog base-stock* (ABBS) policies. With our cost characterization, we are then able to solve a single stochastic program to optimize the base-stock levels and fulfillment decisions, which we use to show the optimal backlog assignment rule is simple and intuitive: each product maps backlog to its lowest cost activity. Thus, ABBS policies offer the theoretical advantage of resolving the backlog assignment and fulfillment question in a way that allows for straightforward policy evaluation and optimization. Further, these policies also offer two main practical benefits: (i) backlogs are assigned in a predictable way, allowing consistency and efficiency in implementation, and (ii) backlogs are filled as soon as possible, providing a positive customer experience.

1.3. Performance Guarantees

Our final contribution is to establish several approximation guarantees for the class of ABBS policies relative to optimal. First, our main result proves a constant factor approximation guarantee of 1.618 under a mild condition on the holding costs. To the best of our knowledge, this is the first approximation algorithm, not only for dynamic newsvendor network problems but for stochastic multiproduct inventory management problems in general. Our stochastic programming approach is key to deriving this result: We directly compare the costs of the upper and lower bound stochastic programs. Then we extend this performance guarantee in two important directions. First, relaxing the cost condition, we derive another approximation guarantee that can be roughly thought of as scaling with the reciprocal of the system's service level (i.e., practical for the many systems in industry that target a high service level). Further, we also extend our policies to systems with lead times and provide two approximation guarantees using our stochastic programming approach: the first scales with the square root of both the lead time and the size of the largest activity in the system, and the second is independent of both the lead time and network structure, and depends only on the cost and demand parameters of the system. These bounds demonstrate the versatility of our framework, and open the door for future research into effective inventory management on networks.

The rest of the paper is organized as follows. We review related literature in more detail in the remainder of the introduction. Section 2 introduces our formal model and Section 3 derives our main stochastic program lower bound. Section 4 develops the class of ABBS policies and proves our main approximation guarantees for the case with no lead times. We then extend our approximation results to lead times in Section 5. For conciseness, mathematical proofs are given in the online appendices.

1.4. Literature Review

Newsvendor networks have been studied in various models and contexts for the past few decades. Van Mieghem (1998), Harrison and Van Mieghem (1999), and Van Mieghem and Rudi (2002) offer some of the earliest formal models, which have been followed up by research on a range of topics including network design (Van Mieghem 2004, Bassamboo et al. 2010), risk mitigation (Tomlin and Wang 2005, Van Mieghem 2007), and optimization (Govindarajan et al. 2021a, DeValve 2023, Jiang et al. 2023). Our work builds most directly on Van Mieghem and Rudi (2002), who first identified the difficulty of mapping backlogs to resources while attempting to extend the single-stage solution of a general newsvendor network to a dynamic setting. They were successful with a lost sales model but were only able to establish optimality of a base-stock policy in a backlog model under several restrictions on the demand, cost, and network structure that essentially require a unique optimal basis in the second-stage linear program for any demand realization and thus a consistent optimal backlog mapping. We build on this result by proving in a much broader class of networks (subject to just one mild cost condition) that a base-stock policy with an intuitive backlog assignment rule is guaranteed to perform within a constant factor of optimal.

The defining characteristics of newsvendor networks are resource commonality and fulfillment flexibility. However, owing to the general nature of the model, different authors have placed varying emphasis on these characteristics, and several alternative modeling details exist across the literature. For example, Harrison and Van Mieghem (1999) consider only resource commonality, Tomlin and Wang (2005), Bassamboo et al. (2010), Govindarajan et al. (2021a), and Birge and DeValve (2024) consider just flexibility, and Van Mieghem and Rudi (2002) and Van Mieghem (2004) model both. Our model is among the most general in the literature, as we include both resource commonality and fulfillment flexibility in general network structures. Essentially the only feature not allowed in our model is coproduction (i.e., using the same activity to produce multiple products; Tomlin and Wang 2008). We also note that, although Van Mieghem and Rudi (2002) include capacity decisions in their single-stage analysis, their dynamic results are for uncapacitated systems (same as ours).

Our paper also builds on a very long stream of research analyzing policies for dynamic and stochastic inventory systems with backlogging. A classic insight from this literature, known as early as Scarf (1959), is that the state variable can be effectively transformed to deal with *inventory position* (i.e., on-hand plus pipeline inventory minus backlogs) that naturally leads to the optimality of base-stock policies (alternatively called order-up-to or, more generally, (s, S) policies). This insight has been applied broadly, including, for example, inventory management with multiple delivery modes, risk aversion, and pricing (Feng et al. 2006; Chen and Sun 2012; Chen et al. 2016, 2019). Of particular relevance to our setting is the assemble-to-order (ATO) problem, which models resource commonality but lacks fulfillment flexibility. Thus, an ATO system has only one way to map product backlogs to resources, and the standard techniques allow analyzing base-stock policies (Lu and Song 2005, Lu et al. 2010, DeValve et al. 2023a), which have been shown to be optimal or asymptotically optimal in a variety of settings (Lu et al. 2015, Reiman and Wang 2015, Doğru et al. 2017, DeValve et al. 2020, Chen et al. 2021, Song and Xue 2021, Zhao et al. 2023). However, as mentioned earlier, how product backlogs should be assigned to resources has remained an open question in the literature for systems with fulfillment flexibility. We contribute to this literature by providing both a method for assigning backlogs to resources when

there is fulfillment flexibility and the first performance guarantee for a base-stock policy in this setting.

Other recent work on network inventory management includes Akturk (2022), Akturk et al. (2024), and Jiang et al. (2022), who establish asymptotic optimality of rebalancing policies for networks with reusable resources, as well as Govindarajan et al. (2021b), who show asymptotic near-optimality of an order-up-to policy in a single-product network with lost sales. Further, Hu et al. (2008) and Chen et al. (2015) characterize the form of an optimal policy in a two-location network with uncertain capacities. Heuristic policies are proposed and numerically tested in Acimovic and Graves (2017) for a network replenishment/allocation problem and in Qin et al. (2022) for a network inventory positioning problem. Relative to this literature, our paper proves the first performance guarantee in a finite (i.e., nonasymptotic) regime for a more general multiproduct, multiresource network with backlogging. Further, our problem bears similarities with the classic literature on safety stock placement (Graves and Willems 2000, 2008; Schoenmeyr and Graves 2009; Graves and Schoenmeyr 2016), which typically assumes the use of a base-stock policy, whereas we prove near-optimality of a base-stock policy in general newsvendor networks.

We also note that our model and network structure bear significant similarities with the growing literature on e-commerce fulfillment (Acimovic and Graves 2015, DeValve et al. 2023b), especially for multi-item orders (Jasin and Sinha 2015, Cheung et al. 2022, Amil et al. 2023, Ma 2023). The main difference is that, although the aforementioned literature is focused on the fulfillment problem with a fixed stock of initial resources, our work considers both resource replenishment and fulfillment.

Our work also relates to the literature on approximation algorithms for dynamic inventory problems. Starting with Levi et al. (2007), a long line of work has established constant factor approximation guarantees for dynamic *single-product* problems with stochastic demand, including with lost sales (Levi et al. 2008a), capacity constraints (Levi et al. 2008b), and perishable inventory (Chao et al. 2015, 2018; Zhang et al. 2016, 2023). The majority of this stream of research relies on cost accounting schemes that balance expected future inventory cost against other costs in the system (shortage, backlog, perishability, etc.), which is well suited for analyzing single product systems. In this sense, our work is somewhat tangential to the methodology of this literature as our result requires developing a different cost accounting scheme based on stochastic program lower bounds. More broadly, however, we contribute to this literature by providing the first approximation guarantee for a general multiproduct problem, which is possible because of our novel stochastic programming approach.

2. Model

In this section, we introduce the newsvendor network model. A newsvendor network uses M resources (indexed by i) to fill demand for N products (indexed by j) by means of K processing activities (indexed by k). Each processing activity uses inventory from a subset of the resources in order to fulfill demand for one of the products, whereas demand for each product can be fulfilled from a variety of activities. The defining features of a newsvendor network are (i) resource commonality, that is, the resources used by each activity may be shared with other activities and (ii) fulfillment flexibility, that is, each product may have multiple activities to choose from for demand fulfillment.

Before describing the rest of the model in detail, we first define several notational conventions used in the paper. Let $\mathbb{R}_+$ and $\mathbb{Z}_+$ denote the nonnegative reals and integers, respectively. For an event E, its indicator function is denoted $\mathbb{1}_{\{E\}}$, and its probability is denoted $\mathbb{P}[E]$. The notation $\mathbb{E}[X]$ denotes the expectation of a random variable X, and we use $(x)^+ = \max(x, 0)$ to denote the positive part of a number x. Time is discrete and indexed into periods $t = 1, 2, \ldots$.

2.1. Input Data

A particular newsvendor network is comprised of three sets of inputs: a network structure, a demand distribution, and cost parameters, which we describe next.

2.1.1. Network Structure. The network structure is composed of two integer matrices, a capacity consumption matrix, A, and a flexible fulfillment matrix R. The capacity consumption matrix A represents the resource requirements for each processing activity, with element $a_{ik} \in \mathbb{Z}_+$ representing the number of units of resource i needed per unit of activity k. If $a_{ik} > 0$, we say resource i *serves* activity k, and we let $\mathcal{S}(k) = \{i | a_{ik} > 0\}$ denote the set of resources serving activity k.

The flexible fulfillment matrix R represents the set of activities that can fill demand for each product, with element $r_{kj} = 1$ denoting that demand j can be filled by activity k, and $r_{kj} = 0$ denoting it cannot. Because each activity can fulfill demand for one product, we have $\sum_j r_{kj} = 1$ for each k, and for each k we let $j(k)$ denote the product that activity k serves (i.e., $j(k) = j$ for the j such that $r_{kj} = 1$). Finally, let $\mathcal{N}(j) = \{k | r_{kj} > 0\}$ denote the set of activities that can fulfill product j demand.

2.1.2. Demand Distribution. In each period t, there is random demand for product j denoted by $D_j^t \geq 0$. The demand vector $\mathbf{D}^t = (D_1^t, \ldots, D_N^t)$ is independent and identically distributed (i.i.d.) across periods but may be correlated between products within a given period. We assume $\mu_j = \mathbb{E}[D_j^t] < \infty$ for all j, t. We also let $\mathbf{D}$ represent a random vector with the same distribution as $\mathbf{D}^t$.

2.1.3. Cost Parameters. We consider four different types of costs. Let $q_k \geq 0$ denote the per unit cost of processing activity k, $c_i \geq 0$ denote the per unit cost of resource i inventory, $h_i \geq 0$ denote the per unit inventory holding cost for resource i per period, and $p_j \geq 0$ denote the per unit backlog cost for product j per period.

