CORC Report TR-2005-09

Computing robust basestock levels

Daniel Bienstock and Nuri Özbay
Columbia University
New York, NY 10027

November 2005 (version 1-19-2006)

Abstract

This paper considers how to optimally set the basestock level for a single buffer when demand is uncertain, in a robust framework. We present a family of algorithms based on decomposition that scale well to problems with hundreds of time periods, and theoretical results on more general models.

1 Introduction

In this paper we develop procedures for setting the basestock levels for a buffer in a supply chain subject to uncertainty in the demands. Our work is motivated by experience with an industrial partner in the electronics industry who was subject to the following difficulties: short product cycles, a complex supply chain with multiple suppliers and long production leadtimes, and a very competitive environment. The combination of these factors produced a paucity of demand data and a significant exposure to risk, in the form of either excessive inventory or shortages.

We consider a buffer evolving over a finite time horizon. For $t = 1, 2, \ldots, T$, the quantity x_t denotes the inventory at the start of period t (possibly negative to indicate a shortage) with x_1 given. We also have a (per unit) *inventory holding cost* h_t, a *backlogging cost* b_t, and a production cost c_t. The dynamics during period t work out as follows:

(a) First, one *orders* (produces, etc) a quantity $u_t \geq 0$, thereby increasing inventory to $x_t + u_t$, and incurring a cost $c_t u_t$,

(b) Next, the demand $d_t \geq 0$ at time t is realized, decreasing inventory to $x_{t+1} \doteq x_t + u_t - d_t$,

(c) Finally, at the end of period t, we pay a cost of $\max\{h_t x_{t+1}, -b_t x_{t+1}\}$.

This model can be extended in a number of ways, for example by considering capacities, setup costs, or termination conditions. These features can easily be added to the algorithms described in this paper.

We are interested in operating the buffer so that the sum of all costs incurred between time 1 and T is minimized. In order to devise a strategy to this effect, we need to make precise steps (a) and (b). In what follows, we will refer to the minimum-cost problem as the "basic inventory problem".

We consider (b) first. A large amount of supply-chain literature considers the case where demands are stochastically distributed with known distributions – this assumption has produced an abundance of significant and useful results. On the other hand the assumption that the demand distribution is known is nontrivial. In recent years, a growing body of literature has considered optimization problems where some of the input data is uncertain with an unknown distribution – in such a setting, we want to make decisions that are robust with regards to deviations of the data away from nominal (expected) values. One may think of the data as being picked by an adversary with limited power.

In general, we are given a set $\mathcal{D}$ (the *uncertainty set*). Each element of $\mathcal{D}$ is a vector $(d_1, d_2, \ldots, d_T)$ of demands that is available to the adversary. At time t, having previously chosen demand values $\hat{d}_i$ $(1 \leq i \leq t-1)$, the adversary can choose any demand value $\hat{d}_t$ such that there is some vector $(\hat{d}_1, \ldots, \hat{d}_{t-1}, \hat{d}_t, d_{t+1}, \ldots, d_T) \in \mathcal{D}$.

Given an uncertainty set $\mathcal{D}$, we need a strategy to produce orders u_t so as to minimize the maximum cost that can arise from demands in $\mathcal{D}$. To make this statement precise, we need to specify how (a) is implemented. In other words, we need to describe an algorithm, such that at each time t the decision maker observes the current state of the system (e.g. the current inventory x_t) and possibly prior actions on the part of the adversary, and chooses u_t appropriately. A classical approach found in the supply-chain literature is that of using a *basestock* policy. A *basestock* is a value $\sigma \geq 0$, such that at time t we set

$$u_t \quad = \quad \max\{\sigma - x_t, 0\}, \tag{1}$$

i.e. we order "up to" level σ.

The main focus of this paper concerns how to pick optimal basestock policies in the robust setting, under various demand uncertainty sets $\mathcal{D}$. Our focus is motivated, primarily, by the fact that the mechanism described by (1) has acquired very wide use. It can be shown to be optimal under many inventory models. See, [FZ84], [CS60], [I63a, I63b], [V66], [E84] [MT01], [Z00]. Further, even if such a policy may not be optimal, it is viewed as producing easily implementable policies in the broader context of a "real-world" supply chain, where it is necessary to deal with a number of complex details (such as the logistics of relationships with clients and suppliers) not easily handled by a mathematical optimization engine. In the concrete example of our industrial partner, we stress that using a (constant) basestock policy was an *operational constraint*.

The inventory problem in the robust setting, using a constant (time-independent) basestock, can be described as follows:

$$\min_{\sigma \geq 0} \; V(\sigma) \tag{2}$$

where for $\sigma \geq 0$,

$$V(\sigma) \quad = \quad \max_{d,x,u} \sum_{t=1}^{T} \left(c_t u_t \, + \, \max\{ \, h_t x_{t+1} \, , \; -b_t x_{t+1} \} \right) \tag{3}$$

s.t.

$$u_t \, = \, \max\{\sigma - x_t \, , \, 0\}, \quad 1 \leq t \leq T, \tag{4}$$

$$x_{t+1} \, = \, x_t + u_t - d_t, \quad 1 \leq t \leq T, \tag{5}$$

$$(d_1, d_2, \ldots, d_T) \, \in \, \mathcal{D}. \tag{6}$$

Here, (3)-(5) is the adversarial problem – once the demand variables $(d_1, d_2, \ldots, d_T) \in \mathcal{D}$ have been chosen, constraints (4)-(5) uniquely determine all other variables. Note that the quantity x_1 (the initial inventory level) is an input. Also, because of the "max" in (3) and (4), the adversarial problem is non-convex.

Note that we assume $\sigma \geq 0$ in (2) – in fact, our algorithms do not require this assumption. Under special conditions, the optimal basestock might be negative; however, we expect that the nonnegativity assumption would be commonly used and hence we state it explicitly.

Problem (2) posits a constant basestock over the entire planning horizon. However, we would expect that in practice (2) would be periodically reviewed (re-optimized) to adjust the basestock in a *rolling horizon* fashion, though perhaps not at every time period. The stipulation for a constant basestock in (2) can be viewed as an operational feature aimed at achieving stability (and "implementability") of the policy used to operate the supply chain. Clearly such a policy could prove suboptimal. However, when used under periodic review, and with an appropriate discounting function and termination conditions, the policy should still prove sufficiently flexible. In the case

of our industrial partner, the use of a constant basestock level was a required feature. In the face of large, and difficult to quantify, demand uncertainty, the use of a constant basestock was seen as endowing the supply chain with a measure of predictability and stability.

At the other extreme one could ask for a time-dependent basestock policy, i.e. we might have a different basestock value σ_t for each $1 \leq t \leq T$. We give a result regarding the adversarial problem in this general setting. Also, there are intermediate models between the two extremes of using different basestocks at each time interval and a constant basestock: for example,we might allow the basestock to change at the midpoint of the planning horizon. Or, with seasonal data, we might use a fixed basestock value for each "season". Even though we do not study such models in this paper, simple extensions of the algorithms we present can handle them.

A different model is that of *safety stocks*. A safety stock policy with margin λ is one that uses, at time t a basestock policy with $\sigma_t = \hat{\mu}^t + \lambda\hat{\delta}^t$, where $\hat{\mu}^t$ and $\hat{\delta}^t$, are given constants (typically, estimates of the mean demand and standard deviation of demand at time t, but not necessarily so, and in our study, arbitrary constants). We also present some results concerning this model.

1.1 Prior work

To the best of our knowledge, the first work on distribution-free supply chain management problems is due to Scarf [S58], who considered a single period newsvendor problem and determined the orders that maximize the minimum expected profit over all possible demand distributions, for a given first and second moments. Later, Gallego and Moon [GM93, MG94] provided concise derivations of his results and extended them to other cases. Gallego, Ryan and Simchi-Levi [GRS01] considered multi-period version of this problem with discrete demand distributions and proved the optimality of basestock policies. Recently, Bertsimas and Thiele [BT05] and Ben-Tal et.al. [BGNV05] studied some supply chain management problems with limited demand information using the robust optimization framework. A central difference between their work and previous work is that instead of assuming partial information about the demand distribution, they use the robust optimization framework outlined before. Also see [BGGN04] and [T05].

Robust Optimization addresses parameter uncertainties in optimization problems. Unlike Stochastic Programming it does not assume that the uncertain parameters are random variables with known distributions; rather it models uncertainty in parameters using deterministic uncertainty sets in which all possible values of these parameters reside. Robust Optimization, in principle, employs a min-max approach that guarantees the feasibility of the obtained solution for all possible values of the uncertain parameters in the designated uncertainty set.

Although the idea is older, the classical references for Robust Optimization are Ben-Tal and Nemirovski [BN98, BN99, BN00], where they studied a group of convex optimization problems with uncertain parameters and showed that they can be formulated as conic programs which can be solved in polynomial time. Since then, there has been an abundance of research that deals with various aspects of robust optimization. Among the most significant contributors is Bertsimas and Sim [BS03] who proposes a new polyhedral uncertainty set that guarantees feasibility with high probability for general distributions for the uncertain parameters. They show that Linear Programs with this uncertainty framework can be reformulated as Linear Programs with a small number of additional variables. Also see [AZ05], where robustness is introduced in the context of a combinatorial optimization problem.

Another field that deals with uncertainty in optimization problems is *Adversarial Queueing*, which was first considered by Borodin et. al [BKRSW96]. They studied packet routing over queuing networks when there is only limited information about demand. Similar to Robust Optimization, they adapted a worst case approach and proved some stability results that holds for all realizations of the demand. They used a demand model that was first introduced by Cruz [C91] to capture the burstiness of inputs in communication networks. Later, Andrews et. al. [AAFKLL96] considered a similar problem with greedy protocols.

In recent work, Bertsimas and Thiele [BT05] studied robust supply chain optimization problems. One particular contribution lies in how they model the demand uncertainty set $\mathcal{D}$. In their model

there are, for each time period t, numbers $0 \leq \delta_t \leq \mu_t$ and Γ_t, such that $0 \leq \Gamma_1 \leq \Gamma_2 \leq \ldots \leq \Gamma_T$ and $\Gamma_t \leq \Gamma_{t-1} + 1$ (for $1 < t \leq T$). A vector of demands d is in $\mathcal{D}$ if and only if there exist numbers $z_1, z_2, \ldots, z_T$, such that for $1 \leq t \leq T$,

$$\begin{aligned} d_t &= \mu_t + \delta_t z_t, && (7)\\ z_t &\in [-1, 1], && (8)\\ \sum_{j=1}^{t} |z_j| &\leq \Gamma_t. && (9) \end{aligned}$$

Here, the quantity μ_j is the "mean" or "nominal" demand at time j, and the model allows for an absolute deviation of up to δ_j units away from the mean. Constraints (9) constitute non-trivial requirements on the ensemble of all deviations. The method in [BT05] handles startup costs and production capacities, but it is assumed that costs are *stationary*, e.g. there are constants h, b and c such that $h_t = h$, $b_t = b$, and $c_t = c$ for all t. If we extend the model in [BT05] to the general case, the approach used in [BT05] formulates our basic inventory problem as the following linear program:

$$\begin{aligned} C^* \quad = \quad \min \quad & \sum_{t=1}^{T} (c_t u_t + y_t) && (10)\\ \text{s.t.} \quad & && (11)\\ & y_t \geq h_t \left(x_1 + \sum_{j=1}^{t} (u_j - \mu_j) + A_t \right) t = 1, \ldots T, && (12)\\ & y_t \geq b_t \left(-x_1 + \sum_{j=1}^{t} (\mu_j - u_j) + A_t \right) t = 1, \ldots T, && (13)\\ & u \geq 0, \end{aligned}$$

where for $t = 1, \ldots T$,

$$\begin{aligned} A_t = \max \quad & \sum_{j=1}^{t} \delta_j \epsilon_j^t && (14)\\ \text{s.t.} \quad & \sum_{j=1}^{t} \epsilon_j^t \leq \Gamma_t, \\ & 0 \leq \epsilon_j^t \leq 1, \quad 1 \leq j \leq t. \end{aligned}$$

Thus, LP (14) computes the *maximum* cumulative deviation away from the mean demands, by time t, that model (7)-(9) allows. If we denote by $\hat{\epsilon}_j^t$ $(1 \leq j \leq t)$ the optimal solution to LP (14), then constraint (12) yields the inventory holding cost that would be incurred at time t if the demands at time $1, 2, \ldots, t$ took values

$$\mu_1 - \delta_1 \hat{\epsilon}_1^t \, , \, \mu_2 - \delta_2 \hat{\epsilon}_2^t \, , \, \ldots \, , \, \mu_t - \delta_t \hat{\epsilon}_t^t,$$

whereas constraint (13) yields the backlogging cost that would be incurred at time t if the demands at time $1, 2, \ldots, t$ took values

$$\mu_1 + \delta_1 \hat{\epsilon}_1^t \, , \, \mu_2 + \hat{\delta}_2 \epsilon_2^t \, , \, \ldots \, , \, \mu_t + \delta_t \hat{\epsilon}_t^t,$$

e.g. in each case the deviations maximize the respective cost (see the discussion following equation (13) in [BT05]). Also, note that when computing A_t and $A_{t'}$ for $t \neq t'$ we will in general obtain different implied demands, e.g. $\hat{\epsilon}_i^t \neq \hat{\epsilon}_i^{t'}$ for $i \leq t, t'$.

Linear program (11) should be contrasted with the "true" min-max problem:

$$R^* \quad = \quad \min_{u \geq 0} \; R(u) \tag{15}$$

where for $u = (u_1, u_2, \ldots, u_T) \geq 0$,

$$\begin{aligned} R(u) \quad &= \quad \max_{d,z,x} \sum_{t=1}^{T} (\, c_t u_t \, + \, \max\{ \, h_t x_{t+1} \, , \; -b_t x_{t+1} \} \,) \qquad (16) \\ \text{s.t.} \\ x_{t+1} \quad &= \quad x_t + u_t - d_t, \quad 1 \leq t \leq T, \\ d_t \quad &= \quad \mu_t \, + \, \delta_t z_t, \\ z_t \quad &\in \quad [-1, 1], \\ \sum_{j=1}^{t} |z_j| \quad &\leq \quad \Gamma_t, \quad 1 \leq t \leq T. \end{aligned}$$

We have that $R^* \leq C^*$ and the gap can be large. However, [BT05] empirically shows that in the case of stationary costs (11) provides an effective approximation to (15). This is significant because (16) is a non-convex optimization problem.

In addition, again in the case of stationary costs, it is shown in [BT05] that LP (11) is essentially equivalent to an inventory problem with known demands, and as a result the solution to the LP amounts to a basestock policy with basestock $\sigma_t = \mu_t + \frac{b-h}{b+h}(A_t - A_{t-1})$, with $A_0 = 0$. In fact, LP (11) can be solved "greedily" (every y_t simultaneously minimized) in the stationary case. Although the non-stationary case is not considered in [BT05], we can say that in that case similar results do not hold.

