Solving Robust Inventory Problems

by

Nuri Sercan Özbay

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2006

ABSTRACT

Solving Robust Inventory Problems

In this work we consider setting the optimal inventory control policies for a single buffer when demand is uncertain, in a robust framework. Unlike traditional inventory models we do not assume that the demand is random with a known distribution. Instead, demand can take values from a given uncertainty set. Our objective is to find the policy that minimize the maximum cost that is attainable by the demand vectors in our uncertainty set. We consider the problem for two different types of policies which are very common in practice and present a family of algorithms based on decomposition that scale well to problems with hundreds of time periods. We also present theoretical results on more general models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Designing an efficient supply chain and operating it efficiently is one of the most important issues for a large number of modern organizations. For the past three decades, increasing economic and competitive pressure has forced manufacturers to create better ways to control every single step in their supply chain, from supplier contracts to distribution channels. Due to recent fluctuations in the economy, matching supply to demand has become even more challenging. Nowadays, there is a growing need for more robust supply chains that are responsive to the changes in market conditions. This work was inspired by a project carried out with an industrial partner and it concerns the optimal stock ordering policies for a buffer in a supply chain in an uncertain environment.

1.1 Literature review

The origins of Supply Chain/Inventory Management can be found as early as the beginning of 20th century when Harris [H13] derived the Economic-Order-Quantity formula that applies when the demand is assumed to be constant over time. Over the few decades preceding his work numerous authors elaborated different variations of Harris' EOQ model. Although pioneers of the field were aware of the uncertainties associated with the problem, the study of the problem in a stochastic setting was only started in the 1950's, with the seminal papers by Arrow, Harris and Marschak [AHM51] and Dworetzky, Kiefer and Wolfowitz [DKW52]. Since then, supply chain optimization problems have been studied extensively under stochastic settings using different methodologies such as dynamic programming and stationary analysis. We refer the reader to Zipkin [Z00] for a comprehensive discussion of various models in supply chain/inventory management.

One of the most important advancements in supply chain management took place when Clark and Scarf [CS60] proved the optimality of basestock policies for serial systems using dynamic programming, a powerful technique that would later be used by many other authors to derive structural results about optimal inventory control policies. Subsequently, basestock policies became increasingly popular and were proved to be optimal for many other inventory models. For further work on basestock policies see Iglehart [I63a, I63b], Veinott [V66], Ehrhart [E84] and Muharremoglu and

Tsitsiklis [MT01]. While such a policy is not necessarily optimal, it may be preferable over optimal policies since it is easy to implement and often performs very well. Due to its simplicity it is widely used in practice and finding optimal basestock levels itself has drawn a lot of attention by both practitioners and researchers.

Traditional models for supply chain management are often criticized by practitioners for their strong assumptions, among which is full knowledge of the underlying demand distribution. In most real world applications, especially in industries with short product life cycles, paucity of historical demand data makes it very hard to determine a demand distribution that fits the observed data. In such situations the inventory controller should make decisions using partial information, such as inaccurate forecasts, about future demand, and estimates for the error in these forecasts.

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 given first and second moments. Later, Gallego and Moon [GM93, MG94] provided concise derivations of his results and extended it to other cases. Gallego, Ryan and Simchi-Levi [GRS01] considered the multi-period version of this problem with discrete demand distribution and proved the optimality of basestock policies. Recently, Bertsimas and Thiele [BT04] and Ben-Tal et. al. [BGNV05] studied some supply chain management problems with limited demand information using the robust optimization framework. The central difference of their work from previous work is that instead of assuming partial information about the distribution of the demand, they assume that uncertain demand is explicitly represented by a set that defines all possible demand values. In Chapter 2 we provide a detailed discussion of their results. Also see [BGGN04] and [T05].

Robust Optimization is an increasingly accepted way to handle uncertainty. It addresses parameter uncertainties in deterministic optimization problems. Unlike Stochastic Programming it does not assume that the uncertain parameters are random variables with known distributions, instead it represents uncertainty in parameters using deterministic uncertainty sets in which all possible values of these parameters reside. Typically, Robust Optimization adopts a min-max approach that guarantees the feasibility of the obtained solution for all possible values of the uncertain parameters in the given uncertainty sets.

Although the underlying ideas are 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 a large amount of research dealing with various aspects of Robust Optimization. For example, Bertsimas and Sim [BS03] proposed 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.

Robust Optimization methodology was originally developed to deal with static problems in which all of the decision variables are set prior to resolution of any uncertainty. However, in most real life applications, the dynamic nature of the problems allows decision makers to revise their decisions as more information about the uncertain parameters becomes available. This is especially true for multi-period decision making problems where static robust optimization models are unable to capture the fact that the decision maker knows the values of uncertain parameters in the preceding periods and can exploit this information when making his decisions. Recognizing the need for incorporating the dynamic nature of the multi-period decision making problems into Robust Optimization models, Ben-Tal et. al. [BGGN04], recently proposed Affinely Adjustable Robust Counterpart models which feature the idea of dynamically determining decision variables as affine functions of the portion of the uncertain data that has been realized. By restricting the decisions to affine functions of the past data they managed to produce tractable formulations for uncertain Linear Programs. Ben-Tal et. al. [BGNV05] applied these ideas to a supply chain management problem to get a polynomial time solvable formulation. We will review this work more closely in Section 2.1.

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

1.2 Our model and contributions

In this thesis we develop procedures for setting the stock ordering policies for a buffer in a supply chain subject to uncertainty in the demands. As mentioned before, our work is motivated by experience with an industrial partner in the electronics industry who faced the following difficulties: short product cycles, a complex supply chain with multiple suppliers and long production leadtimes, a paucity of demand data and a very competitive environment. The combination of these factors produced a significant exposure to risk, in the form of either excessive inventory or shortages. The supply chain of our industrial partner consisted of a network with multiple buffers; however, in this work we consider a system made up of a single buffer.

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 thesis.

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".

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 an 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, for each time t we need to describe a decision rule, such that the decision maker observes the current state of the system (e.g. the current inventory x_t) and prior actions on the part of the adversary, and chooses u_t appropriately. A policy is the the sequence of such decision rules and we denote the set of all available policies by Π. Typical examples of policies are the *basestock policy* in which our decision rule in each period is given by the function $\max\{\sigma_t - x_t\}$ for a given *basestock level* σ_t and the *static policy* in which the decision maker determines the orders in advance independent of the state of the system or actions of the adversary.

The main focus of this thesis concerns how to pick the optimal policy in the robust setting, under various demand uncertainty sets $\mathcal{D}$. In the succeeding three chapters we consider the "basic inventory problem" for three different variants of Π. We propose a generic methodology and using this methodology we develop algorithms to compute the optimal stock ordering policies.

The inventory problem in the robust setting can be described as follows:

$$\min_{\pi\in\Pi} \ \max_{d\in\mathcal{D}} \ cost(\pi, d), \tag{1.1}$$

where for $\pi \in \Pi$ and $d \in \mathcal{D}$,

$$cost(\pi, d) = \sum_{t=1}^{T} \left(\, c_t u_t(\pi, d) \ + \ \max\{ \, h_t x_{t+1}(\pi, d) \, , \ -b_t x_{t+1}(\pi, d)\} \, \right) \tag{1.2}$$

where $u_t(\pi, d)$ denotes the order that would be placed by policy π at time t under demands $d_1, d_2, \ldots, d_{t-1}$, and $x_t(\pi, d)$ would likewise denote the inventory at the start of period t. Here, the quantity x_1 (the initial inventory level) is an input and once the demand variables $(d_1, d_2, \ldots, d_{t-1}) \in \mathcal{D}$ and the policy π have been chosen, u_t and x_t are uniquely determined, for $1 \leq t \leq T$. Notice that $cost(\pi, d)$ is the cost corresponding to policy π and demand pattern d; and worst-case the cost arising from applying policy π is given by

$$\max_{d \in \mathcal{D}} \; cost(\pi, d). \tag{1.3}$$

We call (1.3) the *adversarial problem.*

In Section 1.2.1 we discuss a generic algorithm for solving (1.1) for different types of policies and demand uncertainty sets (for different sets Π and $\mathcal{D}$). The algorithm is based on a common approach, Benders' decomposition [B62], and extensive experimentation shows it to be quite fast for the cases that are considered in this thesis. Although we consider a specific inventory management problem, our methodology is general and can be used to solve many other minmax type problems.

In Chapter 2, we limit our policy space to static policies. We assume that in each period t, the order quantity is determined in advance and fixed regardless of the state of the system, i.e. our decision rule is defined by a constant function of the state of the system and the past actions by the adversary. We develop an algorithm to compute the optimal static policy. We numerically prove the efficiency of our algorithm by testing it on many large examples.

The static policy does not allow the inventory controller to dynamically use the information that becomes available as the uncertainty in the system is resolved. However, this is unrealistic since in real world applications the decision maker can make dynamic decisions. In Chapter 3 we consider the basestock policies as a tool to incorporate the dynamic nature of the problem into our methodology. We construct our policy space with constant basestock policies, i.e. policies such that for $1 \leq t \leq T$ and for a real constant σ, the order quantity in period t is equal to $\max\{\sigma - x_t, 0\}$ where x_t denotes the inventory on-hand at the beginning of period t. Notice that when using a basestock policy, the inventory controller determines an order-up-to level and if the on-hand inventory is less that that level he places an order to push it back up to its ideal level. Although basestock policies have their own limitations, the effect of uncertainty on inventory levels (therefore inventory cost) is not as severe as under a static policy because of the cap it places on inventory level. In Chapter 3 we present a numerical comparison of optimal basestock policies with optimal static policies.

Part of the reason for our focus on basestock policies is that they have acquired very wide use and can be shown to be optimal under stochastic inventory models. Further, though basestock policies may not always be optimal, they are viewed as producing easily implementable policies for practitioners. In Chapter 3 we propose an algorithm to pick the optimal basestock level for an inventory buffer under several robust uncertainty models. Extensive experimentation shows that our algorithm proves to be very efficient.

At the other extreme we may consider making decisions dynamically. Instead of determining the policy at the very beginning of the horizon, the inventory controller can delay the ordering decision in each period t until the beginning of period t, which makes it possible to use all of the information that becomes available by period t. Naturally, such a policy performs very well under uncertainty since it gives the inventory controller greater freedom to set the orders. We call this problem the dynamic problem, and in Chapter 4 we give a characterization of the optimal policies and show how to compute them.