2.2. Evolution and Objective

To specify the evolution of the system, let $I_i^t \geq 0$ and $B_j^t \geq 0$ denote the on-hand inventory of resource i and backlog of product j, respectively, at the end of period t. Let x_k^t denote the processing level of activity k in period t (which is decided after period t demand is realized), and let z_i^t denote the order placed for resource i inventory at the beginning of period t, which is delivered after a lead time of L periods,[1] that is, at the beginning of period $t+L$. The system is assumed to start empty, that is, $I_i^0 = B_j^0 = 0$ for all i, j. The evolution equations for inventory and backlog, respectively, are

$$I_i^t = I_i^{t-1} + z_i^{t-L} - \sum_k a_{ik} x_k^t, \quad \forall i, \tag{1}$$

$$B_j^t = B_j^{t-1} + D_j^t - \sum_k r_{kj} x_k^t, \quad \forall j. \tag{2}$$

The goal is to determine a policy, π, which specifies the replenishment orders, $z_i^t \geq 0 \ \forall i$, and processing activity levels, $x_k^t \geq 0 \ \forall k$, in each period to minimize long-run average cost:

$$C(\pi) = \limsup_{T\to\infty} \frac{1}{T} \mathbb{E}\left[\sum_{t=1}^{T}\left(\sum_i (h_i I_i^t + c_i z_i^{t-L}) + \sum_j p_j B_j^t + \sum_k q_k x_k^t\right)\right]. \tag{3}$$

We note that, although this definition leaves out the cost of z_i^s for periods $s > T-L$ for any fixed T, this cost is inconsequential in the limit (and indeed could be included without altering our results). A feasible policy must be *nonanticipating*, meaning that z_i^t and x_k^t can only depend on the realized demand and executed decisions up to and including time $t-1$ and t, respectively. More specifically, a policy is nonanticipating if z_i^t depends only on $(\mathbf{D}^1, \ldots, \mathbf{D}^{t-1}, \mathbf{x}^1, \ldots, \mathbf{x}^{t-1}, \mathbf{z}^1, \ldots, \mathbf{z}^{t-1})$, and x_k^t depends only on $(\mathbf{D}^1, \ldots, \mathbf{D}^t, \mathbf{x}^1, \ldots, \mathbf{x}^{t-1}, \mathbf{z}^1, \ldots, \mathbf{z}^t)$.[2] Thus, the firm's problem can be stated as

$$\begin{aligned} \min \quad & C(\pi) \\ \text{s.t.} \quad & (1) \text{ and } (2), \\ & z_i^t \text{ and } x_k^t \text{ are nonanticipation}, \\ & z_i^t, x_k^t, I_i^t, B_j^t \geq 0 \ \ \forall i,k,j,t. \end{aligned} \tag{4}$$

Let C^* denote the minimum value of (4), that is, the optimal long-run average cost. It is well known that for general newsvendor networks, solving (4) exactly is intractable because of the curse of dimensionality and that the structure of an optimal policy is either unknown or highly complex (Van Mieghem and Rudi 2002). Thus, our goal in this paper is to identify a class of intuitive and implementable policies that are also guaranteed to perform well relative to an optimal policy. We therefore make the following standard definition of an approximately optimal policy.

Definition 1. A feasible policy π has an approximation factor of κ if for any problem instance, the policy's long-run average cost is guaranteed to satisfy $C(\pi) \leq \kappa C^*$.

3. Novel Family of Stochastic Program Lower Bounds

Here we lay the primary foundation for our analysis by developing a novel family of stochastic program lower bounds on the long-run average cost of an optimal policy. Each performance bound we derive in the remainder of the paper relies on a lower bound from the family we characterize below. The key idea in deriving these lower bounds is a cost accounting that incorporates weighted costs across multiple periods, allowing us to balance current and future costs to obtain tighter bounds. Before we provide further intuition however, we first formulate the stochastic program and state our main result and then follow up with a detailed explanation. To define the stochastic program, we specify three weight parameters, $\beta, \delta, \rho \in [0,1]$ (which will control weights of costs from different periods), aggregated cost parameters, $Q_k = \frac{1}{L+1}(q_k + \sum_i a_{ik} c_i)$, and $H_k = \sum_i a_{ik} h_i$, and let $\mathcal{D} = (\mathbf{D}^1, \ldots, \mathbf{D}^{L+1})$ denote a collection of $L+1$ i.i.d. copies of the period random demand vectors. Then the stochastic program is defined as follows:

$$G(\mathbf{S}, \mathbf{B}, \beta, \delta, \rho \mid \mathcal{D}) = \left\{ \begin{aligned} \min_{\mathbf{x},\mathbf{w}\geq 0,\mathbf{y}} \ & \sum_k \left((Q_k - H_k) \sum_{l=1}^{L+1} (x_k^l + \delta w_k^l) + (Q_k - \beta H_k) \sum_{l=2}^{L+2} (1-\delta) w_k^l \right) \\ & + (1-\rho) \sum_j p_j \left(\delta y_j^0 + (1-\delta) y_j^{L+2} + \sum_{l=1}^{L+1} y_j^l \right) \\ \text{s.t.} \ & \sum_{l=1}^{L+1} \sum_k a_{ik} (x_k^l + w_k^l) \leq S_i, \ \forall i \\ & \sum_k r_{kj} w_k^1 + y_j^0 = B_j, \ \forall j \\ & \sum_k r_{kj} x_k^1 + y_j^1 = D_j^1, \ \forall j \\ & \sum_k r_{kj} (x_k^l + w_k^l) + y_j^l = D_j^l, \ \forall j, 1 < l \leq L+1 \\ & \sum_k r_{kj} w_k^{L+2} + y_j^{L+2} = \sum_{l=0}^{L+1} y_j^l, \ \forall j \\ & \sum_k r_{kj} w_k^1 \leq B_j, \ \forall j \\ & \sum_k r_{kj} w_k^l \leq \sum_{l'=0}^{l-1} y_j^{l'}, \ \forall j, 1 < l \leq L+2 \\ & \sum_k r_{kj} x_k^l \leq D_j^l, \ \forall j, 1 \leq l \leq L+1 \end{aligned} \right\}, \tag{}$$

$$\underline{G}(\beta,\delta,\rho) = \min_{\mathbf{S}\geq 0,\mathbf{B}\geq 0} \beta\sum_i h_i S_i + \rho\sum_j p_j B_j + \mathbb{E}[G(\mathbf{S},\mathbf{B},\beta,\delta,\rho|\mathcal{D})]. \tag{5}$$

The next result states our main lower bound.

Theorem 1. *The stochastic program* (5) *for any* $\beta,\delta,\rho \in [0,1]$ *gives the following lower bound on the long-run average cost of any feasible policy* π: $\underline{G}(\beta,\delta,\rho) \leq C(\pi)$.

We now provide further intuition and details on the stochastic program and lower bound. At a high level, the stochastic program is similar to others in the literature (Reiman and Wang 2015, DeValve et al. 2020) as it derives a lower bound on the expected cost of each period by optimizing over the starting inventory position and backlog state vectors a lead time prior. These variables are represented by $\mathbf{S}$ (the base-stock levels) and $\mathbf{B}$ in (5), respectively. Intuitively, the backlog variables $\mathbf{B}$ are required in order to obtain a lower bound because an optimal policy may sometimes leave backlog for a given product in the system to allow higher levels of inventory available for fulfillment of other products (see Doğru et al. (2010) for an example illustrating the necessity of these variables when holding costs are large). The stochastic program then optimizes fulfillment decisions for demand arriving in the following lead time in the second stage linear program, $G(\mathbf{S},\mathbf{B},\beta,\delta,\rho|\mathcal{D})$, which relaxes the nonanticipating constraint to allow fulfillment decisions to be made after all demand has been realized, as in a hindsight optimal relaxation. Thus, at a high level, our stochastic program follows a similar strategy to the existing literature by planning inventory and fulfillment decisions to accommodate demand over $L+1$ periods.

There are also several notable differences between (5) and existing bounds in the literature. In the second stage linear program, $G(\mathbf{S},\mathbf{B},\beta,\delta,\rho|\mathcal{D})$, the variables $\mathbf{x}$ and $\mathbf{w}$ both model fulfillment, but x_k^l represents fulfillment using activity k for demand arriving in period l, whereas w_k^l represents fulfillment for backlog already in the system at the beginning of period l. Distinguishing between these two types of fulfillment is key to deriving our performance guarantees, as it allows us to capture the future cost of fulfilling backlog (see Section 4.2). In particular, note that we include a decision $\mathbf{w}^{L+2}$, denoting a backlog fulfillment decision for the period after the $L+1$ period planning horizon is over. Similarly, the $\mathbf{y}$ variables model backlog added to the system in each period, with the following conventions: $\mathbf{y}^0$ and $\mathbf{y}^1$ denote existing backlog and new demand that are left as backlog in the first period, $\mathbf{y}^l$ for $1 < l \leq L+1$ denote the change in backlog (resulting from any source) in periods 2 through $L+1$ (and thus these variables may be positive or negative, representing an increase or decrease in backlog, respectively), and $\mathbf{y}^{L+2}$ denotes backlog remaining from any source after the $L+1$ periods of demand plus the final backlog fulfillment decision $\mathbf{w}^{L+2}$. From these definitions it is straightforward to derive by induction that the product j backlog remaining from any source after l periods is $\sum_{l'=0}^{l} y_j^{l'}$.

The constraints of the second-stage linear program capture the natural evolution of the system over a lead time. The first constraint ensures all fulfillment uses no more inventory than the base-stock level of each resource. The second through fifth constraints define the backlog variables in terms of the fulfillment, demand, and starting backlog. The sixth and seventh constraints ensure the backlog fulfillment is less than the current backlog in the system, and the eighth constraint ensures the demand fulfillment is less than the current period's demand.

The objective of (5) includes the holding, backlog, ordering, and fulfillment costs, but uses the weights β,δ, and ρ to capture these costs across different periods, which is our key innovation. In particular, β governs a balance between capturing the holding cost in the current period and the backlog fulfillment cost in the following period, δ governs a balance between the backlog fulfillment costs and the backlog costs, and ρ governs a balance between backlog costs in the current period and a lead time ago. To derive the lower bound incorporating each of these weights, our proof simply takes a convex combination of costs across different periods and shifts the sum to gather terms into a single objective. We will see that this straightforward approach yields strong bounds in our subsequent analysis.