Next we review the results in [BGNV05] in the context of our basic inventory problem. There are three ingredients in their model. First, motivated by prior work, and by ideas from Control Theory, the authors propose an *affine control* algorithm. Namely, the algorithm in [BGNV05] will construct for each period $1 \leq t \leq T$ parameters $\hat{\alpha}_t^j$ $(0 \leq j \leq t-1)$ and impose the control law:

$$u_t \quad = \quad \hat{\alpha}_0^t \, + \, \sum_{i=1}^{t-1} \hat{\alpha}_i^t d_i, \tag{17}$$

in addition to nonnegativity of the u_t (this extends the methodology described in [BGGN04]). When used at time t, the values d_j in (17) are the past demands. Using (17), the inventory holding/backlogging cost inequalities for time t become:

$$y_t \; \geq \; h_t \left(x_1 \, + \, \sum_{i=1}^{t-1} \left(\sum_{j=i+1}^{t} \hat{\alpha}_i^j - 1 \right) d_i - d_t + \sum_{j=1}^{t} \hat{\alpha}_0^j \right) t = 1, \ldots T, \tag{18}$$

$$y_t \; \geq \; b_t \left(-x_1 + \sum_{i=1}^{t-1} \left(1 - \sum_{j=i+1}^{t} \hat{\alpha}_i^j \right) d_i + d_t - \sum_{j=1}^{t} \hat{\alpha}_0^j \right) t = 1, \ldots T, \tag{19}$$

In addition, [BGNV05] posits that the quantities y_t can be approximated (or at least, upper-bounded) by affine functions of the past demand; the algorithm sets parameters $\hat{\beta}_j^t$ $(0 \leq j \leq t-1)$ with $y_t = \sum_{j=1}^{t-1} \hat{\beta}_j^t d_j + \hat{\beta}_0^t$. Inserting this expression into (18), and rearranging, we obtain:

$$0 \quad \geq \quad h_t x_1 \, + \, \sum_{i=1}^{t-1} \left(h_t \left(\sum_{j=i+1}^{t} \hat{\alpha}_i^j - 1 \right) - \hat{\beta}_i^t \right) d_i \; - \; h_t d_t \; + \; h_t \sum_{j=1}^{t} \hat{\alpha}_0^j \; - \; \hat{\beta}_0^t, \tag{20}$$

which can be abbreviated as

$$0 \;\geq\; \sum_{i=1}^{t} P_i^t(\hat{\alpha},\hat{\beta})\, d_i \;+\; P_0^t(\hat{\alpha},\hat{\beta}), \tag{21}$$

where each $P_i^t(\hat{\alpha},\hat{\beta})$ is an affine function of $\hat{\alpha}$ and $\hat{\beta}$ (and similarly with (19)). The algorithm in [BGNV05] chooses the $\hat{\alpha}$ and $\hat{\beta}$ values so that (20) holds for each demand in the uncertainty set. This set is given by $d_t \in [\mu_t - \delta_t\,,\, \mu_t + \delta_t]$, where $0 \leq \delta_t \leq \mu_t$ are known parameters. Thus, (21) holds for each allowable demand if and only if there exists values $\hat{\nu}_i^t$, $1 \leq i \leq t$, such that

$$0 \;\geq\; \sum_{i=1}^{t} \left(P_i^t(\hat{\alpha},\hat{\beta})\, \mu_i \;+\; \hat{\nu}_i^t \delta_i \right) \;+\; P_0^t(\hat{\alpha},\hat{\beta}), \tag{22}$$

$$-\hat{\nu}_i^t \;\leq\; P_i^t(\hat{\alpha},\hat{\beta}) \;\leq\; \hat{\nu}_i^t, \quad 1 \leq i \leq t. \tag{23}$$

Inequalities (22) and (23), which are linear in $\hat{\alpha}, \hat{\beta}, \hat{\nu}$ make up the system that is enforced in [BGNV05] (there is an additional set of variables, similar to the $\hat{\nu}$, that is used to handle the backlogging inequalities (19)). Notice, as was the case in [BT05], that this approach is conservative in that we may have $\nu_i^t \neq \nu_i^{t'}$ for some i and $t \neq t'$, i.e. the demands implied by some inequality (22) for some t may be different from those arising from some other period t'. Thus, the underlying min-max problem (over the uncertainty set $d_t \in [\mu_t - \delta_t\,,\, \mu_t + \delta_t]$ for each t) is being approximated.

Partly in order to overcome this conservatism, [BGNV05] introduces its third ingredient. Given that the orders and the holding/backlogging costs are represented as affine functions of the demands, the total cost can be described as an affine function of the demands; let us write the total cost as $Q_0 + \sum_t Q_t d_t$ where each $Q_t = Q_t(\hat{\alpha},\hat{\beta})$ is itself an affine function of $\hat{\alpha}, \hat{\beta}$. To further limit the adversary, [BGNV05] models:

$$\text{cost} \;=\; \max\left\{ Q_0 \;+\; \sum_t Q_t d_t \;:\; d \in \mathcal{E} \right\}, \quad \text{where} \tag{24}$$

$$\mathcal{E} \;=\; \left\{ d \;:\; (d-\mu)' S (d-\mu) \;\leq\; \Omega \right\}. \tag{25}$$

Here, ' denotes transpose, S is a symmetric, positive-definite $T \times T$ matrix of known values, $\Omega > 0$ is given and μ is the vector of values μ_t. Thus, (25) states that the demands cannot simultaneously take values "far" from their nominal values μ_t. As shown in [BGNV05], the system (24), (25) is equivalent to the problem:

$$\text{cost} \;=\; \min E, \tag{26}$$

$$\text{subject to:} \qquad Q_0 \;+\; \sum_t \mu_t Q_t \;+\; \left(\Omega\, Q'\, S^{-1}\, Q \right)^{1/2} \;-\; E \;\leq\; 0. \tag{27}$$

In this inequality, Q is the vector with entries Q_t.

In summary, the approach used in [BGNV05] to handle the robust basic inventory model solves the optimization problem with variables E, $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\nu}$; with objective (26), and constraints (27), (22) and (23) (and nonnegativity of the orders, enforced through (17)). Such a problem can be efficiently solved using modern algorithms. [BGNV05] reports excellent results in examples with $T = 24$.

1.2 Results in this paper

In this paper we present algorithms for solving the optimal robust basestock problem (2) using two different models for the demand uncertainty set $\mathcal{D}$. The algorithms are based on a common approach, Benders' decomposition [B62], and extensive experimentation shows them to be quite fast.

Our results can be summarized as follows:

(i) Our algorithms compute optimal basestock levels, a problem of concrete practical importance due to widespread use. We solve the problems to proved optimality, up to roundoff error. Further, we demonstrate, empirically, that using incorrect basestock settings can lead to a substantial cost increase.

(ii) In our numerical experiments we consider two models of demand uncertainty, and in each case we solve the actual min-max optimization problem, and not a conservative approximation. Despite the fact that we solve non-convex optimization problems, extensive experimentation shows that our algorithms scale well with problem size, typically solving problems with several hundred periods in a few minutes of CPU time, in the worst case, and significantly faster in many cases. Further, in the case of the hardest problems we consider, we also describe an approximation scheme that produces solutions which are proved near-optimal significantly faster.

(iii) All of our algorithms can be viewed as variations on Benders' decomposition – this approach should extend well to many demand uncertainty models.

(iv) We present theoretical results concerning robust safety-stock selection, and extensions to other models, such as *ambiguous* uncertainty models (models where the demand distribution is stochastic but only partially known to the decision maker).

In this paper we consider the following models for the demand uncertainty set:

1. The Bertsimas-Thiele model (7)-(9). We will refer to this as the *risk budgets* model. We also consider a broad generalization of this model, which we term the *intervals model*.

2. Based on empirical data from our industrial partner, and borrowing ideas from *adversarial queueing theory*, we consider a simple model of burstiness in demand. In this model, each time period t is either *normal* or a *exceptional* period, and demand arises according to the rules:

   (B.a) In a normal period, we have $d_t \in [\mu_t - \delta_t\,,\ \mu_t + \delta_t]$, where $0 \leq \delta_t \leq \mu_t$ are given parameters.

   (B.b) In a exceptional period, $d_t = P_t$, where $P_t > 0$ is given.

   (B.c) There is a constant $0 < W \leq T$ such that in any interval of W consecutive time periods there is at most one exceptional period.

   The quantities P_t are called the *peaks*. (B.b) and (B.c) model a severe "burst" in demand, which is rare but does not otherwise impact the "normal" demand. For such a model we would employ a P_t value that is "large" compared to the normal demand, e.g. $P_t = \mu_t + 3\delta_t$. However, our approach does not make any assumption concerning the P_t, other than $P_t \geq 0$. We will refer to (B.a)-(B.c) as the *bursty demand* model.

   There are many possible variations of this model, for example: having several peak types, or non-constant window parameters W. Our algorithms are easily adapted to these models.

We also consider the *static* robust inventory problem, which is defined by:

$$\min_{u \geq 0} \ \sum_{t=1}^{T} c_t u_t \quad + \quad K(u) \tag{28}$$

where for $u = (u_1, u_2, \ldots, u_T) \geq 0$,

$$\begin{aligned} K(u) \quad = \quad & \max \quad \sum_{t=1}^{T} \max\{\, h_t x_{t+1}\,,\ -b_t x_{t+1}\} \\ & \text{s.t.} \quad x_{t+1} \,=\, x_t + u_t - d_t, \quad 1 \leq t \leq T, \\ & \qquad (d_1, d_2, \ldots, d_T) \,\in\, \mathcal{D}. \end{aligned} \tag{29}$$

Here (29) is the adversarial problem: given orders u, the adversary chooses demands d so as to maximize the total inventory cost. We study problem (28) not only because it is of interest on its own right, but because it serves as a proof-of-concept for our basic algorithmic ideas. In addition, by running the static model at every period in a rolling horizon fashion, we obtain a *dynamic* strategy, though of course not a basestock strategy. Our algorithms are especially effective on the static problem, solving instances with thousands of time periods in a few seconds.

Our algorithms can be viewed as variants of Benders' decomposition; next we provide a generic blueprint. Recall that $\mathcal{D}$ denotes the demand uncertainty set. We will use Π to denote the set of available policies: for example, in the constant basestock case Π is the set of basestock policies. Thus, the generic problem we want to solve can be written as:

$$\min_{\pi \in \Pi} \ \max_{d \in \mathcal{D}} \ cost(\pi, d), \tag{30}$$

where for any policy $\pi \in \Pi$ and any demand pattern $d = (d_1, \ldots, d_T) \in \mathcal{D}$,

$$cost(\pi, d) \ = \ \sum_{t=1}^{T} c_t \ u(\pi, d, t) \quad + \quad \max\{ \, h_t \ x(\pi, d, t+1) \, , \ -b_t \ x(\pi, d, t+1) \, \}. \tag{31}$$

In this expression, $u(\pi, d, t)$ denotes the order that would be placed by policy π at time t under demands d, and $x(\pi, d, t)$ would likewise denote the inventory at the start of period t. For example, in the basestock case with basestock σ^π, we would have $u(\pi, d, t) = \max\{0 \, , \, \sigma^\pi - x(\pi, d, t)\}$.

Our generic algorithm, given next, will maintain a *working list* $\tilde{D}$ of demand patterns – each member of $\tilde{D}$ will be demand vector $(d_1, d_2, \ldots, d_T) \in \mathcal{D}$. The algorithm will also maintain an upper bound U and a lower bound L on the value of problem (30).

Algorithm 1.1 *GENERIC ALGORITHM*

Initialize: $\tilde{D} = \emptyset$, $L = 0$ and $U = +\infty$.

1. **Decision maker's problem.** Let $\tilde{\pi}$ be the solution to the problem:

   $\min_{\pi \in \Pi} \ \max_{d \in \tilde{D}} cost(\pi, d)$.

   Set $L \leftarrow \max_{d \in \tilde{D}} cost(\tilde{\pi}, d)$.

2. **Adversarial problem.** Let $\bar{d}$ be the solution to the problem:

   $\max_{d \in \mathcal{D}} \ cost(\tilde{\pi}, d)$.

   Set $U \ \leftarrow \ \min \{U \, , \, cost(\tilde{\pi}, \bar{d})\}$.

3. **Termination test.** If $U - L$ is small enough, then **EXIT**.

4. **Formulation update.** Otherwise, add $\bar{d}$ to $\tilde{D}$ and return to Step 1.

Note that the decision maker's problem is of the same general form as the generic problem (30) – however, the key difference is that while $\mathcal{D}$ is in general exponentially large, at any point $\tilde{D}$ has size equal to the number of iterations run so far. One of the properties of Benders' decomposition is that, when successful, the number of iterations until termination will be small. In our implementations, this number turned out quite small indeed, as we will see.

In fact, the decision maker's problem proves to be quite tractable: roughly speaking, it amounts to an easily solvable convex optimization problem. For example, in the case of static policies the problem can be formulated as a linear program with $O(T|\tilde{D}|)$ variables and constraints.

On the other hand, the adversarial problem is non-convex. In at least one case we can show that it is NP-hard [O06]. (But this is only half of the story, because in that case the adversarial problem can be ϵ-approximated, i.e. solutions arbitrarily close to the optimum can be efficiently computed [O06]). The problem can be also modeled as a mixed-integer program, but tackling this mixed-integer program directly turns out not to be the best approach. Instead, we devise simple combinatorial algorithms that prove efficient.

Benders' decomposition algorithms have long enjoyed popularity in many contexts. In the case of stochastic programming with large number of scenarios, they prove essential in that they effectively reduce a massively large continuous problem into a number of much smaller independent problems. In the context of non-convex optimization (such as the problem handled in this paper) the appeal of decomposition is that it vastly reduces *combinatorial* complexity.

Benders' decomposition methods can be viewed as a special case of cutting-plane methods. As is the case for cutting-plane methods for combinatorial optimization, there is no adequate general theory to explain why Benders' decomposition, when adequately implemented, tends to converge in few iterations. In the language of our algorithm, part of an explanation would be that the demand patterns $\bar{d}$ added to $\tilde{D}$ in each execution of Step 4 above are "important" or "essential", as well as being "extremal".

A final point regarding Algorithm 1.1 is that neither Step 1 nor Step 2 need be carried out exactly, except for the last iteration (in order to prove optimality). When either step is performed approximately, then we cannot update the corresponding bound (U or L) as indicated in the blueprint above. However, for example, performing Step 2 approximately can lead to faster iterations, and at an early stage an approximate solution can suffice since all we are trying to do, at that point, is to quickly improve the approximation to the set $\mathcal{D}$ provided by the existing (and much smaller) set $\tilde{D}$.

Notation 1.2 *In what follows, for any time period t, and any value z, we write*

$$\mathcal{W}_t(z) \;=\; \max\{\, h_t z\,,\; -b_t z\,\}.$$

We will refer to the inventory holding/backlogging cost in any period as the inventory cost.

This paper is organized as follows. Section 2 presents our results on the static problem. Section 4 presents our algorithms for the robust constant basestock problem, while Section 5 presents numerical experiments involving these algorithms. Our algorithmic results for the robust safety-stocks problem are described in Section 6.2, and ambiguous models are considered in Section 6.3.

2 The static problem

In this section we present algorithms for the static problem (15) for the risk budgets and the bursty demand models of demand uncertainty. Our algorithms follow the template provided by Algorithm 1.1.

In the case of the static problem, clearly the decision maker's problem (step 1 of Algorithm 1.1) can be formulated as a linear program with $O(T|\mathcal{D}|)$ variables and constraints. Our algorithms for the risk budgets and the bursty demand model will differ in how we handle the adversarial problem.

2.1 The adversarial problem in the risk budgets model

Here we consider the adversarial problem (step 2 of Algorithm 1.1) under the demand uncertainty model (7)-(9). For simplicity of presentation, in this section we will assume that the quantities Γ_t are integral – in Section 4.2, where we consider the basestock problem with the risk budgets,

we will allow the Γ_t to be fractional. But the integral Γ_t case already incorporates most of the complexity of the problem.
We have the following simple result:

Lemma 2.1 *Let $\bar{d}$ be an extreme point of $\mathcal{D}$. Then for $1 \le t \le T$, either $\bar{d}_t = \mu_t$ or $|\bar{d}_t - \mu_t| = \delta_t$.* ■

Using this lemma, we can now devise an algorithm for the adversarial problem. Let $\tilde{u}$ be a vector of orders. In the remainder of this section we will assume that $\tilde{u}$ is fixed. For $1 \le t \le T$, and for any integer k with $0 \le k \le \Gamma_t$, let $\mathcal{A}_t(x,k)$ denote the maximum cost that the adversary can attain in periods $t,\ldots,T$, assuming starting inventory at time t equal to x, and that k "units" of risk have been used in all periods preceding t. Formally,

$$\begin{aligned} \mathcal{A}_t(x,k) &= \max_d \sum_{j=t}^{T} \mathcal{W}_j \left(x + \sum_{i=1}^{j} \tilde{u}_i - \sum_{i=1}^{j} d_i \right) \qquad (32) \\ \text{Subject to} \quad & \sum_{i=t}^{j} \left\{ \frac{|d_i - \mu_i|}{\delta_i} \,:\, \delta_i > 0 \right\} \le \Gamma_j - k, \quad t \le j \le T, \\ & \mu_j - \delta_j \le d_j \le \mu_j + \delta_j, \quad t \le j \le T. \end{aligned}$$

Using this notation, the value of the adversarial problem equals $\sum_{t=1}^{T} c_t \tilde{u}_t \quad + \quad \mathcal{A}_1(x_1, 0)$. Now (32) amounts to a linearly constrained program (in the d variables plus some auxiliary variables) and it is easily seen that the demand vector that attains the maximum in $\mathcal{A}_1(x_1,0)$ is an extreme point of $\mathcal{D}$. Thus, using Lemma 2.1, we have the following recursion:

$$\mathcal{A}_t(x,\Gamma_t) = \mathcal{W}_t(x + \tilde{u}_t - \mu_t) + \mathcal{A}_{t+1}(x + \tilde{u}_t - \mu_t, \Gamma_t), \qquad (33)$$

while for $k < \Gamma_t$,

$$\mathcal{A}_t(x,k) = \max\left\{ f^u_{t,k}(x)\,,\, f^d_{t,k}(x)\,,\, f^m_{t,k}(x) \right\}, \qquad (34)$$

where

$$\begin{aligned} f^u_{t,k}(x) &= \mathcal{W}_t(x + \tilde{u}_t - \mu_t - \delta_t) + \mathcal{A}_{t+1}(x + \tilde{u}_t - \mu_t - \delta_t, k+1), & (35) \\ f^d_{t,k}(x) &= \mathcal{W}_t(x + \tilde{u}_t - \mu_t + \delta_t) + \mathcal{A}_{t+1}(x + \tilde{u}_t - \mu_t + \delta_t, k+1), & (36) \\ f^m_{t,k}(x) &= \mathcal{W}_t(x + \tilde{u}_t - \mu_t) + \mathcal{A}_{t+1}(x + \tilde{u}_t - \mu_t, k). & (37) \end{aligned}$$

Here we write $\mathcal{A}_{T+1}(x,k) = 0$ for all x, k As a result of equation (34) and the definition of the $f^u_{t,k}(x)$, $f^d_{t,k}(x)$, $f^m_{t,k}(x)$, we have:

Lemma 2.2 *For any t and k, $\mathcal{A}_t(x,k)$ is a convex, piecewise-linear function of x.* ■

Equations (33) and (34)-(37) provide a dynamic programming algorithm for computing $\mathcal{A}_1(x_1,0)$. In the rest of this section we provide simple details needed to make the algorithm efficient. We will use the following notation: the *representation* of a convex piecewise-linear function f is the description of f given by the slopes and breakpoints of its pieces, sorted in increasing order of the slopes (i.e., "left to right").