1.2.1 Generic algorithm

Our generic algorithm, given next, maintains a *working list* $\tilde{D}$ of demand patterns – each member of $\tilde{D}$ is a demand vector $(d_1, d_2, \ldots, d_T) \in \mathcal{D}$. The algorithm also maintains an upper bound U and a lower bound L on the value of problem (1.1). This algorithm can be viewed as a form of Bender's decomposition. [B62]

Note that the decision maker's problem is of the same general form as the generic problem (1.1) – 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 experimental testing, 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.

```
Algorithm 1.2.1 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 $U \leftarrow \min \left\{ U \, , \, cost(\tilde{\pi}, \bar{d}) \right\}$.

    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.
```

On the other hand, the adversarial problem is non-convex. In at least one case we can show that it is NP-hard. 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 1.2.1 is that neither Step 1 nor Step 2 need be carried out exactly, except in 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 $\tilde{D}$. Our implementations run the exact adversarial problem only at certain iterations, as will be discussed in Chapter 3.

1.3 Notation

Notation 1.3.1 *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.

Chapter 2

The Static Problem

In this chapter we consider the *static* robust inventory problem, which is defined by:

$$\min_{u \geq 0} \sum_{t=1}^{T} c_t u_t \quad + \quad K(u) \tag{2.1}$$

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

$$\begin{aligned} K(u) \quad = \quad \max \quad & \sum_{t=1}^{T} W_t(x_{t+1}) \\ \text{s.t.} \qquad & x_{t+1} \; = \; x_t + u_t - d_t, \quad 1 \leq t \leq T, \\ & (d_1, d_2, \ldots, d_T) \; \in \; \mathcal{D}. \end{aligned} \tag{2.2}$$

Here (2.2) is the adversarial problem: given orders u, the adversary chooses demands d so as to maximize the total inventory cost. We study problem (2.1) 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, and consequently extensions of the static problem should also prove efficiently solvable.

2.1 Prior work

In recent work, Bertsimas and Thiele [BT04] 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 $1 \leq t \leq T$,

$$d_t = \mu_t + \delta_t z_t, \tag{2.3}$$

$$z_t \in [-1, 1], \tag{2.4}$$

$$\sum_{j=1}^{t} |z_j| \leq \Gamma_t. \tag{2.5}$$

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 (2.5) constitute non-trivial requirements on the ensemble of all deviations. The method in [BT04] 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 [BT04] to the general case, the approach used in [BT04] formulates our basic inventory problem as the following linear program:

$$
\begin{aligned}
C^* \quad = \quad \min \quad & \sum_{t=1}^{T} (c_t u_t \,+\, y_t) && (2.6)\\
\text{s.t.} \quad & \\
& y_t \,\geq\, h_t \left(x_1 \,+\, \sum_{j=1}^{t} (u_j - \mu_j) \,+\, A_t \right) t = 1, \ldots T, && (2.7)\\
& y_t \,\geq\, b_t \left(-x_1 \,+\, \sum_{j=1}^{t} (\mu_j - u_j) \,+\, A_t \right) t = 1, \ldots T, && (2.8)\\
& u \geq 0,
\end{aligned}
$$

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

$$
\begin{aligned}
A_t \,=\, \max \quad & \sum_{j=1}^{t} \delta_j \epsilon_j^t && (2.9)\\
\text{s.t.} \quad & \sum_{j=1}^{t} \epsilon_j^t \,\leq\, \Gamma_t,\\
& 0 \,\leq\, \epsilon_j^t \,\leq\, 1, \quad 1 \leq j \leq t.
\end{aligned}
$$

Thus, LP (2.9) computes the *maximum* cumulative deviation away from the mean demands, by time t, that model (2.3)-(2.5) allows. If we denote by $\hat{\epsilon}_j^t$ $(1 \leq j \leq t)$ the optimal solution to LP (2.9), then constraint (2.7) 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 (2.8) 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 [BT04]). 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 (2.6) should be contrasted with the "true" min-max problem:

$$R^* \quad = \quad \min_{u \geq 0} \; R(u) \tag{2.10}$$

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

$$R(u) \quad = \quad \max_{d,z,x} \sum_{t=1}^{T} \left(\, c_t u_t \, + \, \max\{ \, h_t x_{t+1} \, , \, -b_t x_{t+1} \} \, \right) \tag{2.11}$$

s.t.

$$\begin{aligned} x_{t+1} &= x_t + u_t - d_t, \quad 1 \leq t \leq T, \\ d_t &= \mu_t \, + \, \delta_t z_t, \\ z_t &\in [-1, 1], \\ \sum_{j=1}^{t} |z_j| &\leq \Gamma_t, \quad 1 \leq t \leq T. \end{aligned}$$

We have that $R^* \leq C^*$ and the gap can be large. However, [BT04] empirically shows that in the case of stationary costs (2.6) provides an effective approximation to (2.10). This is significant because (2.11) is a non-convex optimization problem.

In addition, again in the case of stationary costs, it is shown in [BT04] that LP (2.6) 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 (2.6) can be solved "greedily" (every y_t simultaneously minimized) in the stationary case. Although the non-stationary case is not considered in [BT04], we can say that the results from the stationary case do not directly apply.

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 [GW74], and by ideas from Control Theory [GSc71], 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:

$$u_t \;=\; \hat{\alpha}_0^t + \sum_{i=1}^{t-1} \hat{\alpha}_i^t d_i, \tag{2.12}$$

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 (2.12) are the past demands. Using (2.12), the inventory holding/backlogging cost inequalities for time t become:

$$y_t \;\geq\; h_t \left(x_1 + \sum_{i=1}^{t-1} \left(\sum_{j=i+1}^{t} \hat{\alpha}_i^j - 1 \right) d_i - d_t + \sum_{j=1}^{t} \hat{\alpha}_0^j \right) t = 1, \ldots T, \tag{2.13}$$

$$y_t \;\geq\; b_t \left(-x_1 + \sum_{i=1}^{t-1} \left(1 - \sum_{j=i+1}^{t} \hat{\alpha}_i^j \right) d_i + d_t - \sum_{j=1}^{t} \hat{\alpha}_0^j \right) t = 1, \ldots T, \tag{2.14}$$

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 $\hat{\beta}^t_j$ $(0 \leq j \leq t-1)$ with $y_t = \sum_{j=1}^{t-1} \hat{\beta}^t_j d_j + \hat{\beta}^t_0$. Inserting this expression into (2.13), and rearranging, we obtain:

$$0 \ \geq \ h_t x_1 + \sum_{i=1}^{t-1} \left(h_t \left(\sum_{j=i+1}^{t} \hat{\alpha}^j_i - 1 \right) - \hat{\beta}^t_i \right) d_i \ - \ h_t d_t \ + \ h_t \sum_{j=1}^{t} \hat{\alpha}^j_0 \ - \ \hat{\beta}^t_0, \quad (2.15)$$

which can be abbreviated as

$$0 \ \geq \ \sum_{i=1}^{t} P^t_i(\hat{\alpha}, \hat{\beta})\, d_i \ + \ P^t_0(\hat{\alpha}, \hat{\beta}), \quad (2.16)$$

where each $P^t_i(\hat{\alpha}, \hat{\beta})$ is an affine function of $\hat{\alpha}$ and $\hat{\beta}$ (and similarly with (2.14)). The algorithm in [BGNV05] chooses the $\hat{\alpha}$ and $\hat{\beta}$ values so that (2.15) 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, (2.16) holds for each allowable demand if and only if there exists values $\hat{\nu}^t_i$, $1 \leq i \leq t$, such that

$$0 \ \geq \ \sum_{i=1}^{t} \left(P^t_i(\hat{\alpha}, \hat{\beta})\, \mu_i \ + \ \hat{\nu}^t_i \delta_i \right) \ + \ P^t_0(\hat{\alpha}, \hat{\beta}), \quad (2.17)$$

$$-\hat{\nu}^t_i \ \leq \ P^t_i(\hat{\alpha}, \hat{\beta}) \ \leq \ \hat{\nu}^t_i, \quad 1 \leq i \leq t. \quad (2.18)$$

Inequalities (2.17) and (2.18), 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 (2.14)). Notice, as was the case in [BT04], that this approach is conservative in that we may have $\nu^t_i \neq \nu^{t'}_i$ for some i and $t \neq t'$, i.e. the demands implied by some inequality (2.17) 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:

$$\text{cost} \quad = \quad \max \left\{ Q_0 + \sum_t Q_t d_t \; : \; d \in \mathcal{E} \right\}, \quad \text{where} \tag{2.19}$$

$$\mathcal{E} \quad = \quad \{ d \, : \, (d - \mu)' S (d - \mu) \, \leq \, \Omega \} \, . \tag{2.20}$$

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, (2.20) states that the demands cannot simultaneously take values "far" from their nominal values μ_t. As shown in [BGNV05], the system (2.19), (2.20) is equivalent to the problem:

$$\text{cost} \quad = \quad \min E, \tag{2.21}$$

$$\text{subject to:} \qquad Q_0 + \sum_t \mu_t Q_t + \left(\Omega \, Q' \, S^{-1} \, Q \right)^{1/2} \; - \; E \, \leq \, 0. \tag{2.22}$$

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

(2.21), and constraints (2.22), (2.17) and (2.18) (and nonnegativity of the orders, enforced through (2.12)). Such a problem can be efficiently solved using modern algorithms. [BGNV05] reports excellent results in examples with $T = 24$.

2.2 Demand uncertainty

Here we consider the following models for the demand uncertainty set:

1. The Bertsimas-Thiele model (2.3)-(2.5). 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.

The risk budgets and the bursty demand model will also be considered in the context of basestock policies and the dynamic problem. Later in the thesis we will describe other models.

2.3 The decision maker's problem

Step 1 of our generic algorithms requires us to solve the decision maker's problem for a subset of the demand uncertainty set. Let $\hat{\mathcal{D}} \subset \mathcal{D}$. Then it is easy to see that the decision maker's problem for the set $\hat{\mathcal{D}}$ can be formulated as the following linear program.

$$\text{Min} \quad \sum_{t=1}^{T} \left(c_t u_t + K_t^d \right) \tag{2.23}$$

$$\text{s.t} \quad K_t^d \geq h_t(x_1 + \sum_{i=1}^{t}(u_i - d_i)) \quad t = 1, 2, ..., T \quad \forall\ d \in \hat{\mathcal{D}} \tag{2.24}$$

$$K_t^d \geq -b_t(x_1 + \sum_{i=1}^{t}(u_i - d_i)) \quad t = 1, 2, ..., T \quad \forall\ d \in \hat{\mathcal{D}} \tag{2.25}$$

$$u \geq 0 \tag{2.26}$$

Here, K_t^d is a variable indexed by the period, t, and the demand vector, d.