To provide intuition on the activity cost coefficient, we observe that $Q_k - H_k = \frac{1}{L+1}(q_k + \sum_i a_{ik}c_i) - \sum_i a_{ik}h_i$, which measures the unit cost incurred by activity k on the scale of a single period. Noting that the stochastic program includes demand over $L+1$ periods (i.e., the sum of x_k and w_k variables in the objective captures the demand fulfilled over $L+1$ periods), the activity cost q_k and ordering costs c_i are scaled by $1/(L+1)$ to get them on the scale of a single period. The holding costs, meanwhile, do not need to be scaled because these are already calibrated to the lead time, which follows from the classic inventory result that present on-hand inventory equals the inventory position a lead-time prior minus lead-time demand.

4. Developing a Base-Stock Policy

In this section, we turn our attention to developing a practical and effective class of base-stock policies. To focus our analysis, we first concentrate on the case with no lead time; that is, we assume $L = 0$ for the remainder of this section and defer the lead time analysis to Section 5. We design base-stock policies that specify both how to assign backlogs to activities, and how to fulfill backlogs. We then demonstrate how to optimize the backlog assignment, fulfillment policy, and base-stock levels

within this class of policies. In particular, we characterize the optimal policy within this class via the solution of a stochastic program, whose optimal objective value equals the long-run average cost of the policy. This characterization of the policy cost in terms of a stochastic program greatly aids our analysis in Section 4.2 where we develop a performance bound for this policy class.

4.1. Class of Base-Stock Policies

To begin our development, let $S_i \geq 0$ denote the base-stock level for resource i and let $\mathbf{S}$ denote a vector of base-stock levels for all the resources. A standard base-stock policy aims to keep the *inventory position* for resource i at the constant level S_i, where the inventory position refers to the sum of on-hand inventory, newly ordered inventory, and backlogs. However, as discussed earlier, the complicating feature of newsvendor networks is that it is unclear how to translate product backlogs into resource backlogs in order to execute such a policy. We consider this issue next.

4.1.1. Backlog Assignment. Here we specify a rule for assigning product backlogs to activities. Although there are many possible ways to assign backlogs, we aim to provide a simple approach that maintains straightforward intuition, which is more likely to be implemented in practice. In this vein, we focus on state-invariant mappings, meaning that product backlogs are always assigned to activities in the same proportions, regardless of the state of the system. This approach offers two main advantages. First, we will see in Section 4.1.3 that it allows us to easily track and optimize the system costs. Second, as we show in Section 4.2.2, it is guaranteed to be close to optimal.

To specify the policy, let V be a $K \times N$ matrix representing the assignment of product backlogs to activities; that is, v_{kj} represents the proportion of product j backlog that will be filled by activity k. Naturally, we require that backlogs are only assigned in nonnegative quantities to activities that can fulfill a given product, that is, $0 \leq V \leq R$, and that all backlogs are assigned, that is, $\sum_k r_{kj} v_{kj} = 1$ for each j. Then, in period t we use the following base-stock ordering policy for each resource i

$$z_i^t = \left(S_i - I_i^{t-1} + \sum_k a_{ik} v_{kj(k)} B_{j(k)}^{t-1} \right)^+. \tag{6}$$

A few points on this policy are worth highlighting. First, given the backlog assignment matrix V, the quantity $I_i^{t-1} - \sum_k a_{ik} v_{kj(k)} B_{j(k)}^{t-1}$ represents the inventory position of resource i before ordering at the beginning of period t, and so the order quantity in (6) performs the usual task of bringing the inventory position up to the base-stock level S_i. Second, the order policy for i in (6) is independent of the order policy for other resources, an attractive feature for its ease of implementation in practice. Finally, we highlight upfront that the optimal assignment matrix V we derive in Section 4.1 is quite intuitive: It simply assigns each backlog to its lowest cost activity. But this result is dependent on the backlog fulfillment rule we specify next.

4.1.2. Backlog Fulfillment. Now that we have established a mapping of backlogs to resources, we consider how to fulfill the backlogs. Although there may be many fulfillment policies that could work well with the backlog assignment policy of Section 4.1.1, we consider a simple class of fulfillment policies that commit to fulfilling all backlogs using the activities in the assignment matrix V. In particular, we require the fulfillment vectors $\mathbf{x}^t$ to satisfy

$$x_k^t \geq v_{kj(k)} B_{j(k)}^{t-1}, \ \forall k. \tag{7}$$

Given a backlog assignment matrix V, we call the class of base-stock policies satisfying (6) and (7) an *assigned backlog base-stock* (ABBS) policy. This class of fulfillment policies provides a few important benefits. First, it guarantees that all backlogs are cleared as soon as possible, so that demand remains backlogged in the system for at most one period. This may be beneficial practically for managers seeking to minimize the waiting times of customers in the system. Second, it ensures that each backlog is fulfilled by the activity it was mapped to by V. Although this may not lead to the lowest possible cost in the current period (i.e., diverting the required resources away to other activities may give lower immediate cost), it makes for a straightforward policy that is easy to track in practice, and we show that it controls the cost well over the long run. In particular, we will show that within class of fulfillment policies (7) and base-stock policies (6), it is optimal to make myopic decisions; that is, these policies decouple the impact of the base-stock levels and fulfillment decisions across periods.

4.1.3. Optimizing the Base-Stock Policy. We now consider optimizing the base-stock policy by optimizing the fulfillment decisions, base-stock levels, and the backlog assignment matrix V. We first focus on optimizing the base-stock levels and fulfillment decisions given a fixed backlog assignment matrix V, assuming we are using the base-stock policy (6) and fulfillment decisions satisfying (7). To do so, recall that $\mathbf{S} = (S_1, \ldots, S_M)$ denotes the vector of base-stock levels, and let

$$g_j^V = \sum_k v_{kj} \left(q_k + \sum_i a_{ik} c_i \right),$$

denote the V weighted average of activity and resource costs for product j's backlog assignment. Note that under the base-stock policy (6) and a fulfillment policy satisfying (7), each unit of backlog for product j in period t will incur cost g_j^V in period $t+1$. Then consider the following linear program that minimizes the joint

fulfillment, holding and backlog costs for demand vector $\mathbf{D}$:

$$F^V(\mathbf{S}|\mathbf{D}) = \left\{ \begin{array}{ll} \min\limits_{\mathbf{x},\mathbf{y}\geq 0} & \sum\limits_k \left(q_k + \sum\limits_i a_{ik}(c_i - h_i) \right) x_k + \sum\limits_j (p_j + g_j^V) y_j \\ \text{s.t.} & \sum\limits_k a_{ik} x_k \leq S_i, \ \forall i \\ & \sum\limits_k r_{kj} x_k + y_j = D_j, \ \forall j \end{array} \right\}. \tag{8}$$

In (8), the variable x_k represents the processing level of activity k. The unit cost we charge to this variable is q_k, the activity processing cost, plus $\sum_i a_{ik} c_i$, the cost of ordering the resources required for activity k, minus $\sum_i a_{ik} h_i$, the savings from not holding these resources in inventory. The variable y_j represents the leftover backlog for product j, which is charged the backlogging cost p_j for the current period, plus the cost g_j^V for fulfilling this backlog in the next period. The constraints of (8) ensure that the activities consume no more than S_i units of resource i and that the total demand for each product equals the sum of its processing activities plus its backlog. Using this function F^V, we define the following stochastic program:

$$\underline{C}^V = \min_{\mathbf{S}\geq 0} \sum_i h_i S_i + \mathbb{E}[F^V(\mathbf{S}|\mathbf{D})]. \tag{9}$$

Let $\mathbf{S}^V$ denote a minimizer of (9) and $\mathbf{x}^V(\mathbf{D}), \mathbf{y}^V(\mathbf{D})$ denote a solution of $F^V(\mathbf{S}^V|\mathbf{D})$ for demand $\mathbf{D}$. Then, our ABBS policy uses base-stock levels $\mathbf{S}^V$ and the following fulfillment decision:

$$x_k^{t,V} = x_k^V(\mathbf{D}^t) + v_{kj(k)} B_{j(k)}^{t-1}, \ \forall k, t. \tag{10}$$

Our next result shows that this policy is optimal within the class of ABBS policies and has long-run average cost equal to the optimal objective value of the stochastic program (9).

Proposition 1. *Within the class of ABBS policies with backlog assignment matrix V, it is optimal to use base-stock levels $\mathbf{S}^V$ and fulfillment decisions $\mathbf{x}^{t,V}$. Moreover, the long-run average cost of this ABBS policy is $\underline{C}^V$.*

Given Proposition 1's characterization of the optimal long-run average cost in terms of the backlog assignment matrix V, we now turn our attention to optimizing over V. Fortunately, this task is straightforward after observing that the matrix V only impacts the stochastic program (9) via the objective coefficients g_j^V; that is, it does not enter into any constraints. Thus, to optimize over V we simply need to make each g_j^V as small as possible, which is achieved by mapping each product entirely to its least expensive fulfillment option. In particular, for each product j, let

$$\underline{k}(j) \in \arg\min_{k \in \mathcal{N}(j)} \left\{ q_k + \sum_i c_i a_{ik} \right\},$$

denote the (or an arbitrary selection in case of ties) lowest cost activity for serving its demand. Then define a matrix V^* with entries defined as follows:

$$v_{kj}^* = \begin{cases} 1 & k = \underline{k}(j), \\ 0 & \text{otherwise}, \end{cases}$$

so that V^* maps each product to its lowest cost activity.

Corollary 1. *Within the class of ABBS policies, it is optimal to use backlog assignment matrix V^*, base-stock levels $\mathbf{S}^{V^*}$, and fulfillment decisions $\mathbf{x}^{t,V^*}$, giving a long-run average cost of $\underline{C}^{V^*}$.*

Corollary 1 summarizes the advantages of the ABBS class of base-stock policies: (i) they are optimized by an intuitive backlog assignment rule that maps each product's backlogs to its lowest cost activity, (ii) the optimal base-stock levels and fulfillment decisions are determined by solving a single stochastic program, and (iii) the same stochastic program provides the long-run average cost of the optimal policy. These features will also allow us to prove a robust performance guarantee for this class of policies in the next section.

4.2. Bounding Performance of the Base-Stock Policy

In Section 4.1, we develop the ABBS class of base-stock policies and characterize the optimal policy within this class in Corollary 1. Such a result may be useful for managers seeking guidance on how to design and evaluate a good base-stock policy because these are often the default in practice. However, because the result assumes use of a base-stock policy, it does not reveal whether this is a good class of policies to use in the first place. Indeed, one would hope for a better result guaranteeing that the class of ABBS policies performs well relative to all feasible policies defined by (4).

In this section, we provide such a guarantee by bounding the performance of the ABBS policy designed in Section 4.1 relative to an optimal policy. To do so, we specialize our novel stochastic program lower bound from Section 3 to the zero lead time setting in Section 4.2.1 and then derive a performance guarantee by comparing stochastic program objectives in Section 4.2.2. Finally, in Section 4.2.3, we demonstrate via simulations that the performance of our ABBS policies is often much better than the worst case guarantee. As mentioned earlier in the introduction, we derive our best approximation guarantees under the following mild condition on the processing, holding, and resource cost parameters.