Lemma 2.3 *For $i = 1, 2$, let f^i be a convex piecewise-linear function with slopes $s^i_1 < s^i_2 < \ldots < s^i_{m(i)}$. Suppose that, for some $q > 0$, f^1 and f^2 have q pieces of equal slope, i.e. there are q pairs $1 \le a \le m(1), 1 \le b \le m(2)$, such that $s^1_a = s^2_b$. Then (a) $g = \max\{f^1, f^2\}$ has at most $m(1) + m(2) - q$ pieces. Furthermore (b) given the representations of f^1 and f^2, we can compute the representation of g in time $O(m(1) + m(2))$.*

Proof. First we prove (b). Let $v_1 < v_2 < \ldots < v_n$ be the sequence of all breakpoints of f^1 and f^2, in increasing order, where $n \leq m(1) + m(2) - 2$. Suppose that for some $1 \leq i < n$ we have that $f^1(v_i) \geq f^2(v_i)$ and $f^2(v_{i+1}) \geq f^1(v_{i+1})$ where at least one of the two inequalities is strict. Then the interval $[v_i, v_{i+1}]$ contains one breakpoint of g. In fact, with the exception of at most two additional breakpoints involving the first and last pieces of f^1 and f^2, *every* breakpoint of g arises in this form or by exchanging the roles of f^1 and f^2. This proves (b), since given the representation of f^1 and f^2 we can compute the sorted list $v_1 < v_2 < \ldots < v_n$ in time $O(m(1) + m(2))$. To prove (a), note that any piece of g is either (part) of a piece of $m(1)$ or $m(2)$; thus, since g is convex, for any pair $1 \leq a \leq m(1), 1 \leq b \leq m(2)$, with $s_a^1 = s_b^2$ $(= s$, say) there is at most one piece of g with slope s. ■

For extensions, see [O06]. In our implementation, we use the method implicit in Lemma 2.3 together with the dynamic programming recursion described above. We will present computational experience with this algorithm below. Here we present some comments on its complexity.

Note that in each equation (35)-(37) the corresponding function $f^u_{t,k}$, $f^d_{t,k}$ or $f^m_{t,k}$ has at most one more breakpoint than the A_{t+1} function in that equation. Nevertheless, the algorithm we are presenting is, in the worst case, of complexity exponential in T. However, this is an overly pessimistic worst-case estimate. Comparing equations (35) and (36), we see that $f^u_{t,k}(x) = f^d_{t,k}(x - 2\delta_t)$. Thus, as is easy to see (see Lemma 2.4 below) $\max\{f^u_{t,k}, f^d_{t,k}\}$ has *no* more breakpoints than $f^u_{t,k}$, which also has at most one more breakpoint than $\mathcal{A}_t(x, k+1)$.
Further, Lemma 2.3 (a) is significant in that when we consider equations (35)-(37) we can see that, in general, the functions f^u, f^d and f^m will have *many* pieces with equal slope. In fact, in our numerical experiments, we have not seen any example where the number of pieces of $\mathcal{A}_1(x, 0)$ was large. We conjecture that for broad classes of problems our dynamic-programming procedure runs in polynomial time.

2.1.1 A special case

There is an important special case where we can prove that our algorithm is efficient. This is the case where the demand uncertainty set is described by the condition that $d_t \in [\mu_t - \delta_t\,,\, \mu_t + \delta_t]$ for each t. In terms of the risk budgets model, this is equivalent to having $\Gamma_t = t$ for each t. We will refer to this special case as the *box* model.

In this case, the extreme points of the demand uncertainty set $\mathcal{D}$ are particularly simple: they satisfy $d_t = \mu_t - \delta_t$ or $d_t = \mu_t + \delta_t$ for each t. Let $\mathcal{A}_t(x)$ denote the maximum cost that the adversary can attain in periods $t, \ldots, T$, assuming that the starting inventory at time t equals x. Then:

$$\mathcal{A}_t(x) \;=\; \max\left\{\, f^u_t(x)\,,\, f^d_t(x)\right\}, \tag{38}$$

where

$$f^u_t(x) \;=\; \mathcal{W}_t(x + \tilde{u}_t - \mu_t - \delta_t) \;+\; \mathcal{A}_{t+1}(x + \tilde{u}_t - \mu_t - \delta_t), \tag{39}$$

$$f^d_t(x) \;=\; \mathcal{W}_t(x + \tilde{u}_t - \mu_t + \delta_t) \;+\; \mathcal{A}_{t+1}(x + \tilde{u}_t - \mu_t + \delta_t). \tag{40}$$

and as before we set $\mathcal{A}_{T+1}(x) = 0$. We have, as a consequence of Lemma 2.3:

Lemma 2.4 *Let f be a piecewise-linear, convex function with m pieces, and let a be any value. Then $g(x) \doteq max\{\, f(x)\,,\, f(x+a)\,\}$ is convex, piecewise-linear with at most m pieces.* ■

Corollary 2.5 *For any t, the number of pieces in $\mathcal{A}_t(x)$ is at most $T - t + 2$.* ■

Corollary 2.6 *In the box model, the adversarial problem can be solved in time $O(T^2)$.* ■

Corollary 2.6 is significant for the following reason. In the box case, our min-max problem (30) can be written as:

$$\min_{u \geq 0} \quad \sum_{t=1}^{T} c_t \, u_t \quad + \quad z \tag{41}$$

$$\text{Subject to} \qquad z \geq \sum_{j \in J} h_j \left(x_1 + \sum_{i=1}^{j} \tilde{u}_i - \sum_{i=1}^{j} d_i \right) - \sum_{j \in \bar{J}} b_j \left(x_1 + \sum_{i=1}^{j} \tilde{u}_i - \sum_{i=1}^{j} d_i \right),$$

$$\text{for all } d \in \mathcal{D}, \text{ and each partition } (J, \bar{J}) \text{ of } \{1, \ldots, T). \tag{42}$$

This linear program has $T + 1$ variables but $2^T \, |\mathcal{D}|$ constraints. However, by Corollary 2.6, we can solve the *separation problem* for the feasible set of the linear program in polynomial time – hence, we can solve the min-max problem in polynomial time, as well [GLS93]. This result is of theoretical relevance only – in the box demands case, our generic Benders' algorithm proves especially efficient.

2.1.2 The adversarial problem as a mixed-integer program

Even though we are using a dynamic-programming algorithm to solve the adversarial problem, we can also use mixed-integer programming. In the following formulation $\tilde{u}$ is the given orders vector. For each period t, there is a zero-one variable p_t which equals 1. All other variables are continuous, and the M_t are large enough constants.

$$\max_{d,x,p,I,B,z} \qquad \sum_{t=1}^{T} (I_t + B_t) \tag{43}$$

Subject to

for $1 \leq t \leq T$,

$$x_{t+1} = x_t + \tilde{u}_t - d_t, \tag{44}$$

$$h_t \, x_{t+1} \leq I_t \leq h_t \, x_{t+1} + h_t \, M_t (1 - p_t), \tag{45}$$

$$0 \leq I_t \leq h_t \, M_t \, p_t, \tag{46}$$

$$-b_t \, x_{t+1} \leq B_t \leq -b_t \, x_{t+1} + b_t \, M_t p_t, \tag{47}$$

$$0 \leq B_t \leq b_t \, M_t \, (1 - p_t), \tag{48}$$

$$d_t = \mu_t + \delta_t \, z_t, \tag{49}$$

$$p_t = 0 \text{ or } 1, \tag{50}$$

$$\sum_{j=1}^{t} |z_t| \leq \Gamma_t. \tag{51}$$

Equations (45)-(48) imply that when if $h_t \, x_{t+1} > 0$ then $p_t = 1$, and when $p_t = 1$ then $I_t = h_t \, x_{t+1}$ and $B_t = 0$; whereas if $-b_t \, x_{t+1} > 0$ then $p_t = 0$, and when $p_t = 0$ then $B_t = -b_t \, x_{t+1}$ and $I_t = 0$. Similarly with B_t. In order for the formulation to be valid we need to choose the constants M_t appropriately large – however, for ease of solvability, they should be chosen just large enough, and this can be done in a straightforward fashion.

Problem (43) bears a passing similarity to the traditional *economic lot-sizing* problem. As a result, we would expect modern mixed-integer programming software to handle the problem with ease. The following table shows sample computational experience using Cplex 9.0 on a current workstation to solve three examples. In this table "time" is the time to termination (in seconds) and "BB nodes" is the number of branch-and-cut nodes.

These results are disappointingly poor – in fact, in the example with $T = 96$, achieving a near-optimal solution was already quite expensive. This makes the mixed-integer programming approach uncompetitive with the dynamic programming algorithm given above, which solves problems with $T = 500$ in seconds.

T	24	48	96
time (sec.)	0.12	227	16449
BB nodes	84	215922	7910537

Table 1: Solving the adversarial problem as a mixed-integer program

Nevertheless, it is possible that a more efficient specialized algorithm for solving the mixed-integer program (43), or for a reformulation of it (there are many) could be developed. In fact, notice that by replacing equation (51) with the general condition $d \in \mathcal{D}$ we can in principle tackle the adversarial problem for general polyhedral set $\mathcal{D}$.

2.2 The adversarial problem in the bursty demand model

Here we consider the adversarial problem for the bursty demand model given in Section 1.2. We can adapt the dynamic programming recursion used for the risk budgets model as follows. As previously, we assume a given vector $\tilde{u}$ of orders.

For each period t, and each integer $1 \leq k < \min\{W, t\}$, let $\Pi_t(x, k)$, denote the maximum cost attainable by the adversary in periods $t, \ldots, T$ assuming that the initial inventory at the start of period t is x, and that the last peak occurred in period $t-k$. Similarly, denote by $\Pi_t(x, 0)$ the maximum cost attainable by the adversary in periods $t, \ldots, T$ assuming that the initial inventory at the start of period t is x, and that no peak occurred in periods $t-1, t-2, \ldots, \max\{1, t-W+1\}$. Writing $\Pi_{T+1}(x, k) = 0$, we have, for $1 \leq t \leq T$:

$$\begin{aligned}
\Pi_t(x,k) &= \max_{d \in \{\mu_t - \delta_t, \mu_t + \delta_t\}} \{\mathcal{W}_t(x + \tilde{u}_t - d) + \Pi_{t+1}(x + \tilde{u}_t - d, k+1)\}, \\
&\qquad \text{for } 1 \leq k < \min\{W-1, t\}, && (52)\\
\Pi_t(x, W-1) &= \max_{d \in \{\mu_t - \delta_t, \mu_t + \delta_t\}} \{\mathcal{W}_t(x + \tilde{u}_t - d) + \Pi_{t+1}(x + \tilde{u}_t - d, 0)\}, \\
&\qquad \text{for } W - 1 < t, && (53)\\
\Pi_t(x,0) &= \max\left\{\Pi_t^1(x), \Pi_t^0(x)\right\}, \quad \text{where} && (54)\\
\Pi_t^1(x) &= \max_{d \in \{\mu_t - \delta_t, \mu_t + \delta_t\}} \{\mathcal{W}_t(x + \tilde{u}_t - d) + \Pi_{t+1}(x + \tilde{u}_t - d, 0)\}, \quad \text{and} && (55)\\
\Pi_t^0(x) &= \mathcal{W}_t(x + \tilde{u}_t - P_t) + \Pi_{t+1}(x + \tilde{u}_t - P_t, 1). && (56)
\end{aligned}$$

We solve this recursion using the same approach as for (33)-(37), i.e. by storing the representation of each function $\Pi_t(x, k)$ (which clearly are convex piecewise-linear).

3 Computational results for the static problem

To investigate the behavior of our algorithms for the static case, we ran several battery of tests, with results reported in Table 3. In this table, we report tests involving the budgets and the bursty model of uncertainty, with three different kinds of data: random, periodic and discounted. Further, we consider $T = 50, 200,$ and 500. We ran 500 tests for each separate category, and for each category we report the average, maximum and minimum running time and number of steps to termination.

For all of the data types, we generate problem parameters randomly. We assume that each period corresponds to a week and a year has 52 weeks. In the periodic case we generate cost parameters and demand intervals corresponding to 3 months (13 weeks) and assume that data repeats every 3 months. For the discounted case we generate the cost data corresponding to one period and generate the cost for the other periods by discounting these cost parameters with a yearly discount rate of 0.95. We generated the demand intervals in that case randomly (see below). For the pure random case, data in each period is generated independent from the other periods.

In generating the cost parameters we assumed that there are two possibilities. In each period, each cost parameter is uniformly distributed either in some interval $[l_1, l_2]$ with probability p or in

interval $[h_1, h_2]$ with probability $1-p$. We generated the mid-points of the intervals where demand resides using the same method. The half-lengths of the intervals are generated by multiplying the mid-point with a random number which is uniformly distributed between 0 and 1. Table 2 demonstrates the parameters we used.

	$[l_1, l_2]$	$[h_1, h_2]$	**p**
c	[0,2]	[6,8]	0.5
h	[5,10]	[15,25]	0.5
b	[5,15]	[20,30]	0.5
d	[0,100]	[200,400]	0.7

Table 2: Parameters for data generation

The peak quantities in the bursty demand model were generated by multiplying the mid-point of the demand interval by 5.

For the demand model with risk budgets we generated budgets in two ways. First, randomly. Here, starting from budget 0, we generated a budget for each period by increasing the budget in the previous period by one with probability q which is also randomly generated.

We also tested our algorithm with stationary instances in which the budgets are generated by the algorithm the given in [BT05]. Let d be a demand vector and let $C(d, \Gamma)$ be the cost of this demand vector with the optimal robust policy computed by our algorithm for the budget vector Γ. The method in [BT05] assumes that d is a random vector and generates the Γ vector that minimizes an upper bound on $E[C(d, \Gamma)]$ assuming that the first two moments of the distribution is given. The algorithm gives budgets which are not necessarily integral. We round them down, since our algorithm for the static model can only handle the integral budget case. These results are given in Table 4.

		Running Time (sec.)			Number of Iterations		
	# periods	**average**	**max**	**min**	**average**	**max**	**min**
Random (bursty)	50	0.073	0.28	0.01	4.23	10	3
	200	3.28	1.21	0.37	4.45	9	3
	500	53.6	241	3.94	4.44	11	3
Random (budgets)	50	0.03	0.10	0.01	4.10	8	3
	200	1.22	3.60	0.58	4.39	10	3
	500	20.00	43.90	10.90	4.18	8	3
Periodic (bursty)	50	0.07	0.17	0.01	4.03	7	3
	200	3.00	11.90	0.35	4.26	9	3
	500	42.10	149.00	3.85	3.74	7	3
Periodic (budgets)	50	0.04	0.85	0.01	4.33	26	3
	200	0.61	10.00	0.26	4.13	17	3
	500	5.99	33.70	3.19	3.85	11	3
Discounted (bursty)	50	0.07	0.19	0.01	4.03	7	3
	200	3.47	16.20	0.42	4.83	11	3
	500	55.40	336.00	4.68	4.76	15	3
Discounted (budgets)	50	0.03	0.42	0.01	4.28	20	3
	200	0.92	38.70	0.32	4.37	35	3
	500	9.32	238.00	2.71	4.56	26	3

Table 3: Running time and number of iterations

We note the low number of iterations – this shows that on average approximately four demand patterns suffice to prove optimality (of the optimal policy). The maximum we observed is larger but still quite modest. In fact, Table 3 may overstate the amount of work needed to converge.