Notice that the LP above has constraints (2.24) and (2.25) for every demand vector $d \in \hat{\mathcal{D}}$. Similarly, our main problem (2.1) can be formulated as a semi-infinite LP by having (2.24) and (2.25) for all $d \in \mathcal{D}$. Fortunately, (2.23)-(2.26) has a compact representation, i.e. there exists a small subset $\tilde{\mathcal{D}}$ of $\mathcal{D}$ such that the LP constructed by taking the constraints (2.24) and (2.25) corresponding to demand vectors in $\tilde{\mathcal{D}}$ will have that same optimal solution as our semi-infinite LP. Our algorithms construct this compact representation.

2.4 The adversarial problem under the risk budgets model

Here we consider the adversarial problem (step 2 of Algorithm 1.2.1) under the demand uncertainty model (2.3)-(2.5). For simplicity of presentation, in this section we will assume that the quantities Γ_t are integral – in Chapter 3, 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 result:

Lemma 2.4.1 *Let $\bar{d}$ be an extreme point of $\mathcal{D}$. Then for $1 \leq t \leq T$, either $\bar{d}_t = \mu_t$ or $|\bar{d}_t - \mu_t| = \delta_t$.*

Proof. We define $z_t = |\bar{d}_t - \mu_t|/\delta_t$ for $0 \leq t \leq T$. Suppose that Lemma is not true and there exist a time period t for which z_t is non-integral. Let t' be the largest such index. Suppose $\sum_{t=1}^{t'} z_t < \Gamma_{t'}$. Let $\epsilon = \min\{\Gamma_{t'} - \sum_{t=1}^{t'} z_t, z_{t'}, 1 - z_{t'}\}$. We form two demand patterns d^1 and d^2 by setting $d^1_{t'} = \bar{d}_{t'} - \epsilon\delta_{t'}$, $d^2_{t'} = \bar{d}_{t'} + \epsilon\delta_{t'}$ and $d^1_t = d^2_t = \bar{d}_t$. Note that $d_1, d_2 \in \mathcal{D}$ and $\bar{d} = \frac{d_1+d_2}{2}$ which contradicts with the fact that $\bar{d}$ is an extreme point of $\mathcal{D}$.

Suppose that $\sum_{t=1}^{t'} z_t = \Gamma_{t'}$. This means that there exists an index $1 \leq t < t'$ such that z_t is fractional. Let t'' be the largest such index. Note that $\sum_{t=1}^{t} z_t < \Gamma_t$ for every $t'' \leq t < t'$. We set

$$\epsilon = \min\{\Gamma_{t''} - \sum_{t=1}^{t''} z_t, ..., \Gamma_{t'-1} - \sum_{t=1}^{t'-1} z_t, z_{t''}, z_{t'}, 1 - z_{t''}, 1 - z_{t'}\}.$$

Similar to the previous case we form two vectors d^1 and d^2 by setting $d^1_{t'} = \bar{d}_{t'} - \epsilon\delta_{t'}$, $d^1_{t''} = \bar{d}_{t''} + \epsilon\delta_{t''}$, $d^2_{t'} = \bar{d}_{t'} + \epsilon\delta_{t'}$, $d^2_{t''} = \bar{d}_{t''} - \epsilon\delta_{t''}$ and $d^1_t = d^2_t = \bar{d}_t$. $d^1, d^2 \in \mathcal{D}$, and we can write $\bar{d}$ as a linear combination of d^1 and d^2. Therefore $\bar{d}$ is not an extreme point. ■

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,

$$\begin{aligned} \mathcal{A}_t(x,k) \;&=\; \max_d \sum_{j=t}^{T} \mathcal{W}_j\left(x + \sum_{i=1}^{j} \tilde{u}_i - \sum_{i=1}^{j} d_i\right) && (2.27)\\ \text{Subject to} \quad & \sum_{i=t}^{j} \left\{ \frac{|d_i - \mu_i|}{\delta_i} \;:\; \delta_i > 0 \right\} \le \Gamma_j \,-\, k, \quad t \le j \le T, \\ & \mu_j - \delta_j \;\le\; d_j \;\le\; \mu_j + \delta_j, \quad t \le j \le T. \end{aligned}$$

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 (2.27) 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 2.4.1, we have the following recursion:

$$\mathcal{A}_t(x,\Gamma_t) \;=\; \mathcal{W}_t(x + \tilde{u}_t - \mu_t) \;+\; \mathcal{A}_{t+1}(x + \tilde{u}_t - \mu_t, \Gamma_t), \qquad (2.28)$$

while for $k < \Gamma_t$,

$$\mathcal{A}_t(x,k) \;=\; \max\left\{ f^u_{t,k}(x)\,,\, f^d_{t,k}(x)\,,\, f^m_{t,k}(x) \right\}, \qquad (2.29)$$

where

$$\begin{aligned} f^u_{t,k}(x) \;&=\; \mathcal{W}_t(x + \tilde{u}_t - \mu_t - \delta_t) \quad + \; \mathcal{A}_{t+1}(x + \tilde{u}_t - \mu_t - \delta_t, k+1), && (2.30)\\ f^d_{t,k}(x) \;&=\; \mathcal{W}_t(x + \tilde{u}_t - \mu_t + \delta_t) \quad + \; \mathcal{A}_{t+1}(x + \tilde{u}_t - \mu_t + \delta_t, k+1), && (2.31)\\ f^m_{t,k}(x) \;&=\; \mathcal{W}_t(x + \tilde{u}_t - \mu_t) \quad + \; \mathcal{A}_{t+1}(x + \tilde{u}_t - \mu_t, k). && (2.32) \end{aligned}$$

Here we write $\mathcal{A}_{T+1}(x,k) = 0$ for all x, k As a result of equation (2.29) and the definition of the $f^u_{t,k}(x)$, $f^d_{t,k}(x)$, $f^m_{t,k}(x)$, we have:

Lemma 2.4.2 *For any t and k, $\mathcal{A}_t(x,k)$ is a convex, piecewise-linear function of x.*

■

Equations (2.28) and (2.29)-(2.32) 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 2.4.3 *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 $1 \leq a \leq m(1), 1 \leq b \leq m(2)$, 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 a 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 $s^1_a = s^2_b$ ($= s$, say) there is at most one piece of g with slope s. ■

In our implementation, we use the method implicit in Lemma 2.4.3 together with the dynamic programming recursion described above. We will present computational experience with this algorithm later. Here we present some comments on its complexity.

Note that in each equation (2.30)-(2.32) 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 (2.30) and (2.31), we see that $f^u_{t,k}(x) = f^d_{t,k}(x - 2\delta_t)$. Thus, as is easy to see (see Lemma 2.4.4 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 2.4.3 (a) is significant in that when we consider equations (2.30)-(2.32) 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.

2.4.1 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:

$$\mathcal{A}_t(x) \quad = \quad \max \left\{ f_t^u(x)\, ,\, f_t^d(x) \right\}, \tag{2.33}$$

where

$$f_t^u(x) \quad = \quad \mathcal{W}_t(x + \tilde{u}_t - \mu_t - \delta_t) \quad + \quad \mathcal{A}_{t+1}(x + \tilde{u}_t - \mu_t - \delta_t), \tag{2.34}$$

$$f_t^d(x) \quad = \quad \mathcal{W}_t(x + \tilde{u}_t - \mu_t + \delta_t) \quad + \quad \mathcal{A}_{t+1}(x + \tilde{u}_t - \mu_t + \delta_t). \tag{2.35}$$

and as before we set $\mathcal{A}_{T+1}(x) = 0$. We have, as a consequence of Lemma 2.4.3:

Lemma 2.4.4 *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.*

Proof. This follows directly from part (a) of Lemma 2.4.3. ■

Corollary 2.4.5 *For any t, the number of pieces in $\mathcal{A}_t(x)$ is at most $T - t + 2$.* ■

Corollary 2.4.6 *In the box model, the adversarial problem can be solved in time $O(T^2)$.* ■

Corollary 2.4.6 is significant for the following reason. In the box case, our min-max problem (2.1) can be written as:

$$\min_{u \geq 0} \quad \sum_{t=1}^{T} c_t\, u_t \quad + \quad z \tag{2.36}$$

$$\text{Subject to} \quad z \;\geq\; \sum_{j \in J} h_j \left(x_1 + \sum_{i=1}^{j} \tilde{u}_i - \sum_{i=1}^{j} d_i \right) \;-\; \sum_{j \in \bar{J}} b_j \left(x_1 + \sum_{i=1}^{j} \tilde{u}_i - \sum_{i=1}^{j} d_i \right),$$
$$\text{for all } d \in \mathcal{D}, \text{ and each partition } (J, \bar{J}) \text{ of } \{1, \ldots, T\}. \tag{2.37}$$

This linear program has $T+1$ variables but $2^T\,|\mathcal{D}|$ constraints. However, by Corollary 2.4.6, 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.

2.4.2 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 vector of orders. 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.

$$\max_{d,x,p,I,B,z} \quad \sum_{t=1}^{T} (I_t + B_t) \tag{2.38}$$

Subject to

for $1 \le t \le T$,

$$\begin{aligned}
x_{t+1} &= x_t + \tilde{u}_t - d_t, && (2.39)\\
h_t\, x_{t+1} &\le I_t \le h_t\, x_{t+1} + h_t\, M_t(1-p_t), && (2.40)\\
0 \le I_t &\le h_t\, M_t\, p_t, && (2.41)\\
-b_t\, x_{t+1} &\le B_t \le -b_t\, x_{t+1} + b_t\, M_t p_t, && (2.42)\\
0 \le B_t &\le b_t\, M_t\,(1-p_t), && (2.43)\\
d_t &= \mu_t + \delta_t\, z_t, && (2.44)\\
p_t &= 0 \text{ or } 1, && (2.45)\\
\sum_{j=1}^{t} |z_t| &\le \Gamma_t. && (2.46)
\end{aligned}$$

Equations (2.40)-(2.43) 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 $I_t = 0$. In order for the formulation to be valid we need to choose the constants M_t appropriately large – however, for efficiency of solvability, they should be chosen just large enough, and this can be done in a straightforward fashion.

Problem (2.38) bears a passing similarity to the traditional *economic lot-sizing* problem [WW58, BRW84]. 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.

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

Table 2.1: Solving the adversarial problem as a mixed-integer program

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.

Nevertheless, it is possible that a more efficient specialized algorithm for solving the mixed-integer program (2.38), or for a reformulation of it (there are many) could be developed. In fact, notice that by replacing equation (2.46) with the general condition $d \in \mathcal{D}$ we can in principle tackle the adversarial problem for general polyhedral set $\mathcal{D}$.