Condition 1. *For each processing activity k, the total cost per unit of processing the activity and procuring its required resources is larger than the total holding cost for its required resources per unit per period, that is, $q_k + \sum_i a_{ik}c_i \geq \sum_i a_{ik}h_i$ for all k.*

Condition 1 often holds in practice, as the standard method for determining annual holding costs is to multiply the resource cost by some percentage (Chopra 2018), with 25% often given as a rule of thumb, and typical ranges given are 10%–50% (Azzi et al. 2014). This holding cost percentage captures a variety of costs, including the cost of capital, and a variety of methods have been proposed for estimation, with general agreement that the final percentage should be less than about 50% (Berling and Rosling 2005, Berling 2008, Azzi et al. 2014, Odedairo et al. 2020).[3] We further note that this value represents the *annual* holding cost, which is typically scaled down to the model's period length, so for periods representing days or weeks holding costs are typically less than 0.1%–1.0% of the resource costs. This suggests that typically we have h_i significantly lower than c_i for each i, meaning Condition 1 is naturally satisfied for each k.

We also note that Condition 1 is not required for our analysis of optimal base-stock policies in Section 4.1 and is only used to prove our approximation guarantee in Section 4.2. Further, we extend this analysis to the general setting without Condition 1 in Section 4.3.

4.2.1. Specialized Stochastic Program Lower Bound. In this section, we specialize the family of lower bounds (5) into a more amenable form for comparison with the stochastic program (9) characterizing the performance of the ABBS policy. In particular, in (5), we choose $\delta = \rho = 0$ while leaving $\beta \in [0,1]$ to be chosen later. Then, define the following costs for each product j:

$$g_j^\beta = \min_{k \in \mathcal{N}(j)} \left\{ q_k + \sum_i a_{ik}(c_i - \beta h_i) \right\}, \qquad f_j^\beta = \min(p_j, g_j^\beta).$$

We observe with the definition of g_j^β, for $\beta = 0$, we have $g_j^0 = g_j^{V^*}$; that is, in this case, g_j^β captures the minimum cost of fulfillment for product j used in the optimal ABBS policy and its associated stochastic program (9). Henceforth we will use g_j^0 and $g_j^{V^*}$ interchangeably, given their equality.

Next, let $\mathcal{J} \subseteq \{1,\dots,N\}$ denote a subset of products (to be chosen later), and $\mathcal{K} = \cup_{j \in \mathcal{J}} \mathcal{N}(j)$ denote the set of activities serving these products. The idea behind this construction is that we want to focus on the subsystem composed of the subset $\mathcal{J}$ of products and their associated activities in $\mathcal{K}$; we will explain shortly how we choose such a subset. We also highlight that the set $\mathcal{K}$ depends on the choice of $\mathcal{J}$, but we suppress this dependence in our notation to keep the formulation concise. With this notation, choosing $\delta = \rho = 0$ the stochastic program (5) simplifies to the following for this subsystem (recalling we are considering the case $L = 0$).

$$F^\beta(\mathbf{S}|\mathbf{D},\mathcal{J}) = \left\{ \begin{array}{ll} \min\limits_{\mathbf{x},\mathbf{y}\geq 0} & \sum\limits_{k\in\mathcal{K}} \left(q_k + \sum\limits_i a_{ik}(c_i - h_i) \right) x_k + \sum\limits_{j\in\mathcal{J}} (p_j + f_j^\beta) y_j \\ \text{s.t.} & \sum\limits_{k\in\mathcal{K}} a_{ik}x_k \leq S_i, \quad \forall i \\ & \sum\limits_k r_{kj}x_k + y_j = D_j, \quad \forall j \in \mathcal{J} \end{array} \right\},$$

$$\underline{C}^\beta(\mathcal{J}) = \min_{\mathbf{S}\geq 0} \beta \sum_i h_i S_i + \mathbb{E}[F^\beta(\mathbf{S}|\mathbf{D},\mathcal{J})]. \tag{11}$$

Note that the stochastic program (11) is restricted to the products in the set $\mathcal{J}$. For the remaining products, we obtain a lower bound in terms of their minimum fulfillment cost g_j^0 times their average demand, leading to the following lower bound as a result of Theorem 1.

Corollary 2. *Under Condition 1, the stochastic program (11) for any $\beta \in [0,1]$ and any $\mathcal{J} \subseteq \{1,\dots,N\}$ gives the following lower bound on the long-run average cost of a feasible policy π: $\underline{C}^\beta(\mathcal{J}) + \sum_{j\notin\mathcal{J}} g_j^0 \mu_j \leq C(\pi)$.*

A few comments on Corollary 2 are in order. First, we highlight the tradeoff governed by the scaling factor β. If $\beta = 0$, then as noted above, we have $g_j^0 = g_j^{V^*}$ and the lower bound (11) captures the true cost of the feasible backlog fulfillment policy characterized in (9). However, if $\beta = 0$, then Program (11) does not capture the holding cost associated with the base-stock levels $\mathbf{S}$, and hence the program will be of little use. Hence, β governs a tradeoff between capturing the full holding cost and the full backlog fulfillment costs in the lower bound. This tradeoff stems from the accounting procedure used to prove the lower bound: β and $1-\beta$ represent weights placed on this and the next period's inventory decisions, respectively. We will exploit this tradeoff to prove our performance guarantee in the next section.

Second, we explain the role of the set $\mathcal{J}$ in the lower bound. For each product $j \notin \mathcal{J}$, the bound in Corollary 2 includes the cost $g_j^0\mu_j$, which represents the expected cost of meeting all of product j's demand at the lowest possible fulfillment cost. This is an intuitive lower bound because all demand is backlogged and so must be met at this cost or higher eventually. Thus, the bound in Corollary 2 uses this simple lower bound for some products (those not in $\mathcal{J}$) and solves a stochastic program for the rest (those in $\mathcal{J}$). This is helpful in proving our approximation guarantee because some products may have a backlog cost p_j that is small relative to the minimum fulfillment cost g_j^0. In this case, if this product were included in the stochastic program (11), it may end up with most of its demand unfulfilled in order to incur the lower cost p_j. Thus, for such products the simple lower bound $g_j^0\mu_j$ can actually end up being tighter than

the stochastic program solution. Indeed, in our analysis in the next section, we exclude from $\mathcal{J}$ those products whose minimum fulfillment costs g_j^0 are high relative to their backlog cost p_j.

4.2.2. Performance Bound for ABBS Policy. Now that we have the specialized stochastic program lower bound in (11), we are ready to compare these lower bounds with the upper bound stochastic program (9) in order to bound the performance of the ABBS policy developed in Section 4.1. As discussed in Section 4.2.1, for the stochastic program (11) we wish to choose a set of products $\mathcal{J}$ whose minimum fulfillment costs g_j^0 are not too high relative to their backlog costs p_j. Accordingly, for the chosen scaling factor β from Section 4.2.1, fix a subset of products $\mathcal{J}^\beta = \{j | \beta g_j^0 \le p_j\}$, and let $\mathbf{S}^\beta$ be an optimal solution to (11) for $\mathcal{J} = \mathcal{J}^\beta$. Then we show the following key lemma comparing the second stage linear programs of (9) and (11).

Lemma 1. *For any* $\beta \in [0.5, 1)$ *and* $\mathcal{J}^\beta = \{j | \beta g_j^0 \le p_j\}$, *under* Condition 1 *we have*

$$F^{V^*}(\mathbf{S}^\beta | \mathbf{D}) \le (1+\beta)\left(F^\beta(\mathbf{S}^\beta | \mathbf{D}, \mathcal{J}^\beta) + \sum_{j \notin \mathcal{J}^\beta} g_j^0 D_j\right).$$

Thus, the cost of the second stage linear program in (9) is bounded by a factor $1+\beta$ relative to the same quantity in the lower bound (11). Because it is also clear that the remaining cost in the upper bound (9) (i.e., the first stage holding cost) is bounded by a factor $1/\beta$ relative to the lower bound, these two observations can be combined to obtain an overall approximation guarantee.

Theorem 2. *Under Condition 1, the optimal ABBS policy provides an approximation factor of* $\frac{1+\sqrt{5}}{2} \approx 1.618$.

Thus, under Condition 1, the class of ABBS policies is guaranteed to be within a constant factor of an optimal policy. This is theoretically significant because it provides the first guarantee for base-stock policies in newsvendor networks, but also offers practical insight, because it provides a justification for using base-stock policies in more general network settings.

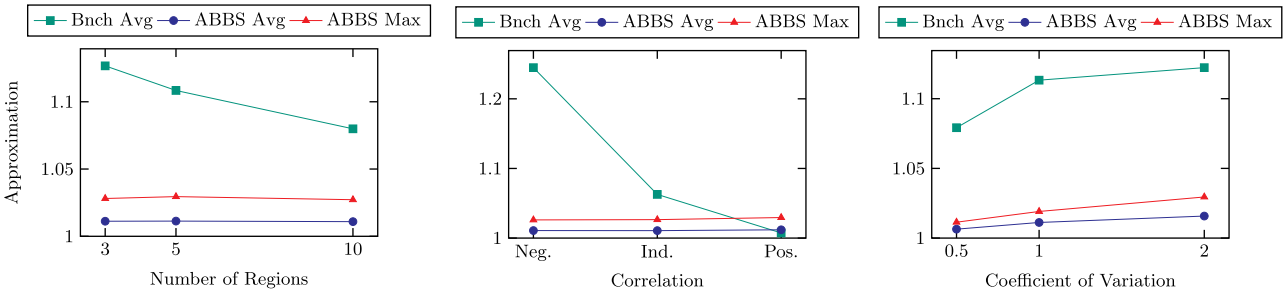
4.2.3. Simulations for ABBS Policies. In this section, we perform numerical simulations to test the effectiveness of ABBS policies in a variety of settings. First we test a simple fulfillment network (similar to the network depicted in Figure 1) to see how the ABBS policy performs as we vary problem parameters. Next we test several more complex networks with both fulfillment flexibility and resource commonality to demonstrate the robustness of our policy. Finally we present an example where the true optimal policy can be computed, and compare with our policy.