This is because in addition to requiring few iterations, the algorithm, usually, quickly converged to close to the optimum and the additional iterations were needed in order to close a very small gap. Figure 1 shows a typical example of this behavior.

	Running Time (sec.)			Number of Iterations		
# periods	**average**	**max**	**min**	**average**	**max**	**min**
50	0.22	4.51	0.00	9.21	42	2
200	5.73	39.82	0.05	8.90	23	2
500	50.28	1049.00	0.61	7.11	13	2

Table 4: Running time and number of iterations for model from [BT05]

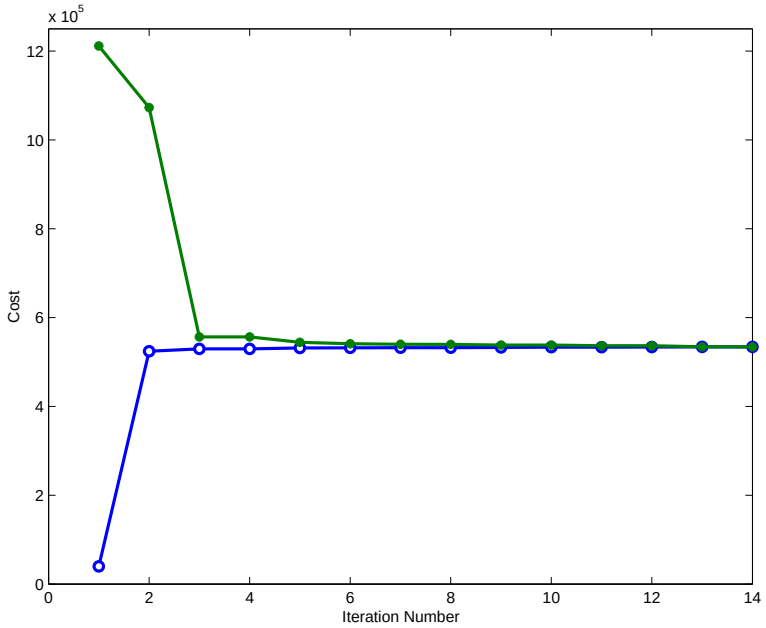

Figure 1: Example with many steps

4 The basestock problem

This section considers how to solve problem (2) using our generic algorithm (1.1), under the risk budgets and bursty demand uncertainty models. Section 4.1 considers the decision maker's problem. The adversarial problem is studied in Section 4.2 (for the risk budgets model) and Section 4.4 (bursty demands model). In the context of algorithm (1.1), a policy $\tilde{\pi}$ consists of a basestock value $\tilde{\sigma}$, and this will be the output of each decision maker's problem; the corresponding adversarial problem will consist of computing the quantity $V(\tilde{\sigma})$ defined in equations (3)-(6).

Prior to describing our algorithms, we note a simple observation.

Definition 4.1 *Consider a demand vector d. $1 \leq t \leq T$, let $\mathcal{R}_{t,d} = x_1 - \sum_{j=1}^{t-1} d_j$. Write $\mathcal{R}_{0,d} = +\infty$.*

Definition 4.2 *Consider a demand vector d and a basestock value σ. We denote by $t^* = t^*_{\sigma,d}$ the smallest $t \leq T$ with $\mathcal{R}_{t,d} \leq \sigma$. If no such t exists we set $t^*_{\sigma,d} = T + 1$.*

In other words, $\mathcal{R}_{t,d}$ is the amount of inventory at the start of period t if no orders are placed in periods $1, \ldots, t-1$, and $t^*_{\sigma,d}$ indicates the first period where, under the policy using basestock σ, the starting inventory does not exceed σ.

Example 4.3 *Suppose* $T = 6$, $d = (10, 8, 0, 15, 4, 9)$ *and* $x_1 = 100$. *Then* $\mathcal{R}_{1,d} = 100$, $\mathcal{R}_{2,d} = 90$, $\mathcal{R}_{3,d} = \mathcal{R}_{4,d} = 82$, $\mathcal{R}_{5,d} = 67$ *and* $\mathcal{R}_{6,d} = 63$. *Also,*

$$t^*_{\sigma,d} = \begin{cases} 1, & for \quad 100 \leq \sigma, \\ 2, & for \quad 90 \leq \sigma < 100, \\ 3, & for \quad 82 \leq \sigma < 90, \\ 5, & for \quad 67 \leq \sigma < 82, \\ 6, & for \quad 63 \leq \sigma < 67, \\ 7, & for \quad \sigma < 63. \end{cases}$$

Remark 4.4 *For* $1 \leq t \leq T$, *we have that* $t^*_{\sigma,d} = t$ *for* $\sigma \in [\,\mathcal{R}_{t,d}\,,\,\mathcal{R}_{t-1,d}\,)$. *Further, (writing* t^* *for* $t^*_{\sigma,d}$*) if we use basestock* σ *under demands* d,

(a) For every $t \geq t^*$, $x_t \leq \sigma$, *and for every* $t \leq t^*$, $x_t = \mathcal{R}_{t,d}$.

(b) For $t < t^*$, $u_t = 0$. *For* $t < t^* - 1$ *we have, by definition of* t^*, *that* $0 \leq \sigma \leq \mathcal{R}_{t+1,d} = x_{t+1}$, *hence the cost incurred at* t *equals* $h_t\,(\,\mathcal{R}_{t+1,d}) = \mathcal{W}_t\,(\mathcal{R}_{t+1,d})$. *We might have that* $x_{t^*} < 0$, *in which case in period* $t^* - 1$ *we pay a backlogging cost. In any case, the cost incurred in period* $t < t^*$ *can be summarized as* $\mathcal{W}_t\,(\mathcal{R}_{t+1,d})$.

(c) At $t = t^*$ *the ordering cost equals* $c_{t^*}\,(\,\sigma - \mathcal{R}_{t^*,d}\,)$ *and the inventory cost is* $\mathcal{W}_t(\sigma - d_{t^*})$.

(d) For $t > t^*$ *we incur an ordering cost of* $c_t d_{t-1}$ *and an inventory cost of* $\mathcal{W}_t(\sigma - d_t)$.

4.1 The decision maker's problem

Here we have a finite set $\tilde{D} \subseteq \mathcal{D}$ and we wish to compute the basestock value that minimizes the maximum cost over any demand pattern in $\tilde{D}$. Consider any demand $d \in \mathcal{D}$. Let $cost_t(\sigma, d)$ denote the cost incurred at time t, under demands d, if we use basestock σ.

Lemma 4.5 *For any fixed* $1 \leq t \leq D$ *and* d, $cost_t(\sigma, d)$ *is a piecewise convex function of* σ *with at most three pieces, each of which is piecewise linear.*

Proof. Suppose that $\sigma < \mathcal{R}_{t,d}$. Then $t^*_{\sigma,d} > t$, and so $cost_t(\sigma, d) = \mathcal{W}_t\,(\mathcal{R}_{t+1,d})$ which is independent of σ. Suppose now that $\sigma \in [\,\mathcal{R}_{t,d}\,,\,\mathcal{R}_{t-1,d}\,)$. Then $t^*_{\sigma,d} = t$ and $cost_t(\sigma, d) = c_t\,(\,\sigma - \mathcal{R}_{t,d}\,) + \mathcal{W}_t(\sigma - d_t)$. Finally, suppose that $\sigma \geq \mathcal{R}_{t-1,d}$. Then $t^* < t$, and $cost_t(\sigma, d) = c_t(d_{t-1}) + \mathcal{W}_t(\sigma - d_t)$. Note that at $\sigma = \mathcal{R}_{t-1,d}$, we have $\sigma - \mathcal{R}_{t,d} = d_{t-1}$. The result is proved. ■

Denoting (as in (31)), $cost(\sigma, d) = \sum_t cost_t(\sigma, d,)$ we have:

Corollary 4.6 *For any demand vector* d, $cost(\sigma, \tilde{d})$ *is a piecewise convex function of* σ *with at most* $3T$ *pieces, each of which is piecewise linear.* ■

Corollary 4.7 $\max_{\tilde{d} \in \tilde{D}} cost(\sigma, \tilde{d})$ *is piecewise convex, with each convex piece being piecewise-linear.* ■

Our objective is to compute $\sigma \geq 0$ so as to minimize $\max_{\tilde{d} \in \tilde{D}} cost(\sigma, \tilde{d})$. To do this, we rely on Lemma 4.6:

(i) Compute, and sort, the set of breakpoints of all functions $cost(\sigma, \tilde{d})$. Let $0 \leq \beta_1 < \beta_2 \ldots < \beta_n$ be the sorted list of nonnegative breakpoints, where $n \leq 3T|\tilde{D}|$.

(ii) In each interval I of the form $[0, \beta_1]$, $[\beta_i, \beta_{i+1}]$ $(1 \leq i < n)$ and $[\beta_n, +\infty)$, we have that $\max_{\tilde{d} \in \tilde{D}} cost(\sigma, \tilde{d})$ is the maximum of a set of convex functions, and hence convex (in fact: piecewise linear). Let $\sigma_I \in I$ be the minimizer of $\max_{\tilde{d} \in \tilde{D}} cost(\sigma, \tilde{d})$ in I.

(iii) Let $\tilde{I} = \text{argmin}_I \ \max_{\tilde{d} \in \tilde{D}} cost(\sigma_I, \tilde{d})$. We set $\tilde{\sigma} = \sigma_{\tilde{I}}$.

In order to carry out Step (ii), in our implementation we used *binary search*. There are theoretically more efficient algorithms, but empirically our implementation is adequate. Note that in order to carry out the binary search in some interval I, we do not explicitly need to construct the representation of $\max_{\tilde{d}\in\tilde{D}} cost(\sigma, \tilde{d})$, restricted to I. Rather, when evaluating some $\hat{\sigma} \in I$ we simply compute its functional value as the maximum, over $\tilde{d} \in \tilde{D}$, of $cost(\hat{\sigma}, \tilde{d})$; and this can done using the representation of each $cost(\hat{\sigma}, \tilde{d})$.

Further, in the context of our generic algorithm 1.1, Step (i) can be performed incrementally. That is to say, when adding a new demand $\bar{d}$ to $\tilde{D}$, we compute the breakpoints of $cost(\sigma, \bar{d})$ and merge these into the existing sorted list, which can be done in linear time.

In summary, all the key steps of our algorithm for the decision maker's problem run linearly in T and $\tilde{D}$.

We stress that the above algorithm is independent of the underlying uncertainty set $\mathcal{D}$. In what follows, we will describe our algorithms for the adversarial problem, under the risk budgets and bursty demand uncertainty models.

4.2 The adversarial problem under the risk budgets model

In this section we consider the adversarial model under the demand uncertainty set $\mathcal{D}$ given by (7)-(9), and assuming that a fixed basestock σ has been given. We let (d^*, z^*) denote the optimal demand (and risks) vector chosen by the adversary. We want to characterize structural properties of (d^*, z^*). In what follows, we write t^* for t^*_{σ,d^*}. First we have the following easy result:

Lemma 4.8 *Suppose $t^* \geq T$. Then d^* is obtained the two following linear programs, and choosing the solution with higher value:*

$$\begin{aligned} Max \quad & \sum_{t=1}^{T} h_t \left(x_1 - \sum_{j=1}^{t} d_j \right) \\ s.t. \quad & d \in \mathcal{D} \\ & x_1 - \sum_{t=1}^{T-1} d_t \geq \sigma. \end{aligned} \tag{57}$$

$$\begin{aligned} Max \quad & \sum_{t=1}^{T-1} h_t \left(x_1 - \sum_{j=1}^{t} d_t \right) + b_T \left(\sum_{t=1}^{T} d_t - x_1 \right) \\ s.t. \quad & d \in \mathcal{D} \\ & x_1 - \sum_{t=1}^{T-1} d_t \geq \sigma. \end{aligned} \tag{58}$$

■

Lemma 4.8 provides one case for our adversarial algorithm. In what follows we will assume that $t^* < T$ and describe algorithms for this case. We will describe two algorithms: an *exact* algorithm, which solves the problem to proved optimality, and a much faster *approximate* algorithm which does not prove optimality but nevertheless produces a "strong" demand pattern $\bar{d}$ which, in the language of our generic algorithm (1.1), quickly improves on the working set $\tilde{D}$. The exact algorithm requires (in a conservative worst-case estimate) the solution of up to $O(T^4\,\Gamma_T)$ warm-started linear programs with fewer than $4T$ variables; as we show in Section 5 it nevertheless can be implemented to run quite efficiently. The approximate algorithm, on the other hand, is significantly faster.

Some additional remarks on the exact algorithm:

(a) When the Γ_t are integral, the step count reduces to $O(T^2\,\Gamma_T)$. In addition, if $\Gamma_t = t$ for each t (i.e. the uncertainty set reduces to the intervals $[\mu_t - \delta_t, \mu_t + \delta_t]$) the complexity reduces to $O(T^2)$, with no linear programs solved. See [O06].

(b) The case of integral Γ_t is of interest because if we use the uncertainty set with risk budgets $\Gamma_t^f = \lfloor \Gamma_t \rfloor$ we obtain a lower bound on the min-max problem, whereas if we use then risk budgets $\Gamma_t^c = \lceil \Gamma_t \rceil$ we obtain an upper bound. In fact, the *superposition* of the two uncertainty sets should provide a good approximation to the min-max problem (see Section 6.4). Further,we present a bounding procedure based on this idea, which proves excellent bounds, significantly faster than the algorithm for fractional Γ_t.

We begin with the exact algorithm. Lemmas 4.9 and 4.10 and Remark 4.11 provide some structural properties of an optimal solution to the adversarial problem. Sections 4.2.1, 4.2.2 and 4.2.3 describe the technical details of our approach. The overall algorithm is put together in Section 4.2.4. The approximate algorithm is described in Section 4.2.5, and the bounding procedure based on integral budgets is given in Section 4.3. The reader may skip over Sections 4.2.1, 4.2.2 and 4.2.3 without loss of continuity.

Lemma 4.9 *Either (a) there is a period $t^e \geq t^*$ such that $\sum_{j=1}^{t^e} |z_j^*| = \Gamma_t$, or (b) without loss of generality $|z_t^*| = 1$ for every $t \geq t^*$.*

Proof. Assume no period t^e as in (a) exists, and suppose that $|z_t^*| < 1$ for some $t \geq t^*$. Since $\sum_{j=1}^{k} |z_j^*| < \Gamma_k$ for all $k \geq t$, it follows we can increase $|z_t|$, i.e. $|d_t - \mu_t|$, and remain feasible Using Remark 4.4, (c) and (d), the cost paid as a function d_t equals $c_{t+1}d_t + \mathcal{W}_t(\sigma - d_t)$ (where $c_{T+1} = 0$), which is a convex function of d_t. Hence we can increase $|z_t|$ without decreasing the cost, which proves the claim. ■

Note that, given for a given t^*, case (b) of Lemma 4.9 is simple: we simply need to set, for each $t \geq t^*$, either $d_t = \mu_t + \delta_t$ or $d_t = \mu_t - \delta_t$, so as to maximize $\mathcal{W}_t(\sigma - d_t) + c_{t+1}d_t$. For case (b) to hold, we must have that $\sum_{t=1}^{t^*-1} |z_{t^*-1}^*| \leq \Gamma_T - (T - t^* + 1)$. So, for a given t^*, case (b) amounts to solving the linear program:

$$
\begin{aligned}
Max \quad & \sum_{t=1}^{T-1} h_t \left(x_1 - \sum_{j=1}^{t} d_t \right) + c_{t^*} \left(x_1 - \sum_{t=1}^{t^*-1} d_t \right) && (59)\\
s.t. \quad & d_t = \mu_t + \delta_t z_t, \quad 1 \leq t \leq t^* - 1, \\
& z_t \in [-1, 1], \quad 1 \leq t \leq t^* - 1, \\
& \sum_{j=1}^{t} |z_j| \leq \Gamma_t, \quad 1 \leq t \leq t^* - 2, \\
& \sum_{j=1}^{t^*-1} |z_{t^*-1}| \leq \Gamma_T - (T - t^* + 1), \\
& x_1 - \sum_{t=1}^{t^*-2} d_t \geq \sigma, \\
& x_1 - \sum_{t=1}^{t^*-1} d_t \leq \sigma.
\end{aligned}
$$

■

In total, case (b) amounts to T linear programs of type (59). In what follows, we assume that case (a) holds, and that furthermore the period t^e is chosen as small as possible.