2.5 The adversarial problem in the bursty demand model

Here we consider the adversarial problem for the bursty demand model given in Section 2.2. 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 $1 \leq t \leq T$:

$$\Pi_t(x,k) = \max_{d \in \{\mu_t - \delta_t, \mu_t + \delta_t\}} \left\{ \mathcal{W}_t(x + \tilde{u}_t - d) + \Pi_{t+1}(x + \tilde{u}_t - d, k+1) \right\}, \quad \text{for } 1 \leq k < \min\{W-1, t\}, \tag{2.47}$$

$$\Pi_t(x, W-1) = \max_{d \in \{\mu_t - \delta_t, \mu_t + \delta_t\}} \left\{ \mathcal{W}_t(x + \tilde{u}_t - d) + \Pi_{t+1}(x + \tilde{u}_t - d, 0) \right\},$$

$$\text{for } W-1<t, \tag{2.48}$$

$$\Pi_t(x,0) = \max\left\{ \Pi_t^1(x)\, ,\, \Pi_t^0(x)\right\}, \quad \text{where} \tag{2.49}$$

$$\Pi_t^1(x) = \max_{d\in\{\mu_t-\delta_t,\mu_t+\delta_t\}} \left\{\mathcal{W}_t(x+\tilde{u}_t-d) \,+\, \Pi_{t+1}(x+\tilde{u}_t-d,0)\right\}, \tag{2.50}$$

$$\Pi_t^0(x) = \mathcal{W}_t(x+\tilde{u}_t-P_t) \,+\, \Pi_{t+1}(x+\tilde{u}_t-P_t,1). \tag{2.51}$$

We solve this recursion using the same approach as for (2.28)-(2.32), i.e. by storing the representation of each function $\Pi_t(x,k)$ (which clearly are convex piecewise-linear).

2.6 Computational results for the static problem

To investigate the behavior of our algorithms for the static case, we ran several sets of tests, with results reported in Table 2.3. 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 $T = 50, 200, \text{and}\, 500$. 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 that 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 extend this to other periods by discounting this data using a yearly rate of 0.95. We generated the demand intervals

	$[l_1, l_2]$	$[h_1, h_2]$	**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 2.2: Parameters for data generation

in that case randomly (see below). For the pure random case, data in each period is generated independently 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 value with a random number which is uniformly distributed between 0 and 1. Table 2.2 demonstrates the parameters we used.

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 t by setting $\Gamma_t = \Gamma_{t-1} + \zeta$ where $\zeta = 1$ with probability q and $\zeta = 0$ otherwise. The parameter q is likewise randomly chosen with uniform distribution.

We also tested our algorithm on stationary instances in which the budgets are generated by the procedure given in [BT04]. 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 [BT04] 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 necessaryly 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 2.4.

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 2.3 may overstate the amount of work needed to converge. This is because in addition to requiring few iterations, frequently the algorithm quickly converged to a near-optimal solution and the additional iterations were needed in order to close a very small gap. Figure 2.1 shows a typical example of this behavior. In the figure the gap between the lower and the upper bounds for the cost decreases rapidly and after fifth iteration we obtain a solution that is very close to the optimal in terms of cost.

		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 2.3: Running time and number of iterations

	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 2.4: Running time and number of iterations for the budgets model

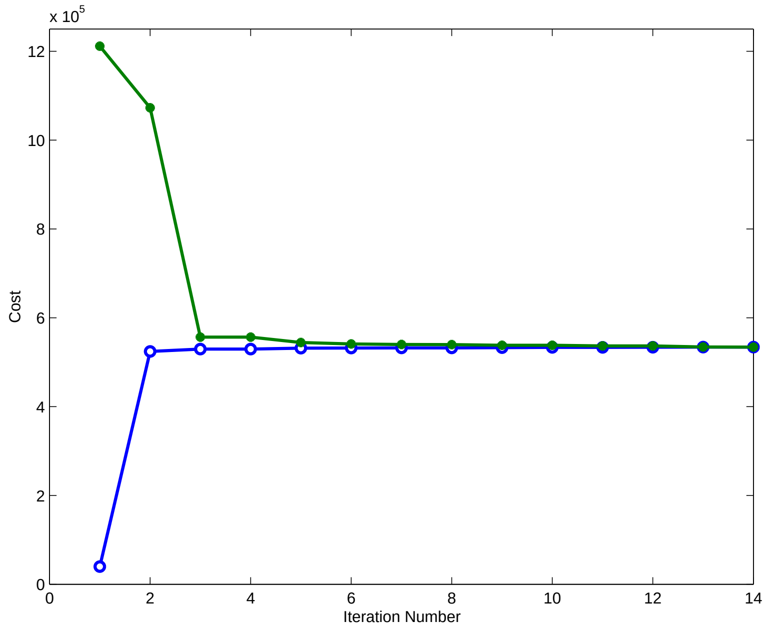

Figure 2.1: Example with many steps

Chapter 3

The Basestock Problem

The main focus of this chapter concerns how to pick optimal basestock policies in a robust setting, under various demand uncertainty sets $\mathcal{D}$. Our focus is motivated, primarily, by the fact that basestock policies have acquired very wide use. Basestock policies can be shown to be optimal under many inventory models [Z00]. Further, even though such policies may not always be optimal, they are 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 the problem faced by 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:

$$\min_{\sigma \geq 0} \; V(\sigma) \tag{3.1}$$

where for $\sigma \geq 0$,

$$V(\sigma) \;=\; \max_{d,x,u} \sum_{t=1}^{T} \left(c_t u_t \;+\; \max\{\, h_t x_{t+1} \,,\; -b_t x_{t+1} \} \right) \tag{3.2}$$

s.t.

$$u_t \;=\; \max\{\sigma - x_t \,,\, 0\}, \quad 1 \leq t \leq T, \tag{3.3}$$

$$x_{t+1} \;=\; x_t + u_t - d_t, \quad 1 \leq t \leq T, \tag{3.4}$$

$$(d_1, d_2, \ldots, d_T) \;\in\; \mathcal{D}. \tag{3.5}$$

Here, (3.2)-(3.5) is the adversarial problem – once the demand variables $(d_1, d_2, \ldots, d_T)$ $\in \mathcal{D}$ have been chosen, constraints (3.3)-(3.4) uniquely determine all other variables. Note that the quantity x_1 (the initial inventory level) is an input. Also, because of the "max" in (3.2) and (3.3), the adversarial problem is non-convex.

Note that we assume $\sigma \geq 0$ in (3.1) – 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 (3.1) posits a constant basestock over the entire planning horizon. However, we would expect that in practice the policy would be periodically reviewed ((3.1) would be 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 (3.1) can be viewed as an operational feature aimed at achieving stability (and "implementability") of the policy used to operate the supply chain in the face of large, and difficult to quantify, demand uncertainty. 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.

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 will give a result regarding the adversarial problem in this general setting in Section 3.6. 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 in principle 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 present some results concerning this model in Section 3.6.2.

The following three sections consider how to solve problem (3.1) using our generic algorithm (1.2.1), under the risk budgets and bursty demand uncertainty models. Section 3.2 considers the decision maker's problem. The adversarial problem is studied in Section 3.3 (for the risk budgets model) and Section 3.4 (bursty demands model).

3.1 Preliminaries

In the context of algorithm (1.2.1), 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 (1.2)-(3.3). Prior to describing our algorithms, we note a simple observation.

Definition 3.1.1 *Let d be a demand vector. For $1 \leq t \leq T$, write $\mathcal{R}_{t,d} = x_1 - \sum_{j=1}^{t-1} d_j$. Write $\mathcal{R}_{0,d} = +\infty$.*

Definition 3.1.2 *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 3.1.3 *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,*

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

Remark 3.1.4 *For $1 \le t \le T$, 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 $t \ge t^*$, $x_t \le \sigma$, and for every $t \le t^*$, $x_t = \mathcal{R}_{t,d}$.

(b) For $t < t^*$, $u_t = 0$. For $t < t^* - 1$ we have, by definition of t^*, that $0 \le \sigma \le \mathcal{R}_{t+1,d} = x_{t+1}$, 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 $\mathcal{W}_t(\sigma - d_t)$.

3.2 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 $icost_t(\sigma, d)$ and $ocost_t(\sigma, d)$ denote the inventory and ordering costs incurred at time t, under demands d, if we use basestock σ, respectively. Further, we denote $cost(\sigma, d) = \sum_t (ocost_t(\sigma, d) + icost_t(\sigma, d))$.

Lemma 3.2.1 *For fixed d, $\sum_t ocost_t(\sigma, d)$ is a piecewise linear function of σ with $T+1$ pieces.*

Proof. Notice that

$$\sum_t ocost_t(\sigma, d) = \begin{cases} 0 & if\ \sigma \leq \mathcal{R}_{T,d} \\ c_t(\sigma - \mathcal{R}_{T,d}) & if\ \mathcal{R}_{T-1,d} < \sigma \leq \mathcal{R}_{T,d} \\ c_t(\sigma - \mathcal{R}_{t,d}) + \sum_{i=t}^{T-1} c_{i+1} d_i & if\ \mathcal{R}_{t,d} < \sigma \leq \mathcal{R}_{t-1,d}\ for\ 1 \leq t \leq T-1 \end{cases}$$

Here we assume that $\mathcal{R}_{0,d} = \infty\ \forall d \in \mathcal{D}$. ■

Lemma 3.2.2 *$\sum_t icost_t(\sigma, d)$ is a convex function of σ if either $\mathcal{R}_{t,d} \geq 0$ for every $1 \leq t \leq T+1$ or $\mathcal{R}_{t,d} \leq 0$ for every $1 \leq t \leq T+1$.*

Proof. Suppose that $\mathcal{R}_{i,d} \geq 0$ for all $1 \leq i \leq T$. Then the cost corresponding to period t can be written as

$$
\begin{aligned}
icost_t(\sigma, d) &= \begin{cases} h_t(\mathcal{R}_{t,d} - d_t) & if \quad \sigma \leq \mathcal{R}_{t,d} \\ \\ h_t(\sigma - d_t) & if \quad \sigma > \mathcal{R}_{t,d} \end{cases} \\
&= h_t \max\{\sigma - d_t, \mathcal{R}_{t,d} - d_t\}
\end{aligned}
$$

which is a convex function of σ. Similarly if $\mathcal{R}_{i,d} \leq 0$ for all $1 \leq i \leq T$, then for $1 \leq i \leq T$

$$icost_t(\sigma, d) \;=\; \max\{h_t(\sigma - d_t), -b_t(\sigma - d_t)\}$$

which is convex in σ.

Consequently, $icost_t(\sigma, d)$ is a convex for all $1 \leq t \leq T$. ■

Now suppose that there exist a time period $2 \leq t' \leq T$ such that $\mathcal{R}_{t',d} < 0$ and $\mathcal{R}_{t'-1,d} \geq 0$.