4.2.3.1. Simple Fulfillment Network. First, we consider a fulfillment network similar to Figure 1, with n demand regions and n warehouses (we consider $n \in \{3, 5, 10\}$). In particular, we assume the geographic centers of the demand regions are evenly spaced around the perimeter of a unit circle, and each region has a warehouse colocated at its geographic center. Each warehouse can fulfill demand originating in any region, and the unit cost of fulfillment, q_k, between any warehouse-region pair is equal to the Euclidean distance between their geographic centers, plus a small constant, 0.5, so that local fulfillment is not costless. For simplicity, in this section we consider only a single product (we consider more complex networks with multiple products in subsequent simulations). In each simulation we fix the ordering cost at $c_i = 1$ for each warehouse i, and the backlog cost at $p_j = 8$ for each region j. We set the holding cost using the standard industry practice of applying a holding cost percentage θ to the ordering cost, that is, $h_i = \theta c_i$. As discussed in Section 2, if periods represent days or weeks then 1% is reasonable for θ, which we test along with 10% and 25% for robustness. Finally, we consider multivariate normal demand distributions with nine different covariance matrices: three coefficients of variation (0.5, 1, and 2) combined with three correlations (negative, independent, and positive[4]). Each distribution has mean 10 for each region, and is truncated at 0. Thus, with three values of n, three holding cost percentages, and nine demand distributions, we test 81 instances in this simulation.

For each instance we compute the optimal ABBS policy parameters using Corollary 1, then simulate this policy over 100 periods for 1,000 sample paths and take the average cost over periods and each sample path. To benchmark our policy's performance, we also simulate a heuristic designed for such fulfillment networks by Govindarajan et al. (2021b). This policy solves a single newsvendor model to obtain the total network inventory level, then allocates inventory to warehouses according to a marginal cost heuristic, and uses a simple transportation LP to decide fulfillment (see Online Appendix E.2 for more details). We compare the performance of each policy to the best lower bound to obtain an approximation ratio for each instance. For the lower bound (11), we use the entire set of products for $\mathcal{J}$ and test a range of values for β from 0.5 to 1.

The policy performance is summarized in Figure 3 by number of regions, demand correlation, and coefficient of variation. The average and worst case approximation factors are reported for the ABBS policy, whereas the average approximation factor is reported for the benchmark. We see that the ABBS worst case outperforms the benchmark average for almost all parameters. Further, the performance of ABBS improves as the number of regions increases, and degrades slightly as demand becomes more correlated, or more variable. Overall, the

Figure 3. (Color online) Approximation Factors of Policies for Number of Regions, Correlation, and Coefficient of Variation



ABBS policy performance is much better than the worst case guarantee of Theorem 2, coming within 1% of optimal on average, and only 2.6% from optimal in the worst case instance.

4.2.3.2. E-Commerce and Production Networks.

In this section, we consider a more complex set of 108 newsvendor network problem instances constructed in DeValve (2023) that model a range of realistic production and e-commerce settings. Importantly, these networks include both fulfillment flexibility and resource commonality. The number of resources, products, and activities in the networks range from less than ten to hundreds, and several different combinations of demand distributions and cost parameters are considered (see DeValve (2023) for details). We translate the costs from DeValve (2023) to our dynamic setting as follows: the resource cost, c_i, and activity cost, q_k, are the same, what they call the "shortage" cost p_j we use as our backlog cost p_j, and we again use a holding cost percentage θ to set the holding cost as $h_i = \theta c_i$. For robustness, we test a range of eight values for θ from 1% to 75%, for a total of 864 problem instances.

We again simulate the optimal ABBS policy over 100 periods and 1,000 sample paths and compare with the best lower bound to obtain an approximation ratio for each instance. For the lower bound (11), we use the entire set of products for $\mathcal{J}$ and test a range of values for β from 0.5 to 1. The average and worst case approximation factors are reported by the problem instance type in Table 1, which shows the ABBS policy performs within about 0.8% of optimal on average and at worst 4.3% above optimal across these 864 problem instances, demonstrating its practical effectiveness. Further, Table 2 has the average and worst case performance broken out by the holding cost percentage, demonstrating stable performance across the range of holding costs considered.

Table 1. Average and Worst-Case Approximation Factors of ABBS Policy Across 864 Instances

Instance type	Average	Worst case
E-commerce	1.003	1.012
Production	1.014	1.043
Overall	1.008	1.043

4.2.3.3. Lower Bound on Worst-Case Approximation.

In this section, we provide an example to offer some intuition on when the ABBS policy may be suboptimal. To put this discussion in context, first observe that the simulations considered thus far compare the ABBS policy to the stochastic program lower bound; thus all approximation factors are upper bounds on the true performance relative to optimal and do not immediately imply a matching lower bound. To derive such a lower bound, we need to characterize the true optimal cost, which is challenging because of the complex, high-dimensional, and dynamic nature of the problem, even for moderately sized systems. This also makes it difficult to establish the tightness of the bound in Theorem 2 analytically. However, we are able to provide some intuition, by considering a small example where the optimal policy can be solved exactly with dynamic programming.

Example 1. Consider a fulfillment network with two products, two resources, and three activities, depicted in Figure 2. Product 1 can be fulfilled by resource 1, whereas product 2 can be fulfilled by either resource 1 or resource 2. Demand for each product follows an independent Bernoulli distribution with a success probability of 0.1. Product 1 has backlog cost $p_1 = 7$, whereas product 2 has $p_2 = 1$. Resource 1 has $c_1 = h_1 = 1.01$, whereas resource 2 has $c_2 = h_2 = 1$, and all activities have $q_k = 0$.

Table 2. Average and Worst-Case Approximation Factors Across 864 Instances by Holding Cost Percentage

Holding cost percentage	Average	Worst case
1%	1.010	1.034
3%	1.008	1.025
5%	1.008	1.024
10%	1.007	1.018
15%	1.006	1.017
25%	1.005	1.014
50%	1.005	1.020
75%	1.008	1.043

In this instance, the optimal ABBS policy sets base-stock levels of zero for each resource and maps all product 1 backlog to resource 1, and all product 2 backlog to resource 2, leading to a long-run average cost of 1.001 per period. In this small instance, the dynamic program can be solved exactly for an optimal cost of 0.936 (see Online Appendix E.1 for details); therefore, the ABBS policy has about a 1.069 approximation factor. Further, the optimal policy is not an ABBS policy; in each period, it maps the first unit of product 2 backlog to resource 1 and then maps any remaining backlog to resource 2; that is, the optimal policy orders as follows: $z_1^t = B_1^{t-1} + \min(1, B_2^{t-1})$, $z_2^t = (B_2^{t-1} - 1)^+$ and fulfills as much backlog as possible in each period, prioritizing product 1 fulfillment.

Example 1 illustrates that an optimal policy may require a state-dependent backlog mapping. A brief intuitive explanation is that resource 1, being the more flexible resource, is better suited to handle the first unit of product 2 backlog, because it can provide a higher combined service level to the two products together. However, for additional units of product 2 backlog, this pooling benefit is diminished (because product 1 will see at most one unit of new backlog in the next period), so these backlog units are better assigned to the cheaper resource 2. Although this optimal policy may be characterized in a straightforward way in such a simple network, we note that both the intuition and computation will be much more complex for even moderately sized networks.[5] Thus, our ABBS policy provides an attractive alternative for its simplicity, intuition, and performance guarantee.

4.3. Relaxing Condition 1

Here we extend our performance bound to the case when Condition 1 does not hold, that is, an activity's per-period holding costs may be larger than the sum of its processing and ordering costs. We highlight that relaxing this condition allows the processing and ordering costs to be zero, thus capturing classic inventory models that only include holding and backlog costs. In this case, we specialize our stochastic program (5) in a different way to obtain a useful lower bound on the optimal cost. In particular, here we let $\beta = \delta = 1$ while leaving $\rho \in [0,1]$ to be chosen later to obtain the following second-stage linear program:

$$F^\rho(\mathbf{S},\mathbf{B}|\mathbf{D}) = \left\{ \begin{array}{ll} \min\limits_{\mathbf{x},\mathbf{y}\geq 0} & \sum\limits_k \Big(q_k + \sum\limits_i a_{ik}(c_i - h_i)\Big) x_k + (1-\rho)\sum\limits_j p_j y_j \\ \text{s.t.} & \sum\limits_k a_{ik} x_k \leq S_i, \ \forall i \\ & \sum\limits_k r_{kj} x_k + y_j = B_j + D_j, \ \forall j \end{array} \right\}.$$

Then, with a slight abuse of notation, the stochastic program is

$$\underline{C}^\rho = \min_{\mathbf{S},\mathbf{B}\geq 0} \sum_i h_i S_i + \rho \sum_j p_j B_j + \mathbb{E}[F^\rho(\mathbf{S},\mathbf{B}|\mathbf{D})]. \quad (12)$$

We highlight that the scaling factor ρ governs a tradeoff between the cost of the initial backlog vector $\mathbf{B}$ and the final backlog $\mathbf{y}$ in the second-stage linear program. In this sense ρ plays a similar role to the factor β in the stochastic program (11), which governed a tradeoff between successive inventory decisions and indeed stems from the same accounting procedure for proving the lower bound: ρ and $1-\rho$ represent weights placed on this and the next period's backlog levels, respectively. This leads to the following lower bound.

Corollary 3. *The stochastic program (12) for any $\rho \in [0,1]$ gives the following lower bound on the long-run average cost of any feasible policy π: $\underline{C}^\rho \leq C(\pi)$.*

We use this lower bound to prove the approximation guarantee. Define the following quantities

$$\overline{h}_j = \max_{k\in\mathcal{N}(j)} \sum_i a_{ik} h_i \ \ \forall j, \quad \alpha = \min_j \frac{p_j}{p_j + \overline{h}_j}.$$

We highlight that for a classic single-product/single-resource newsvendor problem, α reduces to the classic critical fractile formula for the optimal service level. Similarly, in a newsvendor network we observe that, roughly speaking, higher backlog costs (relative to holding costs) make the stochastic program (9) implement higher service levels in an ABBS policy. Thus, in a sense we can view α as a rough indicator of the system's service level; that is, newsvendor networks that target high service levels for their products are likely to have α close to one. This motivates our next result, which bounds the approximation guarantee of the ABBS policy class with the inverse of α and therefore is useful for systems with high service level.

Theorem 3. *The optimal ABBS policy provides an approximation factor of $1 + \frac{1}{\alpha}$.*

With this result, we can summarize the main intuition of our results so far by saying that an ABBS policy performs well for newsvendor networks where the holding costs are not too large relative to the fulfillment costs (by Theorem 2) or the backlog costs (by Theorem 3). Thus, these policies may prove useful in many practical settings where quick fulfillment and service constraints take priority over inventory concerns.

We close this section with a brief simulation illustrating that the ABBS policy continues to perform well when Condition 1 does not hold. In particular, we simulate the ABBS policy on the same problem instances as described in Section 4.2.3 but set all activity and ordering costs to zero, that is, $q_k = 0 \ \forall k$ and $c_i = 0 \ \forall i$. We

compute an approximation factor by comparing to the best lower bound (12), tested over a few values of ρ. The results of this simulation are summarized in Table 3. Here we see that the ABBS policy continues to provide good performance, within less than 1% of optimal on average, and at worst about 9%. Thus, compared with Table 1, the numerical performance is only slightly worse without Condition 1.