Lemma 4.10 *Without loss of generality, there is at most one period t^f with $t^* \leq t^f \leq t^e$, such that $0 < |z_{t^f}^*| < 1$.*

Proof. If we have $t^* = t^e$ the result is clear, and if $t^* < t^e$ the result follows because the cost incurred in periods $t^*, \ldots, t^e$ is a convex function of the demands in those periods. ■

Given t^*, t^f and t^e, we partition the time periods into three sets:

$$B = \left\{1, 2, \ldots, t^* - 1, t^f\right\}, \tag{60}$$

$$M = \left\{t^* + 1, t^* + 2, \ldots, t^f - 1, t^f + 1, \ldots t^e\right\}, \tag{61}$$

$$F = \{t^e + 1, t^e + 2, \ldots, T\}. \tag{62}$$

Let $d^*(B), d^*(M)$ and $d^*(F)$ ($z^*(M), z^*(M)$ and $z^*(F)$, respectively) be the subvectors of d^* (resp., z^*) restricted to B, M and F. Below we will show that each of B, M and F gives rise to an optimization problem, for which $(d^*(B), z^*(B))$, $(d^*(M), z^*(M))$ and $(d^*(F), z^*(F))$ are respectively optimal. Thus, essentially, the adversarial problem is partitioned into three problems that can be solved (almost) independently. To ensure that the solutions to the three problems can be joined into a feasible solution to the adversarial problem, we will need to enumerate a polynomial number of boundary cases.

In what follows, we write $\gamma^* = \sum_{t=1}^{t^*-1} |z_t^*|$, and for any period t and $0 \leq \gamma$, write

$$l(\gamma, t) = \begin{cases} \Gamma_t - \lfloor \Gamma_t \rfloor - (\gamma - \lfloor \gamma \rfloor), & \text{if } \Gamma_t - \lfloor \Gamma_t \rfloor \geq \gamma - \lfloor \gamma \rfloor \\ 1 + \Gamma_t - \lfloor \Gamma_t \rfloor - (\gamma - \lfloor \gamma \rfloor), & \text{otherwise.} \end{cases} \tag{63}$$

In other words, $l(\gamma, t)$ equals the smallest nonnegative value that must be added to γ in order to obtain a quantity with *fractional* part equal to $\Gamma_t - \lfloor \Gamma_t \rfloor$. Note that $0 \leq l(\gamma, t) \leq 1$, and that our interpretation of $l(\gamma, t)$ is correct even if one or both of Γ_t and γ are integral.

Remark 4.11 $|z_{t^f}^*| = l(\gamma^*, t^e)$.

In the following sections 4.2.1, 4.2.2, 4.2.3 we we describe optimization problems arising from M, B and F that are solved by $(d^*(M), z^*(M))$, $(d^*(B), z^*(B))$, and $(d^*(F), z^*(F))$, respectively, assuming that there is a period t^f as in Lemma 4.10.

4.2.1 Handling M.

We consider first

$\boldsymbol{P_M(\gamma, t^*, t^f, t^e)}$ **:**

$$\max_{d,z} \quad \sum_{i \in M} (\mathcal{W}_i(\sigma - d_i) + c_{i+1} d_i)$$

$$s.t.$$

$$d_i = \mu_i + \delta_i z_i \qquad \forall i \in M \tag{64}$$

$$\sum_{j=t^*}^{i} |z_i| \leq \lfloor \Gamma_i - \gamma \rfloor \qquad t^* \leq i \leq t^f - 1 \tag{65}$$

$$\sum_{j=t^*}^{t^f-1} |z_i| \leq \lfloor \Gamma_{t^f} - (\gamma + l(\gamma, t^e)) \rfloor \tag{66}$$

$$\sum_{j=t^*}^{t^f-1} |z_i| + \sum_{j=t^f+1}^{i} |z_i| \leq \lfloor \Gamma_i - (\gamma + l(\gamma, t^e)) \rfloor, \qquad t^f + 1 \leq i \leq t^e, \tag{67}$$

$$-1 \leq z_i \leq 1 \qquad \forall i \in M,$$

Lemma 4.12 *$(d^*(M), z^*(M))$ is an optimal solution to $P_M(\gamma^*, t^*, t^f, t^e)$.*

Proof. First, $(d^*(M), z^*(M))$ is feasible for this problem. This follows by Remark 4.11 because in periods $i \in M$ we have $\sum_{j=t^*}^{i} |z_i^*| \leq \Gamma_i - \gamma^*$, and furthermore the $|z_i^*|$ are integral (0 or

1) by definition of M. Conversely, if $(\hat{d}(M), \hat{z}(M))$ is optimal solution to $P_M(\gamma^*, t^*, t^f, t^e)$, then $(d^*(B), \hat{d}(M), d^*(F))$ is a feasible solution to the adversarial problem, and the result follows. ■

Note that $P_M(\gamma, t^*, t^f, t^e)$ can be formulated as a mixed-integer program, much like (43). A subject for research would be to study under what conditions the linear programming relaxation of the formulation has an integral solution.

Later we will show how to solve $P_M(\gamma, t^*, t^f, t^e)$ in polynomial time. First, note that the right-hand sides of constraints (66) and (67) depend on $\lfloor\gamma\rfloor$, and not γ. Further, suppose we write $f_t = \Gamma_t - \lfloor\Gamma_t\rfloor$ for $t \in \{t^*, t^* + 1, \ldots, t^f - 1\} \cup \{t^e\}$, and let $f_{(1)}, f_{(2)}, \ldots, f_{(t^f - t^* + 1)}$ be the sorted values f_t. Also write $f_{(0)} = 0$ and $f_{(t^t - t^* + 2)} = 1$. Thus, if we fix $\lfloor\gamma\rfloor$, and fix some i, $0 \le i \le t^f - t^*$, then any two values γ with $f_{(i)} < \gamma - \lfloor\gamma\rfloor \le f_{(i+1)}$ will produce the same right-hand sides for constraints (65)-(67). Thus, for fixed $\lfloor\gamma\rfloor$, t^*, t^f, and t^e there are only $O(t^f - t^*)$ distinct problems $P_M(\gamma, t^*, t^f, t^e)$.

4.2.2 Handling B.

Next we turn to set B (c.f. (60)). Consider the optimization problem, for $0 \le k \le \Gamma_{t^*-1}$ and $0 \le j \le t^f - t^* + 1$:

$$\boldsymbol{P_B(t^*, t^f, t^e, k, j)}:$$

$$\begin{aligned}
\max_{d,z,y,\gamma} \quad & \sum_{i=1}^{t^*-2} h_i \left(x_0 - \sum_{h=1}^{i} d_h \right) + \mathcal{W}_{t^*-1} \left(x_1 - \sum_{h=1}^{t^*-1} d_h \right) + \\
& + c_{t^*} \left(\sigma - \left(x_0 - \sum_{h=1}^{t^*-1} d_h \right) \right) + \mathcal{W}_{t^f} \left(\sigma - d_{t^f} \right) + c_{t^f+1} d_{t^f} \\
s.t. \quad & x_0 - \sum_{h=1}^{i} d_h \ge \sigma \qquad \forall i \in \{1, 2, ..., t^* - 1\} \\
& x_0 - \sum_{h=1}^{t^*-1} d_h \le \sigma \\
& d_i = \mu_i + \delta_i z_i \qquad \forall i \in \{1, 2, ..., t^* - 1, t^f\} \\
& |z_i| \le y_i \le 1 \qquad \forall i \in \{1, 2, ..., t^* - 1, t^f\} \\
& \sum_{h=1}^{i} y_h \le \Gamma_i \qquad \forall i \in \{1, 2, ..., t^* - 1\} \\
& \sum_{h=1}^{t^*-1} y_h - \gamma = 0 \\
& k + f_{(j)} \le \gamma \le k + f_{(j+1)} \qquad (68) \\
& y_{t^f} = l(\gamma, t^e) \qquad (69)
\end{aligned}$$

This problem models the behavior of the adversary during those periods in B. Here, γ is the total uncertainty consumed in periods $1 \le t \le t^* - 1$. The first term in the objective is the inventory holding cost incurred in periods $1 \le i \le t^* - 2$, the second term is the inventory cost in period $t^* - 1$; while the last two terms describe the inventory cost during period t^f and the ordering cost in period $t^f + 1$. Constraint (69) controls how much risk the adversary can expend during period t^f. Also note that at optimality $y_t = |z_t|$ for each $t \in B$. The following result is clear, with a slight abuse of notation:

Lemma 4.13 *Suppose that $k + f_{(j)} \le \gamma^* \le k + f_{(j+1)}$ for integers $0 \le k \le \lfloor\Gamma_{t^*-1}\rfloor$ and $1 \le j \le t^f - t^* + 2$. Then, $(d^*(B), z^*(B), \gamma^*)$ solves $P_B(t^*, t^f, t^e, k, j)$.* ■

Next, we show how to replace $P_B(t^*, t^f, t^e, k, j)$ with at most four linear programs. First, since the objective of $P_B(t^*, t^f, t^e, k, j)$ contains just two functions $\mathcal{W}_t$, we can reduce the problem to at most four problems, each with a linear objective function and with the same constraints as $P_B(t^*, t^f, t^e, k, j)$. There remains the expression $l(\gamma, t^e)$ in the right-hand side of (69). We handle this as follows:

(i) Suppose $f_{(j)} \neq f_{t^e}$, and $f_{(j+1)} < 1$. Then, for every $\hat{\gamma} \in [k + f_{(j)}, k + f_{(j+1)}]$, we have that in the definition of $l(\hat{\gamma}, t^e)$ (see eq. (63)) the same case will always apply; and furthermore $\hat{\gamma} - \lfloor \hat{\gamma} \rfloor = \hat{\gamma} - k$. We conclude that $l(\gamma, t^e)$ is linear in γ.

(ii) Suppose now that j is such that $f_{(j)} = f_{t^e}$ and $f_{(j+1)} < 1$. Then we replace (69) with the constraint

$$y_{t^f} = 1 + f_{t^e} - (\gamma - k). \tag{70}$$

The right-hand side of (70) differs from $l(\gamma, t^e)$ only at $\gamma = k + f_{t^e}$ where it equals 1, whereas $l(k + f_{t^e}, t^e) = 0$. Denote the new optimization problem $\boldsymbol{L_B(t^*, t^f, t^e, k, j)}$. We claim that $(d^*(B), z^*(B), \gamma^*)$ solves $L_B(t^*, t^f, t^e, k, j)$. If not, in every optimal solution we must have $k + f_{(j)} = \gamma$ and $k + f_{(j)} < \gamma*$. Consequently, since the right-hand side of (70) is linear, then for $\epsilon > 0$ small enough we can find a feasible solution to $L_B(t^*, t^f, t^e, k, j)$ with $\gamma = k + f_{(j)} + \epsilon$ and with optimality error $O(\epsilon)$. Such a solution would be strictly better than $(d^*(B), z^*(B), \gamma^*)$ if ϵ is small enough; furthermore such a solution would be feasible for $P_B(t^*, t^f, t^e, k, j)$, contradicting Lemma 4.13.

As a final note for this case, suppose $k + f_{t^e} < \gamma^*$, and that we solve $L_B(t^*, t^f, t^e, k, j)$ and obtain an optimal solution $(\hat{d}, \hat{z}, \hat{\gamma})$ with $\hat{\gamma} = k + f_{t^e}$. This is a solution to one of the (up to) four linear programs corresponding to $L_B(t^*, t^f, t^e, k, j)$; thus, without loss of generality, the entire segment between $(\hat{d}, \hat{z}, \hat{\gamma})$ and $(d^*(B), z^*(B), \gamma^*)$ is made up of optimal solutions to $L_B(t^*, t^f, t^e, k, j)$. Hence, by performing a parametric simplex pivot we can obtain, from $(\hat{d}, \hat{z}, \hat{\gamma})$, an optimal solution that has a value of γ strictly larger than $k + f_{t^e}$.

(iii) Finally, assume now that $f_{(j+1)} = 1$. Note that $f_{t^e} < 1$, so $f_{t^e} \leq f_{(j)}$. In this case we again replace (69) with (70). In this case, the substitution is correct except when $\gamma = k+1$ (where $l(k + 1, t^e) = f_{t^e}$, whereas the right-hand side of (70) equals $1 + f_{t^e}$) and, if $f_{t^e} = f_{(j)}$ at $\gamma = k + f_{(j)}$. Note that $\gamma* < k + 1$ (since otherwise $z^*_{t^f}$ would be integral). An argument similar to that in case (ii) shows, again, that $(d^*(B), z^*(B))$ solves $L_B(t^*, t^f, t^e, k, j)$.

The above observations are summarized as follows:

Lemma 4.14 *Problem $P_B(t^*, t^f, t^e, k, j)$ is solved by $(d^*(B), z^*(B), \gamma^*)$ and it reduces to at most four linear programs.* ■

4.2.3 Handling F

Finally, we consider the set F of time periods. For $t < T$, define

$\boldsymbol{P_F(t)}$ **:**

$$\begin{aligned} \max \quad & \sum_{i=t}^{T} \mathcal{W}_i(\sigma - d_i) + \sum_{i=t+1}^{T+1} c_i d_{i-1} & & \\ s.t. \quad & d_i = \mu_i + \delta_i z_i & & t \leq i \leq T \\ & \sum_{j=t+1}^{i} |z_i| \leq \Gamma_i - \Gamma_{t-1} & & t \leq i \leq T \\ & -1 \leq z_i \leq 1 & & t \leq i \leq T \end{aligned} \tag{71}$$

The following is clear:

Lemma 4.15 *If $t^e < T$, $(d^*(F), z^*(F))$ solves $P_F(t^e + 1)$.* ■

4.2.4 The algorithm

Assuming first that there is a time period t^f as in Lemma 4.10, our algorithm examines every 5-tuple $(\bar{t}^*, \bar{t}^f, \bar{t}^e, k, j)$, where $1 \le \bar{t}^* \le \bar{t}^f \le \bar{t}^e \le T$, $0 \le k \le \Gamma_{\bar{t}^*-1}$ and $0 \le j \le \bar{t}^f - \bar{t}^* + 1$. For each such 5-tuple, we solve the three problems $P_B(\bar{t}^*, \bar{t}^f, \bar{t}^e, k, j)$, $P_M(\gamma, \bar{t}^*, \bar{t}^f, \bar{t}^e)$ (where we pick any γ with $k + f_{(j)} < \gamma < k + f_{(j+1)}$) and $P_F(\bar{t}^e + 1)$. By Lemmas 4.14, 4.13 and 4.15 (also see the remarks preceding Lemma 4.14), the solutions to at least one such triple of problems can be assembled into an optimal solution to the adversarial problem.

The case where there is no time period t^f as in Lemma 4.10 is handled in a similar way: here we amend problem P_B by removing the last two terms in the objective, and we amend problem P_M by removing constraints (66) and suitably modifying constraint (67).

In order to complete the description of the algorithm, we need to explain how to solve each problem P_M and P_F (we have already shown that the P_B reduce to linear programs).

A problem $P_M(\gamma, \bar{t}^*, \bar{t}^f, \bar{t}^e)$ can be solved using dynamic programming, with a stage for each time period t between $\bar{t}^*$ and $\bar{t}^e$, and a state corresponding to the risk budget consumed by period t. Rather than solving each problem separately, we can embed them into a smaller number of families. For example, given γ and $\bar{t}^f$ then the stages and states corresponding to periods between $\bar{t}^f$ and $\bar{t}^e$ are independent of $\bar{t}^*$, and the value of a state depends only on $\bar{t}^e$. Further improvements are possible [O06]. The procedure as described requires $O(T^4\,\Gamma_T)$ steps.

Next, consider a problem of type $P_F(t)$. Clearly, this is just the adversarial problem restricted to periods $t \le i \le T$, using the risk budgets $\Gamma_i - \Gamma_{t-1}$, and with starting inventory σ. Notice that this last condition implies that $\bar{t}^* = t$. Consequently, $P_F(t)$ reduces to a set of trivial (one period) problems P_B, and problems of type P_M. Hence, the up-front solution of all problems P_M as described in the previous paragraph can be used to quickly solve each problem $P_F(t)$.