Lemma 3.2.3 *$cost(\sigma, d)$ is a convex function of σ for the sets $L = \{\sigma : \sigma \leq \mathcal{R}_{t'-1,d}\}$ and $R = \{\sigma : \sigma \geq \mathcal{R}_{t'-1,d}\}$.*

Proof. First we consider L. For $\sigma \in L$ the one period cost functions can be written as follows. For $1 \leq t \leq t' - 1$ we have

$$icost_t(\sigma, d) = h_t(\mathcal{R}_{t,d} - d_t)$$

which is a linear function.

For $t' \leq t \leq T$

$$icost_t(\sigma, d) = \max\{h_t(\sigma - d_t), -b_t(\sigma - d_t)\}.$$

Therefore, $cost(\sigma, d)$ is convex in L.

Now we consider $\sigma \in R$. Similar to L we will show that one period cost is a convex function of σ for each time period. For $1 \leq t \leq t' - 1$

$$\begin{aligned} icost_t(\sigma, d) &= \begin{cases} h_t(\mathcal{R}_{t,d} - d_t) & if \quad \sigma \leq \mathcal{R}_{t,d} \\ h_t(\sigma - d_t) & if \sigma > \mathcal{R}_{t,d} \end{cases} \\ &= \max\{h_t(\sigma - d_t), h_t(\mathcal{R}_{t,d} - d_t)\}. \end{aligned}$$

For $t' \leq t \leq T$

$$icost_t(\sigma, d) = \max\{h_t(\sigma - d_t), -b_t(\sigma - d_t)\}.$$

$icost_t(\sigma, d)$ is clearly convex for all $1 \leq t \leq T$ in both cases. ■

Corollary 3.2.4 $\max_{\tilde{d} \in \tilde{D}} cost(\sigma, \tilde{d})$ *is piecewise convex with at most* $(T+2)|\tilde{D}|$ *pieces, 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 Corollary 3.2.4 :

(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 $n \leq (T+2)|\tilde{D}|$.

(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 exist other algorithms that in theory, are more efficient [NW99], but empirically our implementation proves 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 $cost(\hat{\sigma}, \tilde{d})$ function.

Further, in the context of our generic algorithm 1.2.1, 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.

3.3 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 (2.3)-(2.5), 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 first 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 3.3.1 *Suppose $t^* \geq T$. Then d^* is obtained by solving the following linear programs, and choosing the solution with higher value:*

$$\begin{aligned} Max \quad & \sum_{t=1}^{T} h_t \left(x_1 - \sum_{j=1}^{t} d_j \right) \\ s.t. \quad & d \in \mathcal{D} \\ & x_1 - \sum_{t=1}^{T-1} d_t \,\geq\, \sigma. \end{aligned} \tag{3.6}$$

$$\begin{aligned} Max \quad & \sum_{t=1}^{T-1} h_t \left(x_1 - \sum_{j=1}^{t} d_t \right) + b_T \left(\sum_{t=1}^{T} d_t - x_1 \right) \\ s.t. \quad & d \in \mathcal{D} \\ & x_1 - \sum_{t=1}^{T-1} d_t \,\geq\, \sigma. \end{aligned} \tag{3.7}$$

■

Lemma 3.3.1 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 (1.2.1), 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 3.5 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 Section 3.3.6 for details.

(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 3.6.4). 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 3.3.2 and 3.3.3 and Remark 3.3.4, all given below, provide some structural properties of an optimal solution to the adversarial problem. Sections 3.3.1, 3.3.2 and 3.3.3 describe the technical details of our approach. The overall algorithm is put together in Section 3.3.4. The approximate algorithm is described in Section 3.3.5, the case with the integral budgets is considered in Section 3.3.6 and the bounding procedure based on integral budgets is given in Section 3.3.7.

Lemma 3.3.2 *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 3.1.4, (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 3.3.2 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:

$$\begin{aligned} Max \quad & \sum_{t=1}^{t^*-1} h_t \left(x_1 - \sum_{j=1}^{t} d_t \right) + c_{t^*} \left(\sigma - (x_1 - \sum_{t=1}^{t^*-1} d_t) \right) \\ s.t. \quad & d_t = \mu_t + \delta_t z_t, \quad 1 \le t \le t^* - 1, \\ & z_t \in [-1, 1], \quad 1 \le t \le t^* - 1, \\ & \sum_{j=1}^{t} |z_j| \le \Gamma_t, \quad 1 \le t \le t^* - 2, \\ & \sum_{j=1}^{t^*-1} |z_{t^*-1}| \le \Gamma_T - (T - t^* + 1), \\ & x_1 - \sum_{t=1}^{t^*-2} d_t \ge \sigma, \\ & x_1 - \sum_{t=1}^{t^*-1} d_t \le \sigma. \end{aligned} \tag{3.8}$$

■

In total, case (b) amounts to T linear programs of type (3.8). In what follows, we assume that case (a) holds, and that furthermore the period t^e is chosen as small as possible.

Lemma 3.3.3 *Without loss of generality, there is at most one period t^f with $t^* \le t^f \le t^e$, 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:

$$B = \left\{1, 2, \dots, t^* - 1, t^f\right\}, \tag{3.9}$$

$$M = \left\{t^* + 1, t^* + 2, \dots, t^f - 1, t^f + 1, \dots t^e\right\}, \tag{3.10}$$

$$F = \{t^e + 1, t^e + 2, \dots, T\}. \tag{3.11}$$

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

$$l(\gamma, t) = \begin{cases} \Gamma_t - \lfloor \Gamma_t \rfloor - (\gamma - \lfloor \gamma \rfloor), & \text{if } \Gamma_t - \lfloor \Gamma_t \rfloor \geq \gamma - \lfloor \gamma \rfloor \\ \\ 1 + \Gamma_t - \lfloor \Gamma_t \rfloor - (\gamma - \lfloor \gamma \rfloor), & \text{otherwise.} \end{cases} \tag{3.12}$$

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 3.3.4 $|z^*_{t^f}| = l(\gamma^*, t^e)$.

In the following sections 3.3.1, 3.3.2, 3.3.3 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 3.3.3.

3.3.1 Handling M.

We consider first

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

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

$$s.t.$$

$$\begin{aligned}
d_i &= \mu_i + \delta_i z_i & \forall i \in M \quad & (3.13)\\
\sum_{j=t^*}^{i} |z_i| &\le \lfloor \Gamma_i - \gamma \rfloor & t^* \le i \le t^f - 1 \quad & (3.14)\\
\sum_{j=t^*}^{t^f-1} |z_i| &\le \lfloor \Gamma_{t^f} - (\gamma + l(\gamma, t^e)) \rfloor & & (3.15)\\
\sum_{j=t^*}^{t^f-1} |z_i| + \sum_{j=t^f+1}^{i} |z_i| &\le \lfloor \Gamma_i - (\gamma + l(\gamma, t^e)) \rfloor, & t^f + 1 \le i \le t^e, \quad & (3.16)\\
-1 \le z_i \le 1 & & \forall i \in M, &
\end{aligned}$$

Lemma 3.3.5 *$(d^*(M), z^*(M))$ 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 3.3.4 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 (2.38). 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 (3.15) and (3.16) 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, $0 \leq i \leq t^f - t^*$, then any two values γ with $f_{(i)} < \gamma - \lfloor \gamma \rfloor \leq f_{(i+1)}$ will produce the same right-hand sides for constraints (3.14)-(3.16). Thus, for fixed $\lfloor \gamma \rfloor$, t^*, t^f, and t^e there are only $O(t^f - t^*)$ distinct problems $P_M(\gamma, t^*, t^f, t^e)$.

3.3.2 Handling B.

Next we turn to set B (c.f. (3.9)). Consider the optimization problem, for $0 \leq k \leq \Gamma_{t^*-1}$ and $0 \leq j \leq t^f - t^* + 1$:

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

$$\begin{aligned}
\max_{d,z,y,\gamma} \quad & \sum_{i=1}^{t^*-2} h_i \left(x_0 - \sum_{h=1}^{i} d_h \right) + \mathcal{W}_{t^*-1} \left(x_1 - \sum_{h=1}^{t^*-1} d_h \right) + \\
& + c_{t^*} \left(\sigma - \left(x_0 - \sum_{h=1}^{t^*-1} d_h \right) \right) + \mathcal{W}_{t^f} \left(\sigma - d_{t^f} \right) + c_{t^f+1} d_{t^f} \\
s.t. \quad & x_0 - \sum_{h=1}^{i} d_h \geq \sigma \qquad \forall i \in \{1, 2, ..., t^* - 1\} \\
& x_0 - \sum_{h=1}^{t^*-1} d_h \leq \sigma \\
& d_i = \mu_i + \delta_i z_i \qquad \forall i \in \{1, 2, ..., t^* - 1, t^f\} \\
& |z_i| \leq y_i \leq 1 \qquad \forall i \in \{1, 2, ..., t^* - 1, t^f\} \\
& \sum_{h=1}^{i} y_h \leq \Gamma_i \qquad \forall i \in \{1, 2, ..., t^* - 1\} \\
& \sum_{h=1}^{t^*-1} y_h - \gamma = 0 \\
& k + f_{(j)} \leq \gamma \leq k + f_{(j+1)} \qquad (3.17) \\
& y_{t^f} = l(\gamma, t^e) \qquad (3.18)
\end{aligned}$$

This problem models the behavior of the adversary during those periods in B. Here, γ is the total uncertainty consumed in periods $1 \leq t \leq t^* - 1$. The first term in the objective is the inventory holding cost incurred in periods $1 \leq i \leq t^* - 2$, 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 (3.18) 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 3.3.6 *Suppose that* $k + f_{(j)} \leq \gamma^* \leq k + f_{(j+1)}$ *for integers* $0 \leq k \leq \lfloor \Gamma_{t^*-1} \rfloor$ *and* $1 \leq j \leq t^f - t^* + 2$*. 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 (3.18). 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. (3.12)) 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 (3.18) with the constraint

$$y_{t^f} = 1 + f_{t^e} - (\gamma - k). \tag{3.19}$$

The right-hand side of (3.19) 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 (3.19) 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 3.3.6.

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 (3.18) with (3.19). 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 (3.19) 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 3.3.7 *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.* ■

3.3.3 Handling F

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

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

$$\begin{aligned} \max \quad & \sum_{i=t}^{T} \mathcal{W}_i(\sigma - d_i) + \sum_{i=t+1}^{T+1} c_i d_{i-1} && && (3.20) \\ s.t. \quad & d_i = \mu_i + \delta_i z_i && t \le i \le T \\ & \sum_{j=t+1}^{i} |z_i| \le \Gamma_i - \Gamma_{t-1} && t \le i \le T \\ & -1 \le z_i \le 1 && t \le i \le T \end{aligned}$$

The following is clear:

Lemma 3.3.8 *If $t^e < T$, $(d^*(F), z^*(F))$ solves $P_F(t^e + 1)$.* ■

3.3.4 The algorithm

Assuming first that there is a time period t^f as in Lemma 3.3.3, 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 3.3.7, 3.3.6 and 3.3.8 (also see the remarks preceding Lemma 3.3.7), 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 3.3.3 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 (3.15) and suitably modifying constraint (3.16).