5. Generalization to Lead Times

The analysis of base-stock policies in the preceding sections assumes that orders arrive in the period they are placed, which is practical in many supply chains with short delivery lead times, but may be impractical in others. Thus, to demonstrate the validity of our analysis in broader settings and to guide practitioners interested in implementation, in this section we extend our analysis to incorporate lead times for resource orders.

5.1. ABBS Policy for Lead Times

We now extend the base-stock policy developed in Section 4 to adapt to the resource ordering lead times. The extension follows the classic inventory management principle that the base-stock levels, which we again denote with a vector $\mathbf{S} \geq 0$, should govern each resource's *inventory position*, that is, the sum of on-hand and on-order inventory, minus backlogs. The key, as in Section 4, is to map backlogs to resources and fulfill demand in an intelligent way, but doing so with lead times requires a bit more care. This is because backlogs can no longer be cleared immediately in the following period with a new order but rather have to use inventory from an older arriving order or else wait for a new order to arrive. Demand fulfillment decisions only complicate matters, as they impact backlog levels in future periods, creating a complicated dynamic fulfillment problem. In fact, even fixing base-stock levels and a backlog mapping, with lead times the remaining fulfillment problem is a high dimensional dynamic program, which suffers from the curse of dimensionality as the lead time grows. Thus, in this section, we focus on identifying feasible base-stock policies rather than optimizing over a class of policies as we did in Section 4.1.

To do this, we split backlogs into two categories as we allocate them to resources: one category to be filled as soon as possible and the other category left to wait until replenishment arrives to be filled. In particular, we assign arriving product j demand on day t to its activities $k \in \mathcal{N}(j)$ according to values $\hat{D}_k^t$ and $\tilde{B}_k^t$ as follows:

$$D_j^t = \sum_k r_{kj}(\hat{D}_k^t + \tilde{B}_k^t), \ \forall j, \tag{13}$$

where $\hat{D}_k^t \geq 0$ denotes the demand that will be filled by activity k as soon as possible (i.e., "immediate fulfillment" demand), whereas $\tilde{B}_k^t \geq 0$ is resigned to backlog and will wait for the receipt of orders placed in period $t+1$ for activity k's resources before being fulfilled.

Using these allocation categories, we next generalize the ABBS fulfillment policy (10) to account for lead times. The main challenge in this generalization is that we can no longer directly implement a fulfillment policy based on a stochastic program solution (as in the first term of (10)), because, as we show below, the appropriate stochastic program to consider in this setting aggregates demands over a lead time, but the fulfillment decisions need to be made considering only the current period's demand and state variables. To address this challenge, we develop a general framework for each period's fulfillment decisions while noting that the framework allows a good deal of flexibility for designing the specifics of a given policy (of which we give two examples in the next section). The foundation of our policy framework is the construction of a random vector, $\tilde{\mathbf{X}}^t \in \mathbb{R}_+^K$, which intuitively represents the amount of "immediate fulfillment" demand filled in period t with each of the processing activities. Accordingly, we keep track of the following backlog process:

$$\hat{B}_k^t = \hat{B}_k^{t-1} + \hat{D}_k^t - \tilde{X}_k^t, \tag{14}$$

which tracks the backlog that has been assigned to activity k for immediate fulfillment (and assume $\hat{B}_k^0 = 0$ for all k). Then, to maintain a feasible policy, we enforce the following constraints:

$$\tilde{X}_k^t \leq \hat{D}_k^t + \hat{B}_k^{t-1}, \ \forall k, \tag{15}$$

$$\sum_k a_{ik}\tilde{X}_k^t \leq S_i - \sum_k a_{ik}\left(\sum_{s=t-L}^{t-1} \hat{D}_k^s - \hat{B}_k^{t-1}\right), \ \forall i. \tag{16}$$

Then, our policy framework uses the following generalization of (10) as the fulfillment rule:

$$x_k^t = \tilde{X}_k^t + \tilde{B}_k^{t-L-1}. \tag{17}$$

We also keep track of the following backlog process:

$$\overline{B}_k^t = \overline{B}_k^{t-1} + \hat{D}_k^t + \tilde{B}_k^t - x_k^t, \tag{18}$$

where $\overline{B}_k^t$ tracks the total backlog assigned to activity k (so that $B_j^t = \sum_k r_{kj}\overline{B}_k^t$), and assume this backlog starts empty, that is, $\overline{B}_k^0 = 0$ for all k. Then, by (14), (17), and (18), it also follows that $\overline{B}_k^t = \hat{B}_k^t + \sum_{s=t-L}^t \tilde{B}_k^s$. The base-

Table 3. Average and Worst-Case Approximation Factors of ABBS Policy Without Condition 1

Instance type	Average	Worst case
E-commerce	1.003	1.090
Production	1.002	1.038
Overall	1.003	1.090

stock ordering policy for resource i is then defined as

$$z_i^t = (S_i - I_i^{t-1} - \sum_{s=t-L}^{t-1} z_i^s + \sum_k a_{ik}\overline{B}_k^{t-1})^+, \tag{19}$$

where $I_i^{t-1} + \sum_{s=t-L}^{t} z_i^s - \sum_k a_{ik}\overline{B}_k^{t-1}$ represents the inventory position of resource i in period t after receiving order z_i^{t-L} and placing order z_i^t. We will refer to this as an ABBS policy, generalizing our definition from Section 4 to $L \geq 0$. We next show the policy is feasible and allows a simple characterization of the inventory, ordering, and backlog quantities in each period.

Lemma 2. *For nonanticipating random vectors $\tilde{\mathbf{X}}^t$, $\tilde{\mathbf{B}}^t$, and $\hat{\mathbf{D}}^t$ satisfying Constraints (13), (15), and (16), the ABBS policy using base-stock Policy (19) and fulfillment Policy (17) is feasible, and we have the following identities for all periods $t \geq 1$:*

$$I_i^t = S_i - \sum_k a_{ik}\left(\sum_{s=t-L}^{t} \hat{D}_k^s - \hat{B}_k^t\right) \forall i,$$

$$z_i^t = \sum_k a_{ik}(\hat{D}_k^{t-1} + \tilde{B}_k^{t-1}) \ \forall i, \quad \overline{B}_k^t = \hat{B}_k^t + \sum_{s=t-L}^{t} \tilde{B}_k^s \ \forall k.$$

Thus, Lemma 2 shows our policy framework characterizes all relevant quantities in the system in terms of the random vectors $\tilde{\mathbf{X}}^t$, $\tilde{\mathbf{B}}^t$, and $\hat{\mathbf{D}}^t$, and the base-stock levels $\mathbf{S}$. Therefore, within the class of ABBS policies, we can focus on finding good rules for deciding these vectors, which we explore further in the next section. We close this section by specializing the lower bound (5) to a simpler stochastic program focused on incorporating lead times. To do so, we first define the cost $u_k^L = Q_k - H_k$ for each activity k, and let $\tilde{\mathbf{D}}^\tau = \sum_{l=1}^{\tau} \mathbf{D}^l$ denote a cumulative demand vector over τ periods. Then in the lower bound (5), let $\beta = \delta = 1$ and $\rho = 0$ to obtain the stochastic program:

$$F^L(\mathbf{S}|\tilde{\mathbf{D}}^{L+1}) = \left\{ \begin{array}{ll} \min\limits_{\mathbf{x},\mathbf{y}\geq 0} & \sum\limits_k u_k^L x_k + \sum\limits_j p_j y_j \\ \text{s.t.} & \sum\limits_k a_{ik} x_k \leq S_i, \ \forall i \\ & \sum\limits_k r_{kj} x_k + y_j = \tilde{D}_j^{L+1}, \ \forall j \end{array} \right\},$$

$$\underline{C}^L = \min_{\mathbf{S}\geq 0} \sum_i h_i S_i + \mathbb{E}[F^L(\mathbf{S}|\tilde{\mathbf{D}}^{L+1})], \tag{20}$$

where the randomness is over the $L+1$ periods from when an order is placed (i.e., at the beginning of period $t-L$) to when the order can be used to fulfill demand (i.e., at the end of period t). Next, we extend Condition 1 to the setting with lead times.

Condition 2. *For each processing activity k, we have $u_k^L \geq 0$.*

Note that Condition 2 plays the same role as Condition 1 in that it guarantees the objective coefficients of the fulfillment decisions x_k in the stochastic program are nonnegative. From a practice perspective, we note that Condition 2 often holds for the same reasons discussed after Condition 1 in Section 4.2: The *annual* holding cost is typically estimated as some percentage of the resource cost and then scaled down by the number of periods in a year (e.g., 365 or 52 if a period represents a day or week, respectively), so in these cases, Condition 2 holds for lead times less than one year[6] and thus applies to many practical applications. We close with our lower bound.

Corollary 4. *Under Condition 2, the stochastic program (20) provides a lower bound on the long-run average cost of any policy, that is, $\underline{C}^L \leq C(\pi)$.*

5.2. Performance Bound for ABBS Policy with Lead Times

In this section, we illustrate the utility of our policy framework by constructing two feasible ABBS policies and proving associated performance bounds. As discussed in the prior section, our framework allows flexibility to define many different policies, and therefore the present construction and analysis should be viewed as only initial examples of the framework's potential.

5.2.1. Randomized Fulfillment. Our first policy will use a randomized fulfillment rule; that is, $\tilde{\mathbf{X}}^t$ will be determined (at least partially) by a draw from a random variable. To motivate the construction we observe that, over a lead time, we would like the cumulative $\tilde{\mathbf{X}}^t$ vectors to mimic the $\mathbf{x}$ vectors from the stochastic program (20). To do this, let $\mathbf{S}^L$ denote an optimal solution to (20), and let $\mathbf{x}(\tilde{\mathbf{D}}^{L+1})$ and $\mathbf{y}(\tilde{\mathbf{D}}^{L+1})$ denote an optimal solution to the LP defining $F^L(\mathbf{S}^L|\tilde{\mathbf{D}}^{L+1})$. Then we define the procedure for determining $\tilde{\mathbf{X}}^t$ in Algorithm 1. With this vector, we define the rest of the policy vectors as follows:

$$\hat{D}_k^t = \tilde{X}_k^t, \ \forall k, \tag{21}$$

$$\tilde{B}_k^t = v^*_{kj(k)}\left(D_j^t - \sum_k r_{kj}\tilde{X}_k^t\right), \ \forall k. \tag{22}$$

Thus, for this policy, all immediate fulfillment demand is cleared in each period, so that $\hat{B}_k^t = 0$ for all k, and all remaining backlog is assigned to the lowest cost activity using assignment matrix V^*.