Finally, a comment on the problems $L_B(\bar{t}^*, \bar{t}^f, \bar{t}^e, k, j)$. There is a total of $O(T^4\,\Gamma_T)$ such problems, and as discussed above, each such problem reduces to up to four linear programs. These linear programs should be warm-started, i.e. not solved from scratch. For example, parameter j only affects constraint (68); re-optimizing starting from the solution to the problem corresponding to $j+1$ (and all other parameters identical) will typically require a tiny number of pivots. Similarly with $\bar{t}^*$, $\bar{t}^f$, $\bar{t}^e$, and k. This detail, together with other implementational tricks, is important.

4.2.5 The approximate adversarial algorithm

In the discussion above we focused on solving the adversarial problem in Algorithm 1.1 exactly. Even though our algorithm runs in polynomial time, it is very conservative: it examines demand patterns that are unlikely to prove optimal except under extreme data conditions.

Thus, it is appealing to use a possibly suboptimal algorithm. The benefit of this would be that we would have much faster iterations, while, if the suboptimal algorithm were "smart" enough, we would still recap the benefit of updating the set $\tilde{D}$ in Algorithm 1.1 with demand patterns that fairly accurately approximate what the adversary can do. Of course, if we follow this approach, the quantity U computed in Step 2 of Algorithm 1.1 no longer qualifies as an upper bound to the min-max problem, though L certainly is a lower bound.

Hence, we can use the following approach: run Algorithm 1.1 as stated in its description, but using a suboptimal algorithm to handle the adversarial problem. Whenever $U - L$ is small, we run the exact adversarial algorithm, at which point the value of the adversarial problem does become a valid upper bound. This might allow us to terminate immediately if the gap is small. If not, we continue with the generic algorithm, once again using the suboptimal procedure to solve the adversarial problem. In theory, the exact algorithm should be run, for example, every k iterations for some k, but in our experience this was not needed.

The particular suboptimal algorithm we used was based on a simple idea. Our approach for the exact algorithm solved problems $P_M(\gamma, \bar{t}^*, \bar{t}^f, \bar{t}^e)$ and $P_B(\bar{t}^*, \bar{t}^f, \bar{t}^e, k, j)$ for all appropriate 5-tuples $(\bar{t}^*, \bar{t}^f, \bar{t}^e, k, j)$. In the suboptimal algorithm, instead, given $\bar{t}^*$ and $\bar{t}^e$ we compute a particular period to serve as $\bar{t}^f$. Recall (Remark 4.11) that the risk consumption at $\bar{t}^f$ should equal $l(\gamma, \bar{t}^e)$. Further, at period $\bar{t}^f$ inventory is already at or below basestock, and so the inventory cost at $\bar{t}^f$ will equal

$\mathcal{W}_{\bar{t}^f}(\sigma - d_{\bar{t}^f})$; by applying this formula with

$$d_{\bar{t}^f} \quad = \quad \mu_{\bar{t}^f} \pm l(\gamma, \bar{t}^e)\ \delta_{\bar{t}^f} \tag{72}$$

we compute the inventory cost at $\bar{t}^f$, assuming $t^f = \bar{t}^f$. On the other hand, if $t^f \neq \bar{t}^f$, the maximum inventory cost at $\bar{t}^f$ will be attained by

$$d_{\bar{t}^f} \quad = \quad \mu_{\bar{t}^f} \pm \delta_{\bar{t}^f}. \tag{73}$$

Our method picks that period $\bar{t}^f$ for which the decrease from (73) to (72) is minimum. Notice that by doing so we ignore the relation between the period $\bar{t}^f$ and problem $P_B(\bar{t}^*, \bar{t}^f, \bar{t}^e, k, j)$. However, the impact on optimality should be small. As we will see, this approximation dramatically speeds up the algorithm.

4.3 A bounding procedure for the risk budgets model

A simple observation is that the exact algorithm described above runs much faster when the Γ_t are integral. In the language of the previous sections, this follows from the fact that if the Γ_t are integral, then we must have $t^f = t^e$. This observation motivates the following approach:

1. Run the algorithm using risk budgets $\lceil \Gamma_t \rceil$. The value of this problem is an *upper bound* on the min-max problem with the original Γ_t (e.g., the adversary is more powerful).

2. Run the algorithm using risk budgets $\lfloor \Gamma_t \rfloor$. The value of this problem is a *lower bound* on the min-max problem with the original Γ_t, *and* the demand patterns produced by the adversary are valid.

In our testing, this scheme proved *extremely* effective, producing very tight bounds quite quickly. Clearly, we might obtain poor quality bounds in cases where T is small – but then the exact algorithm will be fast enough.

We integrate this procedure into the overall algorithm as follows. Consider the first instance where we would run the exact algorithm as indicated in Section 4.2.5 (e.g. when the lower bound L and the quantity U are close). Then, we run the bounding procedure instead of the exact algorithm. Optionally, if the upper bound proved by the procedure is close to the current lower bound, we can terminate.

4.4 The adversarial problem under the bursty demand model

For the reader's convenience, we restate the bursty demand model. Here, period t is either *normal*, meaning $d_t \in [\mu_t - \delta_t, \mu_t + \delta_t]$ (where $0 \leq \delta_t \leq \mu_t$ are given parameters), or it is *exceptional*, meaning $d_t = P_t$, where P_t is a given parameter. Further, in any set of W consecutive periods there is at most one exceptional period.

From a purely theoretical standpoint, we have the following result [O06]:

Theorem 4.16 *(a) The adversarial problem in the bursty demand model is NP-hard. (b) For every $\epsilon > 0$ a demand pattern of cost at least $(1 - \epsilon)$ times the optimum can be computed in time polynomial in T and $1/\epsilon$.* ■

This result is possibly of theoretical interest only, because it is not clear just how large T would be in a practical application. Nevertheless, the result does highlight that, most likely, a polynomial-time algorithm for the adversarial problem does not exist.

Our approach is as follows. For any demand pattern d, define the time period t^* as in Section 4: t^* is the earliest period such that the starting inventory is $\leq \sigma$. Then the maximum cost attainable by the adversary during periods 1 through $t^* - 1$, plus the order cost at period t^*, assuming that the last exceptional period is $t^* - k$ ($k = 1, \ldots, \min\{t^*, W\}$) is obtained by solving the following optimization problem:

$IP(t^*,k)$:

$$\max \quad \sum_{i=1}^{t^*-2} h_i \left(x_1 - \sum_{j=1}^{i} d_j \right) + \mathcal{W}_{t^*-1} \left(x_1 - \sum_{j=1}^{t^*-1} d_j \right) + c_{t^*}(\sigma - (x_1 - \sum_{j=1}^{t^*-1} d_j))$$

$$s.t. \quad x_1 - \sum_{j=1}^{t} d_j \geq \sigma \qquad 1 \leq t \leq t^* - 2 \tag{74}$$

$$x_1 - \sum_{j=1}^{t^*-1} d_j \leq \sigma \tag{75}$$

$$d_t = s_t + I_t P_t \qquad 1 \leq t \leq t^* - 1 \tag{76}$$

$$(1 - I_t)(\mu_t - \delta_t) \leq s_t \leq (1 - I_t)(\mu_t + \delta_t) \quad 1 \leq t \leq t^* - 1 \tag{77}$$

$$\sum_{i=t}^{t+W-1} I_t \leq 1 \qquad 1 \leq t \leq t^* - W \tag{78}$$

$$I_{t^*-k} = 1 \tag{79}$$

$$I_t \in \{0, 1\} \qquad 1 \leq t \leq t^* - 1$$

In this formulation, the $0-1$ variable I_t is used to indicate exceptional periods. If we set $k = 0$, and replace (79) with the constraints $I_t = 0$ for $t = 1, \ldots, \min\{t^*, W\}$, then we obtain the maximum cost attainable by the adversary assuming that there is no exceptional period among the W last.

We will return to problem $IP(t^*, k)$ below, but first we consider the second half of the problem. This can be done with a simple dynamic programming recursion. For $t = t^*, \ldots, T$ and $k = 0, 1, \ldots, \min\{t - t^*, W\}$, let $V_t(k)$ denote the maximum cost attainable by the adversary in periods $t, \ldots, T$ (not counting the ordering cost at t) assuming that the last exceptional period prior to t is period $t - k$ (with the same interpretation as before for $k = 0$). The recursion goes as follows: For $t = t^*, ..., T - 1$, we have

$$V_t(0) \quad = \quad \max_{d \in \{\mu_t - \delta_t, \mu_t + \delta_t, P_t\}} \{ \ \mathcal{W}_t(\sigma - d) \ + \ c_{t+1} d \ + \ V_{t+1}(I) \ \},$$

where we set $I = 1$ when we choose $d = P_t$, and otherwise we set $I = 0$. For $k = 1, \ldots, W - 1$ and $t < T$,

$$V_t(k) \quad = \quad \max_{d \in \{\mu_t - \delta_t, \mu_t + \delta_t\}} \{\mathcal{W}_t(\sigma - d) \ + \ c_{t+1} d \ + \ V_{t+1}(k + 1 \ (\mathrm{mod}\, W))\}.$$

For $t = T$, we set

$$V_T(0) \quad = \quad \max_{d \in \{\mu_T - \delta_T, \mu_T + \delta_T, P_t\}} \mathcal{W}_T(\sigma - d),$$

and for $k = 1, ..., W - 1$

$$V_T(k) \quad = \quad \max_{d \in \{\mu_T - \delta_T, \mu_T + \delta_T\}} \mathcal{W}_T(\sigma - d).$$

Clearly this recursion runs in polynomial time. Further, for each t and k we can put together a solution to $IP(t, k)$ and the optimizer for $V_t(k)$ to obtain a feasible solution to the adversarial problem, and the best such solution will clearly be the optimal solution. It is clear that the $V_t(k)$ can computed efficiently; now we return to the mixed-integer program $IP(t, k)$.

Consider the system made up of those constraints involving the $0-1$ variables I_t, namely (76), (77) and (78) (we do not include (79) since it just fixes a variable) plus the bounds $0 \leq I_t \leq 1$ for all t. It can be shown that this system defines an integral polyhedron (that is to say, a polyhedron each of whose extreme points has $0-1$ values on the I_t variables). This essentially is a known

fact; in particular constraints (78) describe a vertex-packing polyhedron on an interval graph (see [NW88] for background).

The consequence of this is that problem $IP(t^*, k)$, or, rather, each of the two linear objective problems obtained by considering the two cases for $\mathcal{W}_{t^*-1}$, is a mixed-integer programming problem over an integer polyhedron plus two side constraints (which do not involve the $0-1$ variables). We would thus expect $IP(t^*, k)$ to be easily solvable as a general mixed-integer program. And this proves to be exactly the case: using commercial software, even instances with T in the hundreds, the problem is solved in *hundredths* of a second.

5 Experiments with the basestock model

Our computational experiments are of two kinds. First, we want to study the convergence properties of the algorithms. Second, we want to investigate qualitative properties of the models studied in this paper.

5.1 The risk budgets model

In Table 5 we study the behavior of the exact algorithm and that of the bounding procedure described in Section 4.3. The column headed "Cost Gap" indicates the percentage error between the available upper and lower bounds when the early termination condition provided by the bounding procedure was tested. To produce the statistics in the table, for each data type we ran 150 randomly generated instances each with $T = 100$ time periods. Running times are in seconds. We see that, on average, the bounding procedure proves bounds with approximately a 1.7% gap. Another point to be stressed is that in both the exact algorithm and in the early termination version, the running time is dominated by the adversarial problem computations.

	Exact Algorithm			Early Termination					
	Running Time (sec.)			Running Time (sec.)			Cost Gap (%)		
	Average	Max	Min	Average	Max	Min	Average	Max	Min
Random	187.8	1362.62	1.35	13.91	58.89	2.02	1.52	12.22	0.09
Periodic	186.98	1659.17	2.83	11.41	56.25	1.56	1.42	8.71	0.00
Discounted	61.22	272.33	1.39	8.18	34.60	1.97	1.75	4.97	0.03

Table 5: Performance of algorithm for risk budgets ($T = 100$).

Table 5 may overstate the difference between the early termination solution and the optimal solution: in Table 6 we compare the early termination basestock with the optimal basestock level (same data as Table 5). We see that in most cases the early termination basestock indeed provides an excellent approximation to the optimum.

	% Error in Basestock		
	Average	Max	Min
Random	0.43	5.14	0.00
Periodic	0.42	8.44	0.00
Discounted	0.03	1.06	0.00

Table 6: Error in the basestock produced by using early termination.

Table 7 presents the running time of the algorithm for instances with integral budgets – this restriction is justified by the data above. For each data class we generated 100 examples. Note that even with 150 periods, our algorithm solves the problem very quickly. One fact that is worth noting is that in the discounted data case, on average, our algorithm converges to the optimum in fewer iterations than in the other cases.

		Running Time (sec.)			Number of Iterations		
	# periods	Average	Max	Min	Average	Max	Min
Random	75	5.56	35.20	0.16	4.79	16.00	2.00
	150	37.70	244.40	1.54	4.38	8.00	2.00
Periodic	75	3.85	25.93	0.16	4.04	14.00	2.00
	150	34.65	282.65	1.80	3.51	7.00	2.00
Discounted	75	2.71	80.14	0.11	2.96	6.00	2.00
	150	32.90	465.75	1.37	3.11	6.00	2.00

Table 7: Performance statistics – integral budgets

In Table 8 we compare the time spent solving adversarial problems to the total running time of our algorithm. Each problem category shows an average over 100 sample runs. This table clearly reinforces the idea that an adequate method for approximating the adversarial problem (perhaps by appropriately "sampling" demands) would yield a much faster overall algorithm; though of course the resulting algorithm might simply amount to a heuristic.

	# periods	**average**
Random	75	99.9737
	150	99.9934
Periodic	75	99.9711
	150	99.9977
Discounted	75	99.9765
	150	99.9999

Table 8: Ratio of adversarial time to total running time for the budgets model

In Table 9 we compare an optimal static policy, computed as in Section 2, with an optimal basestock policy (with constant basestock). To conduct these tests, given the optimal static policy, we computed its corresponding worst-case demand pattern and corresponding cost, which is reported in the column headed 'Static Policy'. The column headed 'Basestock policy' was computed in a similar way.

Example	**Static Policy**	**Basestock Policy**	**Error (%)**
1	10,115.00	12,242.17	-17.38
2	9,097.50	9,255.44	-1.71
3	172.94	175.83	-1.64
4	615,000.00	132,000.00	365.91
5	354,000.00	48,900.00	623.93
6	3,440,000.00	76,500.00	4396.73

Table 9: Static vs Basestock Policies

We see that for the first three examples the static policy performs better than the basestock policy. This is understandable: in these examples the uncertainty sets are either a single point or are very restricted. For such uncertainty sets, basestock policies impose an additional constraint on orders. However, for the last three examples, the basestock policy provides a significant gain which savings of up to 4396% in Example 6.

In the next set of experiments we compare the optimal basestock policy, run in a rolling horizon fashion, to the optimal static policy (Section 2), also run with a rolling horizon. We will refer to the latter approach as the *dynamic* policy.

In terms of the basestock model, a formal description is as follows. Let $\mu_t, \delta_t, \Gamma_t$, $t = 1, \ldots, T$ be given. Then, for $t = 1, \ldots, T$,

1. Let $\sigma^{(t)}$ be the optimum basestock computed by restricting the problem to periods $t, \ldots, T$. Then we order $u_t = \max\{0, \sigma^{(t)} - x_t\}$ at period t.

2. Compute the demand d_t by sampling from a normal distribution with mean μ_t and standard deviation $\delta_t/2$. If $d_t < 0$ we reset $d_t = 0$.

3. Set $x_{t+1} = x_t + u_t - d_t$.

4. Let $\bar{d}_t = d_t$. If $\bar{d}_t < \mu_t - \delta_t$, reset $\bar{d}_t \leftarrow \mu_t - \delta_t$. If $\bar{d}_t > \mu_t + \delta_t$, reset $\bar{d}_t \leftarrow \mu_t + \delta_t$. Let r_t be the largest multiple of 0.25 that is less than or equal to $|\bar{d}_t - \mu_t|/\delta_t$. Then we reset $\Gamma_k \leftarrow \Gamma_k - r_t$, for $k = t + 1, \ldots, T$.

The algorithm for the dynamic model is similar. In our experiments, we again consider the three different data types described in Section 3. For each type we ran 100 randomly generated examples with 50 time periods, and for each example we generated 200 sample paths (demand sets). In Table 10 we report the percentage increase in the average cost resulting from using the dynamic policy over using the basestock policy with rolling horizon. In the table, standard deviations are taken over the average cost of the 200 samples for each example. For completeness, we also report on the "pure" static policy, i.e. not run with a rolling horizon.