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 (see Appendix A). 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 $t \leq i \leq T$, 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 (3.17); 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 implementations tricks, is important.

3.3.5 The approximate adversarial algorithm

In the discussion above we focused on solving the adversarial problem in Algorithm 1.2.1 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 reap the benefit of updating the set $\tilde{D}$ in Algorithm 1.2.1 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 1.2.1 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 1.2.1 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 3.3.4) 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

$$d_{\bar{t}^f} \;=\; \mu_{\bar{t}^f} \pm l(\gamma, \bar{t}^e)\, \delta_{\bar{t}^f} \tag{3.21}$$

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

$$d_{\bar{t}^f} \;=\; \mu_{\bar{t}^f} \pm \delta_{\bar{t}^f}. \tag{3.22}$$

Our method picks that period $\bar{t}^f$ for which the decrease from (3.22) to (3.21) 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.

3.3.6 Integral budgets case

When we have integral budgets the problem becomes easier. We can eliminate some of the components in the 5-tuple $(\bar{t}^*, \bar{t}^f, \bar{t}^e, k, j)$ that we examine in our original algorithm. First of all, we do not need last component, since it is only used to keep track of the fractional parts of the budgets. Moreover, the following Lemma shows that we can also eliminate t^e.

Lemma 3.3.9 *z_t^* is integral for all $t^e \leq t \leq T$.*

Proof. Suppose that the Lemma is not true and there exist a time period after t^e that has fractional risk consumption. Let $\bar{t}_1$ be the smallest of such periods. Then, there exists another time period $\bar{t}_2 > \bar{t}_1$ whose risk consumption is fractional: otherwise, we can reduce the risk consumption at $\bar{t}_1$ to zero, without decreasing cost; or we can increase it (without exceeding budgets) and increase cost. However due to convexity of the cost we can either increase the risk in $\bar{t}_1$ and decrease the risk in $\bar{t}_2$ by the same amount or do the reverse and get a new demand pattern that has a cost which is a least as large as the former demand pattern. ■

Using Lemma 3.3.9, we can extend the dynamic program for computing the cost for period $t^* \leq t \leq t^e$ to compute the cost for the periods $t^e < t \leq T$ and we can eliminate

t^e. Therefore, instead of solving P_B, P_M and P_F for each possible value of the 5-tuple $(\bar{t}^*, \bar{t}^f, \bar{t}^e, k, j)$, we solve P_B for each possible value of triple $(\bar{t}^*, \bar{t}^f, k)$ (we change r.h.s. of 3.18 with $1-(\gamma-\lfloor\gamma\rfloor)$) and we use the extended dynamic program to compute the cost for periods $t^*, t^*+1, \ldots, T$ for each value of $(\bar{t}^*, \bar{t}^f, k)$. Consequently, the number of steps is reduced to $O(T^2\Gamma_T)$ for the integral budgets case.

When $\Gamma_t = t$ for each t, we can further decrease the complexity of the algorithm. In this case, since the risk budgets have no impact, we can eliminate the fourth component of our 5-tuples (the integral part of the risk budget that is consumed up to time period t^*). Therefore, the step count further reduces to $O(T^2)$.

3.3.7 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 3.3.5 (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.

3.4 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:

Theorem 3.4.1 *The adversarial problem in the bursty demand model is NP-hard.*

Proof. See Section B.1. ■

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 3.1: t^* is the earliest period such that the starting inventory is at most σ. 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:

$IP(t^*, k)$:

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

$$s.t. \quad x_1 - \sum_{j=1}^{t} d_j \geq \sigma \qquad 1 \leq t \leq t^*-2 \tag{3.23}$$

$$x_1 - \sum_{j=1}^{t^*-1} d_j \leq \sigma \tag{3.24}$$

$$d_t = s_t + I_t P_t, \quad I_t \in \{0, 1\} \qquad 1 \leq t \leq t^*-1 \tag{3.25}$$

$$(1 - I_t)(\mu_t - \delta_t) \leq s_t \leq (1 - I_t)(\mu_t + \delta_t) \qquad 1 \leq t \leq t^*-1 \tag{3.26}$$

$$\sum_{i=t}^{t+W-1} I_t \leq 1 \qquad 1 \leq t \leq t^*-W \tag{3.27}$$

$$I_{t^*-k} = 1 \tag{3.28}$$

In this formulation, the $0-1$ variable I_t is used to indicate exceptional periods. If we set $k = 0$, and replace (3.28) 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 last W periods.

We will return to problem $IP(t^*, k)$ below, but first we consider the periods after t^*. This part can be handled 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

$$V_t(0) \;=\; \max_{d \in \{\mu_t - \delta_t, \mu_t + \delta_t, P_t\}} \{\ \mathcal{W}_t(\sigma - d) \ +\ c_{t+1} d \ +\ V_{t+1}(I)\ \}, \tag{3.29}$$

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

$$V_t(k) \;=\; \max_{d \in \{\mu_t - \delta_t, \mu_t + \delta_t\}} \{\mathcal{W}_t(\sigma - d) \ +\ c_{t+1} d \ +\ V_{t+1}(k + 1 \ (\mathrm{mod}\, W))\}. \tag{3.30}$$

For $t = T$, we set

$$V_T(0) \;=\; \max_{d \in \{\mu_T - \delta_T, \mu_T + \delta_T, P_t\}} \mathcal{W}_T(\sigma - d), \tag{3.31}$$

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

$$V_T(k) \;=\; \max_{d \in \{\mu_T - \delta_T, \mu_T + \delta_T\}} \mathcal{W}_T(\sigma - d). \tag{3.32}$$

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 (3.25), (3.26) and (3.27) (we do not include (3.28) 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 (3.27) 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, instances with T even in the hundreds, are solved in *hundredths* of a second.

3.5 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.

3.5.1 The risk budgets model

In Table 3.1 we study the behavior of the exact algorithm and that of the bounding procedure described in Section 3.3.7. 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.
Table 3.1 may overstate the difference between the early termination solution and the optimal solution: in Table 3.2 we compare the early termination basestock with the optimal basestock level (same data as Table 3.1). We see that in most cases the early termination basestock indeed provides an excellent approximation to the optimum.

Table 3.3 presents the running time of the algorithm for instances with integral

	Exact Algorithm			Early Termination					
	Running Time (sec.)			Running Time (sec.)			Cost Gap (%)		
	Avg.	Max	Min	Avg.	Max	Min	Avg.	Max	Min
Random	187.8	1362.6	1.35	13.91	58.89	2.02	1.52	12.22	0.09
Periodic	186.98	1659.1	2.83	11.41	56.25	1.56	1.42	8.71	0.00
Disc.	61.22	272.3	1.39	8.18	34.60	1.97	1.75	4.97	0.03

Table 3.1: Performance of algorithm for risk budgets ($T = 100$).

	% 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 3.2: Error in the basestock produced by using early termination.

		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 3.3: Performance statistics – integral budgets

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.

In Table 3.4 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**
	75	99.9737
Random	150	99.9934
	75	99.9711
Periodic	150	99.9977
	75	99.9765
Discounted	150	99.9999

Table 3.4: Ratio of adversarial time to total running time for the budgets model

In Table 3.5 we compare an optimal static policy, computed as in Chapter 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.

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

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 3.5: Static vs Basestock Policies

a rolling horizon fashion, to the optimal static policy (Chapter 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 $|\bar{d}_t - \mu_t|/\delta_t$.

	Dynamic policy				Static policy			
	avg.	**stddev**	**min**	**max**	**avg.**	**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
Disc.	-17.34	30.89	-72.31	82.09	606.18	274.49	87.35	1,220.91

Table 3.6: % increase in average cost of dynamic and static policies over the rolling horizon basestock policy

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 2.6. 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 3.6 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.

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

Another issue of interest is to quantify the impact of an incorrect basestock choice.

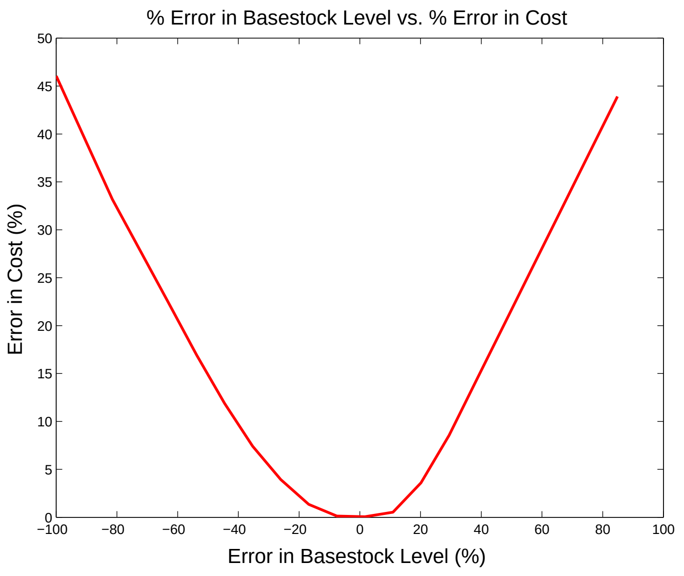

Figure 3.1: % error in basestock vs. % error in cost

Figure 3.1 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 the basestock level, the cost curve is flat indicating that we can use near optimal basestock levels without sacrificing too much 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 3.1 shows that for large enough basestock error, the cost error grows linearly.

3.5.2 The bursty demand model

For each category shown in Table 3.7, 200 tests were performed. Data was generated using the same procedure as in Section 2.6. In addition, in most of these instances

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

the window size was 15.

Table 3.8 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.

In Table 3.9, we investigate the impact of changing the initial inventory amount. Here we use the formula $x_1 = \mathit{step} \times \Phi/15$, where $\mathit{step} = 0, 1, 2, \ldots$ 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

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 3.8: Impact of window size on a 300-period model

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 3.9 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 the 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,

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 3.9: Impact of initial inventory

and then drops to zero – when the initial 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 3.10 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 3.10 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.