Algorithm 1 (Fulfillment Procedure for Period t)

1: Input $\mathbf{S}, \mathbf{D}^t$, and $\tilde{\mathbf{X}}^s$ for $t-L \leq s \leq t-1$
2: Initialize $\tilde{X}_k^t = 0$ for all k
3: Draw an independent realization $\tilde{\mathbf{d}}^L$ of $\tilde{\mathbf{D}}^L$ and set

$$W_k^t = \frac{x_k(\mathbf{D}^t + \tilde{\mathbf{d}}^L)D_{j(k)}^t}{D_{j(k)}^t + \tilde{d}_{j(k)}^L}, \ \forall k$$

4: Iterate through all k and update:

$$\tilde{X}_k^t \leftarrow \min\left(W_k^t, \min_{i\in\mathcal{S}(k)}\left\{\frac{S_i - \sum_{s=t-L}^{t}\sum_{k'} a_{ik'}\tilde{X}_{k'}^s}{a_{ik}}\right\}\right), \quad \forall k.$$

We next provide a brief intuition for determining $\tilde{\mathbf{X}}^t$ in Algorithm 1. First, in Step 3, we randomly select a candidate value for X_k^t, which we denote W_k^t. To do so, we add the current period's demand vector $\mathbf{D}^t$ to a randomly drawn demand vector $\tilde{\mathbf{d}}^L$, which is drawn from the distribution of cumulative demand over L periods, $\tilde{\mathbf{D}}^L$. Thus, together $\mathbf{D}^t + \tilde{\mathbf{d}}^L$ has the same distribution as the $L+1$ period demand $\tilde{D}^{L+1}$ considered in the stochastic program lower bound (20). The purpose of this is to generate the stochastic program's optimal fulfillment vector $\mathbf{x}(\mathbf{D}^t + \tilde{\mathbf{d}}^L)$, which we then use to proportionally route each demand quantity D_j^t to its activities for fulfillment. In particular, W_k^t for each k is set equal to $D_{j(k)}^t$ times the fraction of demand that was served by activity k under the demand vector $\mathbf{D}^t + \tilde{\mathbf{d}}^L$ in the stochastic program. This strategy aims to match the expected fulfillment for each activity in the ABBS policy with its expected fulfillment from the stochastic program. Further, this procedure guarantees that W_k^t is less than $x_k(\mathbf{D}^t + \tilde{\mathbf{d}}^L)$, which aids in bounding the probability that the W_k^t variables are feasible. Finally, in Step 4 of Algorithm 1, we set the actual $\tilde{X}_k^t$ variables by adjusting the W_k^t variables to ensure they are feasible for Constraints (15) and (16) (which hold because $\hat{D}_k^t = \tilde{X}_k^t$ and $\hat{B}_k^t = 0$). With the $\tilde{\mathbf{X}}^t$ vectors defined in Algorithm 1, we are now ready to prove our performance guarantee. To state the result, let $m = \max_k\{|\mathcal{S}(k)|\}$ denote the maximum number of resources used by any activity in the system.

Theorem 4. *Under Condition 2, there exists an ABBS policy using the $\tilde{\mathbf{X}}^t$ vectors defined in Algorithm 1 and the allocation vectors in (21) and (22) with approximation factor $2(1+\sqrt{m\gamma(L+1)})$, where $\gamma = \frac{\sum_j p_j\mu_j}{\sum_j g_j^0\mu_j}$, is the demand weighted average ratio of backlog to minimum fulfillment costs.*

Thus, by Theorem 4, an ABBS policy performs well as long as the lead time and backlog costs are not too large, and each activity does not use too many resources. Next, we define a different ABBS policy using a deterministic fulfillment rule which has a performance guarantee independent of the lead time and network structure, depending only on the cost and demand parameters.

5.2.2. Proportional Fulfillment. We now construct a second policy, which differs from the previous one in a few ways. First, the fulfillment quantities are determined by a simple deterministic rule. Second, some demand assigned for immediate fulfillment may not be fulfilled in the period it was assigned, and we thus may have periods where the associated backlog process is positive, $\hat{B}_k^t > 0$. This is advantageous because it allows us to attempt fulfilling this backlog again in successive periods, without having to wait a lead time for replenishment, which allows us to reduce the expected backlog held in the system.

To construct the policy, recall that $\mathbf{x}(\tilde{\mathbf{D}}^{L+1})$ and $\mathbf{y}(\tilde{\mathbf{D}}^{L+1})$ denote an optimal solution to the LP defining $F^L(\mathbf{S}^L|\tilde{\mathbf{D}}^{L+1})$, and let $\bar{\mathbf{x}} = \mathbb{E}[\mathbf{x}(\tilde{\mathbf{D}}^{L+1})]$ and $\bar{\mathbf{y}} = \mathbb{E}[\mathbf{y}(\tilde{\mathbf{D}}^{L+1})]$ denote their expected values. Then, for each k let $\phi_k = \bar{x}_k/\mathbb{E}[\tilde{D}_{j(k)}^{L+1}] = \bar{x}_k/((L+1)\mu_{j(k)})$ denote the proportion of expected demand fulfilled by activity k in the stochastic program (20). Similarly, for each j, let $\psi_j = \bar{y}_j/\mathbb{E}[\tilde{D}_j^{L+1}] = \bar{y}_j/((L+1)\mu_j)$ denote the proportion of expected demand for product j left unfulfilled in the stochastic program (20). Define values $U_k \geq 0$ for each k that satisfy $\sum_k a_{ik}U_k \leq S_i$ for each i. Then define the policy as follows:

$$\hat{D}_k^t = \phi_k D_{j(k)}^t, \tag{23}$$

$$\tilde{B}_k^t = v_{kj(k)}^*\psi_{j(k)}D_{j(k)}^t, \tag{24}$$

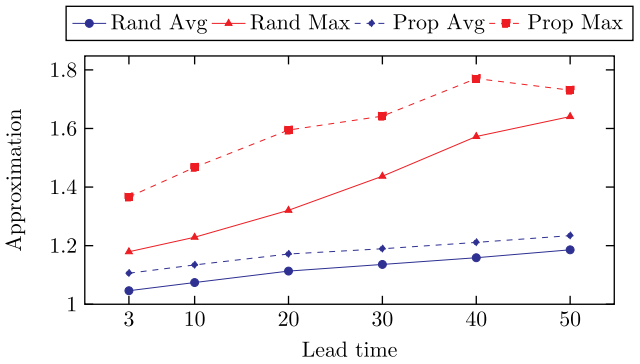
$$\tilde{X}_k^t = \min\left(\hat{D}_k^t + \hat{B}_k^{t-1}, U_k + \hat{B}_k^{t-1} - \sum_{s=t-L}^{t-1}\hat{D}_k^s\right). \tag{25}$$

Intuitively, this policy is very straightforward: It allocates all demand to activities in deterministic proportions following the stochastic program solution and allows up to U_k units of activity k over each lead time. It also deterministically leaves a proportion of demand as backlog until the next replenishment, according to the proportions from the stochastic program solution. In this way, the policy tries to mimic the stochastic program solution. Let $\sigma_j^2 = \mathrm{Var}(D_j)$ denote the variance of demand for product j.

Theorem 5. *Under Condition 2, there exists an ABBS policy using (23), (24), and (25) with approximation factor $2\left(1+\sqrt{\frac{\sum_j(p_j+\bar{h}_j)\mu_j\nu_j^2}{\sum_j g_j^0\mu_j}}\right)$, where $\nu_j = \sigma_j/\mu_j$ denotes the coefficient of variation for product j demand.*

Thus, the approximation guarantee is independent of the lead time and network structure and depends only on the cost and demand parameters of the system. To help interpret the bound, we note that if the coefficients of variation of demand were the same across all products, that is, $\nu_j = \nu$ for all j, and if the cost ratio $(p_j + \bar{h}_j)/g_j^0 = (p+h)/g$ were also constant, then the approximation factor becomes $2\left(1+\nu\sqrt{\frac{p+h}{g}}\right)$, so it can be roughly thought of as scaling with the average coefficient of variation and the square root of the average ratio of backlogging and holding costs to fulfillment costs. Finally, comparing the bounds in Theorems 4 and 5, they suggest the randomized policy may be better if the lead time and maximum activity size are small, whereas the proportional policy may be better if the coefficient of variation is small.

Figure 4. (Color online) Approximation Factors of ABBS Policies as Lead Time Grows



5.3. Simulations

In this section, we perform a final simulation experiment to test the performance of our ABBS policies for lead times. In particular, we simulate the same 108 e-commerce and production networks from Section 4.2.3 with lead times of $L \in \{3, 10, 20, 40, 50\}$. For each instance, we use a holding cost percentage of 1%. We compute approximation ratios in the same way as Section 4.2.3, that is, the simulation average cost divided by the best lower bound. The results are plotted in Figure 4, where we observe the approximation factors are all significantly better than the worst case guarantees of Theorems 4 and 5 (which are both at least two). Further, we observe that the randomized policy tends to outperform the proportional policy, which we attribute to the randomized policy's adaptation to the demand realizations, as opposed to the proportional policy's deterministic fulfillment ratios. We explore these issues with further simulations in Online Appendix E.3.

We close this section by highlighting that we view Theorems 4 and 5 and the analysis in this section as a proof of concept illustrating the ability to prove performance guarantees using our ABBS policy framework with lead times. We expect the framework to allow improved analyses and guarantees for these and other policies within this class.

6. Conclusion

We developed a new class of base-stock policies for newsvendor networks that are straightforward to implement and analyze. Using a novel stochastic programming formulation, we established that this class of base-stock policies is guaranteed to perform well relative to optimal, thus establishing the first approximation algorithm for general newsvendor networks. The policies are also practical in the sense that they significantly outperform the worst case guarantee in our numerical simulations. Further, our stochastic programming methodology for bounding the cost of an optimal policy is robust and may be applied to other problems, which we illustrate through extensions to more general cost and lead time conditions.

Endnotes

[1] Although resource-specific lead times, L_i, are practically relevant, they significantly complicate the model, creating a generalization of the notoriously challenging dual-sourcing problem (Xin and Goldberg 2018), and hence are out of scope for this paper. See Section E.4 for further details.

[2] More formally, z_i^t and x_k^t, should be measurable with respect to the sigma algebras generated by these quantities.

[3] Also, even in cases where holding costs may be higher, due for example to high cost of obsolescence (as with high tech items), suppliers often offer retailers holding-cost subsidies (Chopra 2018), effectively offsetting their impact in the model.

[4] Positive correlation coefficient is 0.5, independent is 0, and negative is $-1/(n-1)$ to maintain valid covariance matrix.

[5] The dynamic program for this four node network has 548 state-action pairs, even after removing redundant actions and unreachable states.