	Dynamic policy				Static policy			
	average	**stddev**	**min**	**max**	**average**	**stddev**	**min**	**max**
Random	-22.07	14.84	-49.03	17.91	831.99	249.64	388.37	1,744.73
Periodic	-8.22	54.92	-84.16	194.84	731.19	515.96	25.80	2,648.07
Discounted	-17.34	30.89	-72.31	82.09	606.18	274.49	87.35	1,220.91

Table 10: % increase in average cost of dynamic and static policies over the rolling horizon basestock policy

Notice that, on average, the dynamic policy outperforms the basestock policy with rolling horizon, though the standard deviation is quite high.

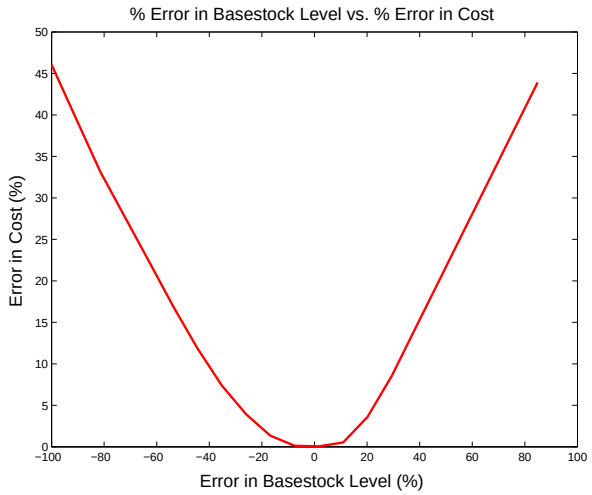

Figure 2: % error in basestock vs. % error in cost

Another issue of interest is to quantify the impact of an incorrect basestock choice. Figure 2 shows the percentage error in cost as a function of the percentage error in basestock value for a

particular example. For small values of error in basestock level, the cost curve is flat indicating that we can use near optimal basestock levels without sacrificing too much from optimality. This implies, for example, that even if numerical precision in an implementation of our algorithm were low, we would not be far from optimal. At the same time, Figure 2 shows that for large enough basestock error, the cost error grows linearly.

5.2 The bursty demand model

For each category shown in Table 11, 200 tests were performed. Data was generated using the same procedure as in Section 3. In addition, in most of these instances the window size was 15.

		Running Time (sec.)			Number of Iterations		
	# periods	**average**	**max**	**min**	**average**	**max**	**min**
Random	75	4.15	19	0.01	4.17	21	3
	150	35.96	228	0.01	6.52	31	4
	300	196.28	866	0.05	8.13	25	4
Periodic	75	4.48	26.5	0.05	4.22	22	3
	150	27.4	188	0.05	5.65	22	3
	300	240.28	1290.00	0.05	7.52	19	4
Discounted	75	3.8	13.3	0.09	2.66	20	3
	150	30.33	146	0.05	4.86	20	3
	300	166.0	869.00	0.05	6.7	21	4

Table 11: Behavior of algorithm for bursty demand model under a constant basestock

Table 12 describes experiments where we change the window size while keeping all other data constant, for a 300-period model in the periodic data case. We see that the number of iterations appears to grow quite slowly.

Window	5	10	15	20	25	30	35	40	45	50
Time	17.1	30.2	43.1	53.7	64.3	73.4	83.2	181.0	192.6	210.05
Iterations	7	7	7	7	7	7	7	11	11	11

Table 12: Impact of window size on a 300-period model

In Table 13, we investigate the impact of changing the initial inventory amount. Here we use the formula $x_1 = step \times \Phi/15$, where $step = 0, 1, 2, \ldots$ and Φ is a crude estimate of the total demand we would altogether see in the T periods – this assumes normal demands are at their mean values, and prorates the peaks. Note that there is no need to test cases with $x_1 < 0$ since they will behave in the same way as those with $x_1 = 0$. The examples in Table 13 all correspond to the same data set (other than x_1) with $T = 300$. When $x_1 > 14000$ the optimal basestock is always zero (and the algorithm takes 4 iterations). The results shown in this Table are quite interesting and are worth explanations. Essentially, what we see are two separate, but related, effects: the complexity of the problem, and the magnitude of the optimal basestock.

First, the larger x_1 is, the later that inventory will first fall below basestock (this is the parameter t^* discussed above). The higher this value is, the more uncertain the problem becomes, and thus, the more difficult. Consider, for example, the case with $x_1 = 12000$. This amounts to, very roughly, approximately 4/5ths of all the demand. So it will take, very roughly, on the order of 200 time periods for inventory to fall below basestock. This makes the decision maker's problem much more complex, than, say if we had t^* on the order of 10. For $x_1 \geq 14000$ we have a much easier problem because inventory never goes below basestock. The other effect we see in the table is that the optimal basestock is essentially constant (approximately equal to two or three periods' worth of demand, in a very crude sense), then grows rapidly, and then drops to zero – when the initial

x_1	**Time**	**Iterations**	**Optimal Basestock**
0	1.00e-02	7	89.39
1000	1.40e-01	8	88.91
2000	1.05e+00	8	89.22
3000	3.58e+00	8	89.06
4000	1.00e+01	8	88.91
5000	1.86e+01	7	89.56
6000	2.93e+01	8	88.91
7000	4.66e+01	7	89.42
8000	6.80e+01	7	88.66
9000	1.04e+02	7	89.05
10000	1.44e+02	7	89.15
11000	1.99e+02	7	88.47
12000	5.03e+02	8	90.78
13000	6.34e+02	12	339.33
14000	3.66e+02	4	0

Table 13: Impact of initial inventory

inventory is large enough no "safety" is needed. The sudden growth of the basestock at, or just before, the "critical" level of x_1 can be explained as follows: if x_1 is large enough the risk that inventory will go below basestock is low, until near the end of the planning horizon – so setting a larger basestock value is unlikely to have a negative effect (i.e., ordering costs) until near the end. However, for t near T the inventory could actually go negative, and a larger basestock will protect against that.

Another important issue is how the optimal robust basestock behaves as a function of the input data, and, in particular, as a function of how "large" the uncertainty sets are. Table 14 demonstrates an interesting phenomenon that is also observed in stochastic inventory theory (see [GKR05]). Here we have an example of the bursty demand model. Our experiment consisted in scaling the window size parameter and the magnitudes of the peaks by the same constant. Notice that by doing so we increase the variability in the system. To understand the intuition behind this suppose that $P_t = P$ for all t. Then, on average, the peak demand at any period in window of size w will be P/w, but the variance will be of the order of P^2/w. Hence, the "expected" demand per period will not change if we scale the window and peak size by the same constant, but demand variance will increase. Table 14 shows that as the variance of the demand increases, the optimal basestock level initially increases, but then it decreases and appears to converge to a constant.

Scale	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.6
Optimal Bs.	74.92	78.10	75.30	119.98	125.63	131.54	74.29	74.29	74.21

Table 14: Variance vs Optimal Basestock

We performed some additional tests to measure the sensitivity of the optimal basestock to problem data, in particular to the magnitude of the peaks P_t. In these tests we varied the problem data by scaling all peaks by the same scale constant, and keeping all other data constant. Figure 3 displays the result of such a test on a problem with $T = 75$, and window parameter $W = 5$.

Note that as the scale factor goes to zero the optimum basestock convergences to a constant. This is easy to understand, since when the peaks are small the adversary does not gain much from using or not using the peaks. When the scale factor is large enough, the optimum basestock also converges to a constant. At first glance this might appear incorrect: perhaps the optimum basestock should also increase, to offset potential large backlogging costs? However, this view is

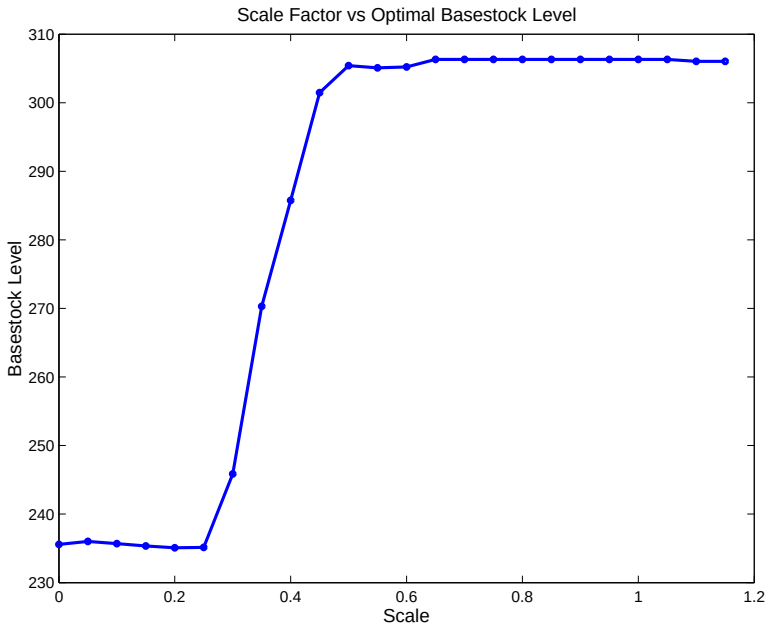

Figure 3: Effect of scaling peaks on optimum basestock

incorrect, because if we set the basestock large, then we have to carry large inventory in all periods.

6 Extensions

There are many possible extensions of the work described above which could prove fruitful. We will describe some problem areas for which we have theoretical results but which have not yet been implemented.

6.1 Polyhedral uncertainty sets

In this paper we considered two models of demand uncertainty. Both models are polyhedral (in the bursty demand case, we need additional variables). Certainly one could consider generalizing our Benders' decomposition approach to a general polyhedral set $\mathcal{D}$. In order to do so, we would need to develop a generic adversarial problem solver.

From a practical standpoint, what seems to make most sense to us would be to formulate the adversarial problem as a mixed-integer program, much like that in Section 2.1.2 but adapted to the basestock or safety stock case. Of course, we would also need to develop an algorithm (e.g. a branch-and-cut algorithm) for solving such a problem – this will be an area for future work.

From a purely theoretical standpoint, we can provide a negative result. We are somewhat uncomfortable with this result, primarily because "in practice" the number T would not be large. Nevertheless, the result is:

Theorem 6.1 *Let G be a graph with n vertices and clique number $\kappa(G)$. Then there is a robust basestock problem with n periods and polyhedral uncertainty set, whose value equals $\kappa(G)$.* ■

There *is* a practical consequence to this theorem. Namely, it is known that unless $P = NP$, there no polynomial-time algorithm that computes an approximation to clique number guaranteed to have constant multiplicative error. Hence, unless $P = NP$, there can not exist any polynomial-time algorithm that approximates the robust basestock problem within any constant factor.

6.2 Robust safety stocks

In the standard probabilistic setting, if the demand at period t has mean $\bar{\mu}_t$ and standard deviation $\bar{\delta}_t$, a *safety stock* policy is a basestock policy with basestock $\sigma_t = \bar{\mu}_t + \lambda \bar{\delta}_t$. We will refer to

the quantity λ as the *margin*. In this section we consider the problem of optimally choosing λ in a robust setting. In principle, the demand uncertainty models we have considered allow us to compute adequate stand-ins for $\bar{\mu}_t$ and $\bar{\delta}$; however, for greater generality, we will assume that we are given quantities $\hat{\mu}_t$ and $\hat{\delta}_t$ for each t, and we want to optimally set a policy with basestock $\hat{\mu}_t + \lambda\hat{\delta}_t$. Further, the problem could generalized by asking that $\lambda = \lambda_t$, but here we will restrict ourselves to the constant case (though, of course, in practice we would consider the rolling horizon implementation).

Formally, given parameters λ^l and λ^u, and an uncertainty set $\mathcal{D}$, we want to solve the problem:

$$\min_{\lambda^l \le \lambda \le \lambda^u} \quad F(\lambda) \tag{80}$$

where, given λ,

$$\begin{aligned} F(\lambda) \quad &= \quad \max_{d,x,u} \sum_{t=1}^{T} (\, c_t u_t \,+\, \mathcal{W}_t(x_{t+1})) && (81)\\ &\text{s.t.} \\ & \qquad u_t \,=\, \max\{\hat{\mu}_t + \lambda\hat{\delta}_t - x_t \,,\, 0\}, \quad 1 \le t \le T, && (82)\\ & \qquad x_{t+1} \,=\, x_t + u_t - d_t, \quad 1 \le t \le T, && (83)\\ & \qquad (d_1, d_2, \ldots, d_T) \,\in\, \mathcal{D}. && (84) \end{aligned}$$

To this end we propose the use of Algorithm 1.1. In this context, the decision maker's problem is that of choosing λ so as to minimize the maximum cost arising from the use of margin λ when the demand vector is one of a finite set of possible vectors. For a vector d of demands, let $cost(\lambda, d)$ be the cost that ensues when we use margin λ and the demands are d.

Theorem 6.2 *Let d be a fixed demand vector. Then $cost(\lambda, d)$ is piecewise convex with $O(T^2)$ pieces, each of which is piecewise-linear.*

Proof. First, let us add a "dummy" period, $T+1$, with $h_{T+1} = b_{T+1} = c_{T+1} = 0$, and with $\hat{\mu}_{T+1}$ and $\hat{\delta}_{T+1}$ chosen so that $\hat{\mu}_{T+1} + \lambda\hat{\delta}_{T+1} \ge x_1$ and $\hat{\mu}_{T+1} + \lambda\hat{\delta}_{T+1} \ge \hat{\mu}_t + \lambda\hat{\delta}_t$ for every $t \le T$ and every $\lambda \in [\lambda^l, \lambda^u]$. [remark: it is clear that it is always possible to find such parameters $\hat{\mu}_{T+1}$, $\hat{\delta}_{T+1}$; the choice guarantees guarantees that $x_{T+1} \le \hat{\mu}_{t+1} + \lambda\hat{\delta}_{T+1}$].

Let $1 \le t_1 < t_2 \le T+1$ be given. Consider the following system of equations:

$$\hat{\mu}_{t_1} + \lambda\hat{\delta}_{t_1} \,-\, \sum_{j=t_1}^{i} d_j \quad \ge \quad \hat{\mu}_{i+1} + \lambda\hat{\delta}_{i+1}, \quad t_1 \le i \le t_2 - 2, \tag{85}$$

$$\hat{\mu}_{t_1} + \lambda\hat{\delta}_{t_1} \,-\, \sum_{j=t_1}^{t_2-1} d_j \quad \le \quad \hat{\mu}_{t_2} + \lambda\hat{\delta}_{t_2}. \tag{86}$$

The interpretation of this system is as follows. Suppose that at time t_1 the starting inventory is at or below basestock. Then if λ satisfies (85) and (86), in periods $t_1+1, t_1+2, \ldots, t_2-1$ inventory will be above or equal basestock, but in period t_2 it will once again below or equal basestock. Note that there are constants $\alpha_L^1(t_1,t_2)$ and $\alpha_U^1(t_1,t_2)$ such that λ satisfies (85) and (86) if and only if $\lambda \in [\alpha_L^1(t_1,t_2),\, \alpha_U^1(t_1,t_2)]$ (possibly, the α are infinite).

Similarly, for $1 \le t < T+1$, consider the system:

$$\hat{\mu}_t + \lambda\hat{\delta}_t \,-\, d_t \quad \le \quad \hat{\mu}_{t+1} + \lambda\hat{\delta}_{t+1}, \tag{87}$$

with a similar interpretation, and defining an interval $[\alpha_L^2(t), \alpha_U^2(t)]$. Finally, for each $2 \le t \le T+1$, consider the system

$$x_1 \quad \ge \quad \hat{\mu}_1 + \lambda\hat{\delta}_1 \tag{88}$$

$$x_1 - \sum_{j=1}^{i} d_j \quad \ge \quad \hat{\mu}_i + \lambda\hat{\delta}_i \quad 1 \le i \le t-2, \tag{89}$$

$$x_1 - \sum_{j=1}^{t-1} d_j \quad \le \quad \hat{\mu}_t + \lambda\hat{\delta}_t, \tag{90}$$

again with a similar interpretation, and corresponding to some interval $[\alpha_L^3(t), \alpha_U^3(t)]$. Let N be the total number of (finite) distinct values α, and let $\beta_1 < \beta_2 < \ldots < \beta_N$ denote the sorted list of such values. Write $\beta_0 = -\infty$, $\beta_{N+1} = +\infty$. Then, it is clear that in each open interval (β_j, β_{j+1}) the behavior of the system is fixed. More precisely, if we take two distinct margin values λ_1, λ_2, both in (β_j, β_{j+1}), then inventory will be over (resp., under) basestock in exactly the same periods under policy λ_1 as under policy λ_2.