We performed some additional tests to measure the sensitivity of the optimal

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

Table 3.10: Variance vs Optimal Basestock

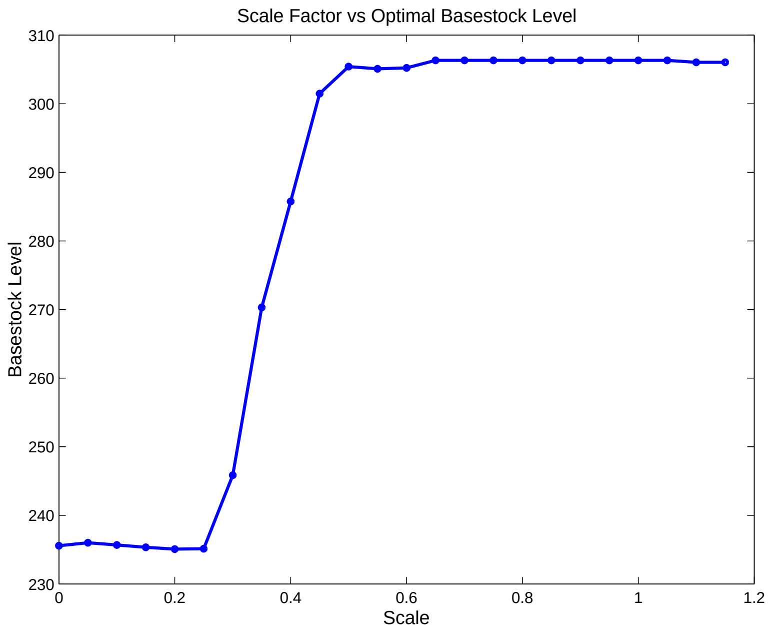

Figure 3.2: Effect of scaling peaks on optimum basestock

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

3.6 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.

3.6.1 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 2.4.2 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 3.6.1 *Let G be a graph with n vertices and clique number $\kappa(G)$. Then there is a robust basestock problem with $n+1$ periods and polyhedral uncertainty set $\mathcal{D}$, with value equal to $\kappa(G)$.*

Proof. See Section B.2. ■

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.

3.6.2 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:

$$\min_{\lambda^l \le \lambda \le \lambda^u} \quad F(\lambda) \tag{3.33}$$

where, given λ,

$$F(\lambda) \quad = \quad \max_{d,x,u} \sum_{t=1}^{T} (\, c_t u_t \, + \, \mathcal{W}_t(x_{t+1})) \tag{3.34}$$

s.t.

$$u_t \;=\; \max\{\hat{\mu}_t + \lambda\hat{\delta}_t - x_t\,,\, 0\}, \quad 1 \le t \le T, \tag{3.35}$$

$$x_{t+1} \;=\; x_t + u_t - d_t, \quad 1 \le t \le T, \tag{3.36}$$

$$(d_1, d_2, \ldots, d_T) \;\in\; \mathcal{D}. \tag{3.37}$$

To this end we propose the use of Algorithm 1.2.1. 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 3.6.2 *Let d be a fixed demand vector. Then cost(λ, 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 $\hat{\mu}_{T+1} + \lambda\hat{\delta}_{T+1} \geq x_1$ and $\hat{\mu}_{T+1} + \lambda\hat{\delta}_{T+1} \geq \hat{\mu}_t + \lambda\hat{\delta}_t$ for every $t \leq T$ 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 $x_{T+1} \leq \hat{\mu}_{t+1} + \lambda\hat{\delta}_{T+1}$].

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

$$\hat{\mu}_{t_1} + \lambda\hat{\delta}_{t_1} - \sum_{j=t_1}^{i} d_j \geq \hat{\mu}_{i+1} + \lambda\hat{\delta}_{i+1}, \quad t_1 \leq i \leq t_2 - 2, \tag{3.38}$$

$$\hat{\mu}_{t_1} + \lambda\hat{\delta}_{t_1} - \sum_{j=t_1}^{t_2-1} d_j \leq \hat{\mu}_{t_2} + \lambda\hat{\delta}_{t_2}. \tag{3.39}$$

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 (3.38) and (3.39), 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 $\alpha^1_L(t_1, t_2)$ and $\alpha^1_U(t_1, t_2)$ such that λ satisfies (3.38) and (3.39) if and only if $\lambda \in [\alpha^1_L(t_1, t_2),\ \alpha^1_U(t_1, t_2)]$ (possibly, the α are infinite).

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

$$\hat{\mu}_t + \lambda\hat{\delta}_t - d_t \leq \hat{\mu}_{t+1} + \lambda\hat{\delta}_{t+1}, \tag{3.40}$$

with a similar interpretation, and defining an interval $[\alpha^2_L(t), \alpha^2_U(t)]$. Finally, for each

$2 \leq t \leq T+1$, consider the system

$$\begin{aligned} x_1 &\geq \hat{\mu}_1 + \lambda \hat{\delta}_1 && (3.41) \\ x_1 - \sum_{j=1}^{i} d_j &\geq \hat{\mu}_i + \lambda \hat{\delta}_i \quad 1 \leq i \leq t-2, && (3.42) \\ x_1 - \sum_{j=1}^{t-1} d_j &\leq \hat{\mu}_t + \lambda \hat{\delta}_t, && (3.43) \end{aligned}$$

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 3.6.3 *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

$$d_t \;=\; \mu_t + k_t \, \alpha_i, \tag{3.44}$$

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. (We assume that Γ_T is polynomial in T).

Comments. We may view each category as describing a variance “class” or “type”. The weakness of (3.44) 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 3.6.4 *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.*

Proof. See Section C.1. ■

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 3.4 would prove far more practical (though nominally of exponential complexity since it solves mixed-integer programs). If anything, the fact that Theorem 3.6.4 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 $\sum_{j=1}^{t} |k_j| \le \Gamma_t$ (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 $0 < k_t \le$

$U(t)$, or $0 < -k_t \le L(t)$, 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 $\sum_{j=1}^{t} k_j \le \Gamma_t$.

In all the above models we were using budget constraints motivated by the original risk budgets model in [BT04]. 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 $\alpha_i \ge 0$, for $1 \le i \le C$ (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 $\sum_{t \in I} |k_t| \le \Gamma_I$, for each $I \in \mathcal{I}$.

We will say that an interval model with family $\mathcal{I}$ is of *width* $\le \omega$, 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 $1 \le k \le m$ and $i_k \le h \le j_k$, and

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

We call the intervals $[i_1, j_1]$, $\ldots$, $[i_m, j_m]$ basis for set $\mathcal{I}$. 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 3.6.5 *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*

Proof. See Section C.2 ■.

3.6.3 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. 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 1.2.1.

Here we focus on basestock policies.

Lemma 3.6.6 *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.*

Proof. To prove part (a) notice that for a fixed demand pattern d, cost of using basestock level σ is equal to

$$c_1(\sigma - x_1) + \sum_{i=1}^{T}(c_{i+1}d_i + \max\{h_i(\sigma - d_i), b_i(d_i - \sigma)\})$$

which is a convex function of σ. Therefore, $\mathbf{E}\,cost(\sigma|P)$ will be a convex function of σ.

For part (b), consider Lemma 3.2.3. This Lemma states that for a fixed demand pattern d, cost is a piecewise convex function of σ with two pieces. Therefore if P has a discrete distribution, $\mathbf{E}\,cost(\sigma|P)$ is a piecewise convex function of σ. (If the distribution has a finite support, then the number of convex pieces will also be finite.)

■

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 3.6.7 *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. Also, 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 3.6.6). The result follows. ■

Comment. The assumption $x_1 \leq 0$ is significant. Without the assumption we can prove a result similar to Lemma 3.6.7, except that the resulting algorithm will be exponential in K.

Lemma 3.6.8 *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 $(p^1, \ldots, p^k) \in \mathcal{P}$.*

Proof. Notice that the adversarial problem reduces to maximizing the following function.

$$\mathbf{E}\left[\sum_{i=1}^{T}(c_{i+1}d_i \;+\; \max\{h_i(\sigma - d_i), b_i(d_i - \sigma)\})\right]$$
$$= \sum_{i=1}^{T} \mathbf{E}\left[\, c_{i+1}d_i \;+\; \max\{h_i(\sigma - d_i), b_i(d_i - \sigma)\}\,\right]$$

Since each of the terms inside the second sum is a linear function of $(p^1, \ldots, p^k)$, the result follows. ■

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.

3.6.4 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. $h^k = (h^k_1, h^k_2, \ldots, h^k_T$ 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:

$$\min_{\pi \in \Pi} \max_{k} \max_{d \in \mathcal{D}^k} cost^k(\pi, d), \tag{3.45}$$

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

$$\min_{\pi \in \Pi} \sum_k \max_{d \in \mathcal{D}^k} cost^k(\pi, d). \tag{3.46}$$

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 (3.45) and (3.46) 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 (3.45) becomes more tractable than solving (1.1).

3.6.5 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.

3.7 Summary of the results

In this chapter we presented algorithms for solving the optimal robust basestock problem (3.1) using two different models for the demand uncertainty set $\mathcal{D}$. The algorithms are based on our generic Bender's decomposition procedure in Chapter 1.

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 *ambigious* uncertainty models (models where the demand distribution is stochastic but only partially known to the decision maker).

Chapter 4

The Dynamic Problem

In this chapter we will consider the robust inventory management problem in the dynamic setting. We allow our policy space to contain all policies, and thus in each period the inventory controller will have the freedom to determine the orders as an arbitrary function of the information that he has at that period. In other words, for each period, a policy amounts to a function that gives the order amount for any possible realization of the demands from previous periods. We assume that the inventory controller wants to minimize the cost over all such ordering functions. Instead of the generic algorithm that we introduced in Chapter 1, we use dynamic programming to solve this problem.

Let $(d_1, d_2, \ldots, d_{t-1})$ be the vector of demands up to period t. We denote the vector that contains information that one has at the beginning of period t prior to placing the order by $I_t = (x_t, d_1, d_2, \ldots, d_{t-1})$ and the set of all possible information vectors in period t is denoted by $\mathcal{I}_t$. Further, we denote our decision rule in period t for determining order quantities with $\pi_t(x_t, d_1, d_2, \ldots, d_{t-1})$. $\pi_t(x_t, d_1, d_2, \ldots, d_{t-1})$ is a function from $\mathcal{I}_t$ to real numbers and in every period t, it specifies the amount of the stock to be ordered in that period. We define the stock ordering policy, π to be the sequence of decision rules, i.e. $\pi = (\pi_t(.) : t = 1, \ldots, T)$. Moreover, we denote the set of all possible policies by Π. Notice that we can also think of the static policies and dynamic problems in this context. The Static policies in Chapter 2 correspond to the case in which the decision rule in each period t is defined by a constant function that is $\pi_t(x_t, d_1, d_2, \ldots, d_{t-1}) = u_t$ for a real number $u_t \geq 0$ for every $(x_t, d_1, d_2, \ldots, d_{t-1}) \in \mathcal{I}_t$ whereas constant basestock policies in Chapter 3 correspond to the case where the decision rule in each period t is of the form $\pi_t(x_t, d_1, d_2, \ldots, d_{t-1}) = \sigma - x_t$.