[6] If there are Δ periods in a year and the company's annual holding cost percentage is $\theta \le 1$, the per period holding cost rate is typically estimated as $h_i = \theta c_i/\Delta$. Thus, if $L < \Delta$, that is, the lead time is less than the number of periods in a year, then we have $L + 1 \le \Delta \le \Delta/\theta$, which implies $c_i/(L+1) \ge h_i \ \forall i$, and Condition 2 holds.

References

Acimovic J, Graves SC (2015) Making better fulfillment decisions on the fly in an online retail environment. *Manufacturing Service Oper. Management* 17(1):34–51.

Acimovic J, Graves SC (2017) Mitigating spillover in online retailing via replenishment. *Manufacturing Service Oper. Management* 19(3):419–436.

Akturk D (2022) Managing inventory in a network: Performance bounds for simple policies. *Oper. Res. Lett.* 50(3):315–321.

Akturk D, Candogan O, Gupta V (2024) Managing resources for shared micromobility: Approximate optimality in large-scale systems. Preprint, submitted July 19, https://dx.doi.org/10.2139/ssrn.4155841.

Amil A, Makhdoumi A, Wei Y (2023) Multi-item order fulfillment revisited: LP formulation and prophet inequality. Preprint, submitted August 4, https://dx.doi.org/10.2139/ssrn.4176274.

Azzi A, Battini D, Faccio M, Persona A, Sgarbossa F (2014) Inventory holding costs measurement: A multi-case study. *Internat. J. Logist. Management* 25(1):109–132.

Bassamboo A, Randhawa RS, Van Mieghem JA (2010) Optimal flexibility configurations in newsvendor networks: Going beyond chaining and pairing. *Management Sci.* 56(8):1285–1303.

Berling P (2008) Holding cost determination: An activity-based cost approach. *Internat. J. Production Econom.* 112(2):829–840.

Berling P, Rosling K (2005) The effects of financial risks on inventory policy. *Management Sci.* 51(12):1804–1815.

Birge JR, DeValve L (2024) Inventory placement on a network. Preprint, submitted April 30, https://dx.doi.org/10.2139/ssrn.4808999.

Chao X, Gong X, Shi C, Zhang H (2015) Approximation algorithms for perishable inventory systems. *Oper. Res.* 63(3):585–601.

Chao X, Gong X, Shi C, Yang C, Zhang H, Zhou SX (2018) Approximation algorithms for capacitated perishable inventory systems with positive lead times. *Management Sci.* 64(11):5038–5061.

Chen X, Sun P (2012) Optimal structural policies for ambiguity and risk averse inventory and pricing models. *SIAM J. Control Optim.* 50(1):133–146.

Chen B, Chao X, Ahn H-S (2019) Coordinating pricing and inventory replenishment with nonparametric demand learning. *Oper. Res.* 67(4):1035–1052.

Chen X, Gao X, Hu Z (2015) A new approach to two-location joint inventory and transshipment control via $L^\natural$-convexity. *Oper. Res. Lett.* 43(1):65–68.

Chen X, Hu P, Shum S, Zhang Y (2016) Dynamic stochastic inventory management with reference price effects. *Oper. Res.* 64(6): 1529–1536.

Chen S, Lu L, Song J-SJ, Zhang H (2021) Optimizing assemble-to-order systems: Decomposition heuristics and scalable algorithms. Research Paper 2021-33, HKUST Business School, Hong Kong.

Cheung WC, Ma W, Simchi-Levi D, Wang X (2022) Inventory balancing with online learning. *Management Sci.* 68(3):1776–1807.

Chopra S (2018) *Supply Chain Management: Strategy, Planning, and Operation*, 7th ed. (Pearson, London).

DeValve L (2023) Cost balancing for general inventory/fulfillment networks with applications to ATO and multi-item e-retail problems. Preprint, submitted November 14, https://dx.doi.org/10.2139/ssrn.3961613.

DeValve L, Pekec S, Wei Y (2020) A primal-dual approach to analyzing ATO systems. *Management Sci.* 66(11):5389–5407.

DeValve L, Song J-SJ, Wei Y (2023a) Assemble-to-order systems. *Research Handbook on Inventory Management* (Edward Elgar Publishing, Cheltenham), 191–212.

DeValve L, Wei Y, Di Wu RY (2023b) Understanding the value of fulfillment flexibility in an online retailing environment. *Manufacturing Service Oper. Management* 25(2):391–408.

Doğru MK, Reiman MI, Wang Q (2010) A stochastic programming based inventory policy for assemble-to-order systems with application to the W model. *Oper. Res.* 58(4-part-1):849–864.

Doğru MK, Reiman MI, Wang Q (2017) Assemble-to-order inventory management via stochastic programming: Chained BOMs and the M-system. *Production Oper. Management* 26(3):446–468.

Feng Q, Sethi SP, Yan H, Zhang H (2006) Are base-stock policies optimal in inventory problems with multiple delivery modes? *Oper. Res.* 54(4):801–807.

Govindarajan A, Sinha A, Uichanco J (2021a) Distribution-free inventory risk pooling in a multilocation newsvendor. *Management Sci.* 67(4):2272–2291.

Govindarajan A, Sinha A, Uichanco J (2021b) Joint inventory and fulfillment decisions for omnichannel retail networks. *Naval Res. Logist.* 68(6):779–794.

Graves SC, Schoenmeyr T (2016) Strategic safety-stock placement in supply chains with capacity constraints. *Manufacturing Service Oper. Management* 18(3):445–460.

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

Graves SC, Willems SP (2008) Strategic inventory placement in supply chains: Nonstationary demand. *Manufacturing Service Oper. Management* 10(2):278–287.

Harrison JM, Van Mieghem JA (1999) Multi-resource investment strategies: Operational hedging under demand uncertainty. *Eur. J. Oper. Res.* 113(1):17–29.

Hu X, Duenyas I, Kapuscinski R (2008) Optimal joint inventory and transshipment control under uncertain capacity. *Oper. Res.* 56(4):881–897.

Jasin S, Sinha A (2015) An LP-based correlated rounding scheme for multi-item ecommerce order fulfillment. *Oper. Res.* 63(6): 1336–1351.

Jiang H, Jiang S, Shen Z-JM (2022) Learning while repositioning in on-demand vehicle sharing networks. Preprint, submitted June 26, https://dx.doi.org/10.2139/ssrn.4140449.

Jiang J, Wang S, Zhang J (2023) Achieving high individual service-levels without safety stock? Optimal rationing policy of pooled resources. *Oper. Res.* 71(1):358–377.

Jordan WC, Graves SC (1995) Principles on the benefits of manufacturing process flexibility. *Management Sci.* 41(4):577–594.

Kranenburg AA, Van Houtum GJ (2009) A new partial pooling structure for spare parts networks. *Eur. J. Oper. Res.* 199(3):908–921.

Levi R, Janakiraman G, Nagarajan M (2008a) A 2-approximation algorithm for stochastic inventory control models with lost sales. *Math. Oper. Res.* 33(2):351–374.

Levi R, Pál M, Roundy RO, Shmoys DB (2007) Approximation algorithms for stochastic inventory control models. *Math. Oper. Res.* 32(2):284–302.

Levi R, Roundy RO, Shmoys DB, Truong VA (2008b) Approximation algorithms for capacitated stochastic inventory control models. *Oper. Res.* 56(5):1184–1199.

Lu Y, Song J-S (2005) Order-based cost optimization in assemble-to-order systems. *Oper. Res.* 53(1):151–169.

Lu L, Song J-S, Zhang H (2015) Optimal and asymptotically optimal policies for assemble-to-order N- and W-systems. *Naval Res. Logist.* 62(8):617–645.

Lu Y, Song J-S, Zhao Y (2010) No-holdback allocation rules for continuous-time assemble-to-order systems. *Oper. Res.* 58(3):691–705.

Ma W (2023) Order-optimal correlated rounding for fulfilling multi-item e-commerce orders. *Manufacturing Service Oper. Management* 25(4):1209–1621.

Odedairo BO, Alaba EH, Edem I (2020) A system dynamics model to determine the value of inventory holding cost. *J. Engrg. Stud. Res.* 26(3):112–123.

Qin H, Simchi-Levi D, Ferer R, Mays J, Merriam K, Forrester M, Hamrick A (2022) Trading safety stock for service response time in inventory positioning. *Production Oper. Management* 31(12):4462–4474.

Reiman MI, Wang Q (2015) Asymptotically optimal inventory control for assemble-to-order systems with identical lead times. *Oper. Res.* 63(3):716–732.

Scarf H (1959) The optimality of (S,s) policies in the dynamic inventory problem. *Mathematical Methods in the Social Sciences* (Stanford University Press, Stanford, CA), 196–202.

Schoenmeyr T, Graves SC (2009) Strategic safety stocks in supply chains with evolving forecasts. *Manufacturing Service Oper. Management* 11(4):657–673.

Song J-S, Xue Z (2021) Demand shaping through bundling and product configuration: A dynamic multiproduct inventory-pricing model. *Oper. Res.* 69(2):525–544.

Tomlin B, Wang Y (2005) On the value of mix flexibility and dual sourcing in unreliable newsvendor networks. *Manufacturing Service Oper. Management* 7(1):37–57.

Tomlin B, Wang Y (2008) Pricing and operational recourse in coproduction systems. *Management Sci.* 54(3):522–537.

Van Mieghem JA (1998) Investment strategies for flexible resources. *Management Sci.* 44(8):1071–1078.

Van Mieghem JA (2004) Commonality strategies: Value drivers and equivalence with flexible capacity and inventory substitution. *Management Sci.* 50(3):419–424.

Van Mieghem JA (2007) Risk mitigation in newsvendor networks: Resource diversification, flexibility, sharing, and hedging. *Management Sci.* 53(8):1269–1288.

Van Mieghem JA, Rudi N (2002) Newsvendor networks: Inventory management and capacity investment with discretionary activities. *Manufacturing Service Oper. Management* 4(4):313–335.

Xin L, Goldberg DA (2018) Asymptotic optimality of tailored base-surge policies in dual-sourcing inventory systems. *Management Sci.* 64(1):437–452.

Zhang C, Ayer T, White CC III (2023) Truncated balancing policy for perishable inventory management: Combating high shortage penalties. *Manufacturing Service Oper. Management* 25(6):2352–2370.

Zhang H, Shi C, Chao X (2016) Approximation algorithms for perishable inventory systems with setup costs. *Oper. Res.* 64(2):432–440.

Zhao Y, Birge JR, DeValve L, Inman R (2023) Managing multi-tier inventory networks with expediting under normal and disrupted modes. Preprint, submitted October 3, https://dx.doi.org/10.2139/ssrn.4204008.