It follows that total cost (including ordering cost), as a function of λ, is convex piecewise-linear for $\lambda \in (\beta_j, \beta_{j+1})$. Since $N = O(T^2)$ the result is proved. ■

Corollary 6.3 *Given a finite set of demand patterns $\tilde{D}$, the decision maker's problem can be solved in polynomial time, by minimizing a piecewise convex function with $O(T^2\tilde{D})$ pieces.* ■

Now we turn to the adversarial problem, which we cast in the following context: we are given basestock levels $\sigma_1, \sigma_2, \ldots, \sigma_T$. We want to find a demand vector $d \in \mathcal{D}$ that maximizes the cost that would ensue from using the basestock policy $\{\sigma_t\}_t$. We expect that both the risk budgets model and the bursty demands model will prove computationally tractable in this general setting, even though they are likely NP-hard (we already know this for the bursty demands model). We will return to this point below. We do have a positive theoretical result for an uncertainty model that amounts to a discrete version of the risk budgets model.

The model is described as follows:

(a) We are given a quantity $0 \leq \mu_t$ for $1 \leq t \leq T$. This is the nominal demand at time t.

(b) We are given quantities $\alpha_i \geq 0$, for $1 \leq i \leq C$ (some C).

(c) Each time period t belongs to a *category* i, $1 \leq i \leq C$. If period t is in category i, then the demand at t will be of the form

$$d_t \;=\; \mu_t \,+\, k_t\, \alpha_i, \tag{91}$$

where k_t is an integer, possibly negative. We have bounds $l^t \leq k^t \leq u^t$ for given l^t, u^t.

(d) For each t we have a constraint of the form $\sum_{j=1}^{t} |k_j| \leq \Gamma_t$, where $0 \leq \Gamma_t$ is a given integer.

Comments. We may view each category as describing a variance "class" or "type". The weakness of (91) is, of course, that ideally, we would have instead a mechanism of the form $d_t = \mu_t + k_t\, \alpha_{i,t}$. But, for example, in a setting with *seasonal* demands, the above model should prove functional. We will refer to the model as the *discrete budgets* model. One can prove the following result, which is primarily of theoretical relevance.

Theorem 6.4 *Given a basestock policy with levels $\{\sigma_t\}_t$, the adversarial problem under a discrete budgets model can be solved in polynomial time, for each fixed C.* ■

We say that this result is only of theoretical importance, because the adversarial problem appears to be "knapsack-like", and hence an algorithm such as the one we described for the bursty demands model in Section 4.4 would prove far more practical (though nominally of exponential since it solves mixed-integer programs). If anything, the fact that Theorem 6.4 can be proved would justify such an approach. Next we list generalizations of the discrete budgets model that are also likely to be numerically tractable:

- A discrete model with demands of the form $d_t = \mu_t + k_t\, \alpha_{i,t}$, where k_t is integral, and we have budgets of the form $\sum_{j=1}^{t} |k_j| \leq \Gamma_t$ (as discussed above).

- A model where, for every t, we are given μ_t, and quantities $0 < \alpha_t^0 < \alpha_t^1 < \ldots < \alpha_t^{U(t)}$, and quantities $0 < \beta_t^0 < \beta_t^1 < \ldots < \beta_t^{L(t)}$, for some $U(t)$ and $L(t)$. At time t, the adversary

chooses an integer k_t, with either $0 < k_t \leq U(t)$, or $0 < -k_t \leq L(t)$, or $k_t = 0$. If $k_t > 0$, then demand will be in the interval $(\mu_t + \alpha_t^{k_t-1}, \mu_t + \alpha_t^{k_t}]$. If $k_t < 0$, then demand will be in the interval $[\mu_t - \beta_t^{-k_t}, \mu_t - \beta_t^{-k_t-1})$. If the adversary chooses 0, then demand lies in $[\mu_t - \beta_t^0, \mu_t + \alpha_t^0]$. Once more we have budgets $\sum_{j=1}^{t} k_j \leq \Gamma_t$.

In all the above models we were using budget constraints motivated by the original risk budgets model in [BT05]. But there is a generalization, which we call the *intervals* model, which still proves tractable. We explain this generalization in the context of the discrete budgets model, but it is easily extended to the original (continuous) risk budgets setting. First, we are given quantities $\alpha_i \geq 0$, for $1 \leq i \leq C$ (some C), and demand at time t is of the form $d_t = \mu_t + k_t \alpha_i$ as for the discrete budgets model. In addition, we are given

- A family $\mathcal{I}$ of closed intervals of $\{1, 2, \ldots, T\}$, and a nonnegative integer Γ_I for each $I \in \mathcal{I}$.
- We must satisfy $\sum_{t \in I} |k_t| \leq \Gamma_I$, for each $I \in \mathcal{I}$.

We will say that an interval model with family $\mathcal{I}$ is of *width* $\leq \omega$, if there exist intervals $[i_1, j_1]$, $\ldots, [i_m, j_m]$, of $\{1, 2, \ldots, T\}$ such that:

(i) $\mathcal{I}$ is the set of all intervals of the form $[i_k, h]$ for some $1 \leq k \leq m$ and $i_k \leq h \leq j_k$, and

(ii) For every $1 \leq t \leq T$ there are at most ω intervals $[i_k, j_k]$ with $i_k \leq t \leq j_k$.

We say that the model has width ω if the condition in (ii) holds with equality for some t. Note that a model of width $\omega = 2$ can be made substantially more restrictive on the adversary than one with $\omega = 1$. The following result can be proved, again primarily of theoretical relevance:

Theorem 6.5 *Suppose we have a basestock policy with levels $\{\sigma_t\}_t$. Suppose we have an interval model of width $\leq \omega$, and with C variance types. Then, for each fixed ω and C, the adversarial problem can be solved in polynomial time*

6.3 Ambiguous uncertainty sets

In the ambiguous setting demands are known to follow some probabilistic distribution, but the decision maker only has limited information regarding the distribution (see [HK03] for related, though different, work). For example, we might know that the distribution (at any time t) is log-normal, and we might know its mean value, but might only know an upper bound on the variance. Conceptually, we may think of the adversary as choosing a probability distribution on the demands that is consistent with the available information. This yields a problem of the form:

$$\min_{\pi \in \Pi} \max_{P \in \mathcal{P}} \mathbf{E}\, cost(\pi|P).$$

Here, Π is the set of available policies, $\mathcal{P}$ is the set of possible probability distributions, and $\mathbf{E}\, cost(\pi|P)$ is the expected cost, under policy π when the demands follow distribution P. To solve this problem, we can again resort to a generic Benders' decomposition method obtained by suitably modifying Algorithm 1.1.

Here we focus on basestock policies. For details and complete proofs, see [O06].

Lemma 6.6 *Let σ be a given basestock and let P be a probability distribution. (a) If $x_1 \leq 0$, then $\mathbf{E}\, cost(\sigma|P)$ is a convex function of σ. (b) If P is a discrete distribution, then $\mathbf{E}\, cost(\sigma|P)$ is piecewise convex.* ■

In part (b) of the above Lemma, and in the results we describe next, we focus on a particular model of ambiguous uncertainty, namely the *discrete distribution* model. Here

- we are given fixed values $v^1, v^2, \ldots, v^K$,

- The adversary will choose probabilities p^k, $1 \leq k \leq K$, such that in any time period demand equals v^k with probability p_k.
- It is assumed that the adversary is constrained in that the vector $(p^1, \ldots, p^k)$ must belong to some set $\mathcal{P}$ (for example, polyhedral).

Lemma 6.7 *Let $P^1, P^2, \ldots, P^J$ be given discrete probability distributions. Suppose that $x_1 \leq 0$. Then we can compute, in polynomial time, a basestock σ that minimizes $\max_{1 \leq j \leq J} \mathbf{E}\, cost(\sigma | P^j)$.*

Proof. Consider a fixed distribution P^j. Note that since $x_1 \leq 0$, the expected inventory cost paid in every period is the same. In fact, the expected ordering cost paid in any period, other than period 1, is the same (and in period 1 it equals $\sigma - x_1$). Further if demand equals $v^k \geq \sigma$ then we incur a backlogging cost, otherwise we incur an inventory holding cost. Hence, we can write a closed form expression on $\mathbf{E}\, cost(\sigma | P^j)$, which will be piecewise linear in σ (and is convex because of Lemma 6.6). The result follows. ■

Comment. The assumption $x_1 \leq 0$ is significant. Without the assumption we can prove a result similar to Lemma 6.7, except that the resulting algorithm will be exponential in K.

Lemma 6.8 *Let σ be given, and assume $x_1 \leq 0$. Then the adversarial problem reduces to maximizing a* **linear** *function of $(p^1, \ldots, p^k)$ over $(p^1, \ldots, p^k) \in \mathcal{P}$.* ■

In future work we plan to report on computational experience with ambiguous problems. A variation worth noting is the rolling horizon model. But here we can expect that the decision maker *learns* from the past behavior of the adversary; i.e. the knowledge of the set $\mathcal{P}$ is sharpened. To be fair, one should then consider a *gaming* setting where the adversary can respond in kind.

6.4 Model superposition

Suppose we have a number of demand uncertainty sets $\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^K$, and for each $k = 1, \ldots, K$, cost vectors h^k, b^k, c^k (e.g. $h^k = (h_1^k, h_2^k, \ldots, h_T^k$ and similarly with b^k and c^k). Let Π be a set of available policies. We are interested in a problem of the form:

$$\min_{\pi \in \Pi} \max_{k} \max_{d \in \mathcal{D}^k} cost^k(\pi, d), \tag{92}$$

(where $cost^k$ is the cost under the k^{th} cost function), or

$$\min_{\pi \in \Pi} \sum_{k} \max_{d \in \mathcal{D}^k} cost^k(\pi, d). \tag{93}$$

As a generalization, we could have that some of the $\mathcal{D}^k$ are ambiguous problems, or even stochastic programs. In all these cases our generalized Benders' methodology applies. We are motivated to study these problems for a number of reasons:

- Consider a case where $\mathcal{D}^1 \subseteq \mathcal{D}^2 \subseteq \ldots \subseteq \mathcal{D}^K$ and there are vectors h, b, c and nonnegative constants $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_K$, such that $(h^k, b^k, c^k) = \alpha_k\ (h, b, c)$. In other words, each uncertainty set $\mathcal{D}^k$ is more conservative than $\mathcal{D}^{k-1}$, but its cost "counts less". Problems (92) and (93) are an attempt to protect against potentially very poorly behaved data (demands) without becoming excessively conservative. A special example of this case is that where the $\mathcal{D}^k$ all have the same structural properties (for example, they are all box-constrained in each time period).
- On the other hand, if the sets $\mathcal{D}^k$ are extremely different and/or so are the cost functions, then we are simply trying to protect against multiple forms of data uncertainty, again while trying to avoid becoming too conservative.
- Finally, we conjecture that using relatively simple sets $\mathcal{D}^k$, and not very large K, one should be able to approximate arbitrarily complex uncertainty sets $\mathcal{D}$, so that e.g. solving (92) becomes more tractable than solving (30).

6.5 More comprehensive supply-chain models

The algorithms described in this paper addressed single-buffer, zero-leadtime problems. In a more realistic setting we should consider positive leadtime systems (e.g. an order placed at time t does not arrive until time $t + k$). Multiple buffers arise, for example, in assembly systems, where a product is obtained by assembling several components, each of which is itself made of (sub)components, and so on. Here each component has its own buffer and material flows from buffer to buffer on a network in a prescribed sequence. Problems of this type with certain data can be solved under appropriate assumptions (for example, involving coordination); in a robust setting the problem is bound to be more complex, but it should be possible to adapt our techniques so as to (at least) efficiently compute local optima.

References

[AAFKLL96] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton and Z. Liu, Universal stability results for greedy contention-resolution protocols, *Proceedings of the 37th Annual Symposium on Foundations of Computer Science* (1996), Burlington, VT, 380 – 389.

[AZ05] A. Atamtürk and M. Zhang, Two-Stage Robust Network Flow and Design under Demand Uncertainty, Research Report BCOL.04.03, Dept. of IEOR, University of California at Berkeley (2004).

[B62] Benders, J.F., Partitioning procedures for solving mixed variables programming problems, *Numerische Mathematik* **4** (1962) 238-252.

[BGGN04] A Ben-Tal, A. Goryashko, E Guslitzer and A. Nemirovski, Adjusting robust solutions of uncertain linear programs, *Mathematical Programming* **99**(2) (2004), 351 – 376.

[BGNV05] A. Ben-Tal, B. Golany, A. Nemirovski and J.-P. Vial, Retailer-Supplier Flexible Commitments Contracts: A Robust Optimization Approach. *MSOM* **7** (2005), 248-271.

[BN98] A. Ben-Tal and A. Nemirovski, Robust convex optimization, *Mathematics of Operations Research* **23** (1998), 769 – 805.

[BN99] A. Ben-Tal and A. Nemirovski, Robust solutions of uncertain linear programs, *Operations Research Letters* **25** (1999), 1-13.

[BN00] A. Ben-Tal and A. Nemirovski, Robust solutions of linear programming problems contaminated with uncertain data, *Mathematical Programming Series A* **88** (2000), 411 – 424.

[BS03] D. Bertsimas and M. Sim, Price of robustness, *Operations Research* **52** (2003), 35 – 53.

[BT05] D.Bertsimas, A.Thiele, A robust optimization approach to inventory theory, to appear *Oper. Research.*

[BKRSW96] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan and D. P. Williamson, Adversarial queueing theory, *Proceedings of the 28th Annual ACM Symposium on Theory of Computing* (1996), 376 – 385.

[CS60] A. Clark and H. Scarf, Optimal policies for a multi-echelon inventory problem, *Management Science* **6** (1960), 475 – 490.

[C91] R. L. Cruz, A calculus for network delay, Part I: Network elements in isolation, *IEEE Transactions on Information Theory*, **37** (1991), 114 – 131.

[E84] R. Ehrhart, (s,S) policies for a dynamic inventory model with stochastic leadtimes, *Operations Research* **32** (1984), 121 – 132.

[FZ84] A. Federgruen and P. Zipkin, An efficient algorithm for computing optimal (s,S) policies, *Operations Research* **32** (1984) 1268 – 1286.

[GKR05] G. Gallego, K. Katircioglu and B. Ramachandran, A Note on The Inventory Management of High Risk Items (2005). To appear, *O. R. Letters*.

[GM93] G. Gallego and I. Moon, The distribution free newsboy problem: Review and extensions, *Journal of Operations Research Society* **44** (1993), 825 – 834.

[GRS01] G. Gallego, J. Ryan and D. Simchi-Levi, Minimax analysis for finite horizon inventory models, *IIE Transactions* **33** (2001), 861 – 874.

[GLS93] M. Grötschel, L. Lovász and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag. 1993.

[HK03] A. Heching and A. King, Supplier Managed Inventory for Custom Manufactured Items with Long Lead Times, manuscript (2003).

[I63a] D. Iglehart, Dynamic programming and stationary analysis in inventory problems, *Multistage Inventory Models and Techniques* (Chapter 1) (1963), Stanford University, Stanford, CA.

[I63b] D. Iglehart, Optimality of (s,S) policies in infinite horizon dynamic inventory problem, *Management Science* **9** (1963), 259 – 267.

[I05] G. Iyengar, Robust Dynamic Programming, *Math. of OR* **30** (2005), 257 – 280.

[MG94] I. Moon and G. Gallego, Distribution free procedures for some inventory models, *Journal of Operations Research Society* **45** (1994), 651 – 658.

[MT01] A. Muharremoglu and J. Tsitsiklis, A single unit decomposition approach to multi-echelon inventory systems, Working paper (2001).

[NW88] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, New York (1988).

[O06] N. Özbay, Ph.D. Thesis, Columbia University (2006).

[S58] H. Scarf, A min-max solution of an inventory problem, *Studies in the Mathematical Theory of Inventory and Production* (Chapter 12) (1958), Stanford University Press, Stanford, CA.

[T05] A.Thiele, Robust dynamic optimization: a distribution-free approach. Manuscript (2005).

[V66] A. Veinott, On the optimality of (s,S) inventory policies: New conditions and a new proof, *SIAM Journal on Applied Mathematics* **14** (1966), 1067 – 1083.

[Z00] P. Zipkin, Foundations of inventory management (2000), McGraw-Hill Higher Education.