The robust inventory problem in the dynamic setting can be described as follows:

$$\min_{\pi \in \Pi} \; V(\pi) \tag{4.1}$$

where for $\pi \in \Pi$,

$$V(\pi) \;=\; \max_{d,x,u} \sum_{t=1}^{T} (\, c_t u_t \,+\, \max\{\, h_t x_{t+1} \,,\, -b_t x_{t+1} \}\,) \tag{4.2}$$

s.t.

$$u_1 \geq 0$$

$$u_t \;=\; \pi_t(d_1, d_2, ..., d_{t-1}), \quad 2 \leq t \leq T, \tag{4.3}$$

$$x_{t+1} \;=\; x_t + u_t - d_t, \quad 1 \leq t \leq T, \tag{4.4}$$

$$(d_1, d_2, \ldots, d_T) \;\in\; \mathcal{D}. \tag{4.5}$$

Here, x_1 is given and $\pi_t(.)$ is the ordering function specified by the policy π. We dropped x_t from the arguments of $\pi_t(.)$ since it is also a function of $(d_1, d_2, \ldots, d_{t-1})$ and the given x_1. Notice that (4.1) is of the same form as (3.1). The only difference lies in the policy space: instead of time-independent basestock policies, we minimize the cost over all policies.

In Section 4.2 we will give a characterization of the optimal policy. The optimal policy turns out to be a state dependent basestock policy where the state space in each period t is defined by $\mathcal{I}_t$. (In each period t, the optimal policy will be a basestock policy and the optimal basestock level will depend on the demand in prior periods). From a practical standpoint, such policies are very hard to compute and implement since the number of possible realizations of demand can be extremely large. However, this problem is of interest, because it provides a justification for the use of time and state independent basestock policy either by itself or in a rolling horizon fashion as an approximation for the pure dynamic policies which is the ideal thing to do.

This chapter is organized as follows: We first prove that problem 4.1 can be solved by dynamic programming in Section 4.1. In Section 4.2 we characterize the optimal policies. We show how to compute optimal policies for the two demand uncertainty sets in Sections 4.3 and 4.4.

4.1 Preliminaries

In this section we use dynamic programming to solve (4.1). We show the optimality of the robust version of the classical DP algorithm. Our result is in the same spirit as the results in [I05].

We start with introducing some new notation. We define the sequence of sets $\mathcal{P}_i = \{p \in \mathbf{R}^i \mid \exists\ d \ \in \ \mathbf{R}^{T-i} : \ (p, d) \ \in \ \mathcal{D}\}$ for $i = 1, 2, ..., T$ and $\mathcal{P}_T = \mathcal{D}$ where $\mathcal{D}$ is the uncertainty set we have and for $p \in \mathbf{R}^i$ and $d \in \mathbf{R}^{T-i}$. (p, d) denotes the T dimensional vector whose first i components are the components of p and the last $T-i$ components are d. Notice that $\mathcal{P}_i$ defines the set of all possible demand patterns up to time i. For each $p \in \mathcal{P}_{i-1}$, we define the set $\mathcal{D}_i(p) = \{d \ \in \ \mathbf{R}^{T-i+1} \mid (p, d) \ \in \ \mathcal{D}\}$ for $i = 2, ..., T$, that is $\mathcal{D}_i(p)$ is the projection of $\mathcal{D}$ onto $d_0 = p_0, d_1 = p_1, ..., d_i = p_i$ plane. For each $p \in \mathcal{P}_{i-1}$, we also define $D_i(p) = \{d_i \ \in \ \mathbf{R} | \ \exists\ d \in \mathbf{R}^{T-i} : \ (d_i, d) \in \mathcal{D}_i(p)\}$ for $i = 2, ..., T$ and $D_1 = \{d_1 \ \in \ \mathbf{R} | \ \exists\ d \in \mathbf{R}^{T-1} : \ (d_t, d) \in \mathcal{D}\}$. Moreover, let Π_t denote the set of policies restricted to the periods $t, t+1, \ldots T$.

For $i = 2, 3, ..., T$, $x \in \mathbf{R}$ and $p \in \mathcal{P}_{i-1}$, we define the following cost-to-go function.

$$J_i(x, p) = \min_{\pi \in \Pi_i} \ \tilde{V}_i(\pi, x, p) \tag{4.6}$$

where for $\pi \in \Pi_i$

$$\tilde{V}_i(\pi, x, p) \quad = \quad \max_{d,x,u} \sum_{t=i}^{T} (\, c_t u_t \, + \, \max\{\, h_t x_{t+1} \, , \; -b_t x_{t+1}\}\,) \tag{4.7}$$

s.t.

$$
\begin{aligned}
& u_i \geq 0 \\
& u_t \;=\; \pi_t(p, d_i, ..., d_t), \quad i+1 \leq t \leq T, \\
& x_i = x \\
& x_{t+1} \;=\; x_t + u_t - d_t, \quad i \leq t \leq T, \\
& (d_i, d_{i+1}, \ldots, d_T) \;\in\; \mathcal{D}_i(p).
\end{aligned}
$$

Moreover,

$$J_1(x) = \min_{\pi \in \Pi} \quad \tilde{V}_1(\pi, x) \tag{4.8}$$

where for every $\pi \in \Pi$, $\tilde{V}_1(\pi, x)$ is defined similar to (4.7).

$J_t(x, p)$ gives the minimum cost that one can attain in periods $t, t+1, \ldots T$ assuming that the starting inventory at the beginning of period t is x and the demand in periods up to $t-1$ is given by p. Theorem 4.1.1 below shows that the problem of computing $J_t(x, p)$ and finding the optimal policy can be solved using dynamic programming.

Theorem 4.1.1 *The set of functions $\{J_t \quad t = 1, ..., T\}$ satisfies the following robust Bellman equations.*

$$J_1(x) \;=\; \min_{u_1 \geq 0} \max_{d_1 \in D_1} \{c_1 u_1 + W_1(x + u_1 - d_1) + J_2(x + u_1 - d_1, d_1)\} \tag{4.9}$$

and for $2 \leq t \leq T$ and $p \in \mathcal{P}_{t-1}$

$$J_t(x, p) \;=\; \min_{u_t \geq 0} \max_{d_t \in D_t(p)} \{c_t u_t + W_t(x + u_t - d_t) + J_{t+1}(x + u_t - d_t, (p, d_t))\} \tag{4.10}$$

and for $p \in \mathcal{D}$

$$J_{T+1}(x,p) = 0. \tag{4.11}$$

Proof. Fix $t > 1$. Let $\pi^* \in \Pi_t$ be the order vector minimizing $\tilde{V}_t(x,\pi,p)$ for $p \in \mathcal{P}_{t-1}$. We define π^*_{t+1} be the sub-policy of π^* restricted to periods $t+1,...,T$. Note that

$$\begin{aligned}
J_t(x,p) &= \max_{d\in\mathcal{D}_t(p)} \{c_t u_t^* + W_t(x_t + u_t^* - d_t) \\
&\quad + \sum_{i=t+1}^{T} c_i \pi_i^*(p,d_t,...,d_{i-1}) + W_i(x_i + \pi_i^*(p,d_t,...,d_{i-1}) - d_i)\} \\
&= \max_{d_i\in D_t(p)} \{c_t u_t^* + W_t(x_t + u_t^* - d_t) \\
&\quad + \max_{\hat{d}\in\mathcal{D}_{t+1}(p,d_t)} \left\{ \sum_{i=t+1}^{T} c_i \pi_i^*(p,d_t,...,d_{i-1}) + W_i(x_i + \pi_i^*(p,d_t,...,d_{i-1}) - \hat{d}_i)\} \right\} \} \\
&\geq \max_{d_t\in D_t(p)} \{c_t u_t^* + W_t(x_t + u_t^* - d_t) \\
&\quad + \min_{\pi\in\Pi_{t+1}} \max_{\hat{d}\in\mathcal{D}_{t+1}(p,d_t)} \left\{ \sum_{i=t+1}^{T} c_i \pi_i(p,\hat{d}_t,...,\hat{d}_{i-1}) + W_i(x_i + \pi_i(p,\hat{d}_t,...,\hat{d}_{i-1}) - \hat{d}_i) \right\} \} \\
&= \max_{d\in D_t(p)} \{c_t u_t^* + W_t(x_t + u_t^* - d_t) + J_{t+1}(x_t + u_t^* - d_t, (p,d_t))\} \\
&\geq \min_{u_t\geq 0} \max_{d\in D_t(p)} \{c_t u_t + W_t(x_t + u_t - d_t) + J_{t+1}(x_t + u_t - d_t, (p,d_t))\} \qquad (4.12)
\end{aligned}$$

where for $i = t+1,...,T$, $\pi_t^*(.)$ is the ordering function that is defined by π^* and $u_t^* = \pi_t^*(p)$.

Let π^* be the policy that minimizes $\tilde{V}_t(\pi,x,p)$ and π^*_{t+1} be the sub-policy of π^* restricted to periods $t+1,\ldots,T$. Moreover, let $\bar{\pi}$ be the optimal policy for $\tilde{V}_{t+1}(\pi,x,p)$ with $\pi_{t+1}(.)$ defined for all values of (x,p). Then we have

$$\begin{aligned} J_t(x,p) &= \max_{d_t \in D_t(p)} \{c_i u_t^* + W(x+u_t^* - d_t) + \tilde{V}_{t+1}(x+u_t^* - d_t, \pi_{t+1}^*, (p,d_t))\} \\ &\leq \min_{u_t \geq 0} \max_{d \in D_t(p)} \{c_i u_t + W(x+u_t-d_t) + \tilde{V}_{t+1}(x+u_t-d_t, \bar{\pi}_{t+1}, (p,d_t))\} \\ &= \min_{u_t \geq 0} \max_{d \in D_t(p)} \{c_i u_t + W(x+u_t-d_t) + J_{t+1}(x+u_t-d_t, (p,d_t))\} \quad (4.13) \end{aligned}$$

Notice that (4.12) with (4.13) proves the result for $t > 1$. Proof for $t = 1$ follows similarly. ■

4.2 Characterization of optimal policies

In this section we will give a characterization of the optimal dynamic policies using the recursive equations (4.9)-(4.11). Notice that the cost-to-go functions defined in Theorem 4.1.1 can equivalently expressed as follows.

$$J_t(x,p) = -c_t x + \min_{y \geq x} \max_{d_t \in D_t(p)} \{c_t y + W(y-d_t) + J_{t+1}(y-d_t,(p,d_t))\} \qquad (4.14)$$

for $t > 1$, $p \in \mathcal{P}_{t-1}$ and $x \in \mathbf{R}$. In addition, for $t = 1$,

$$J_1(x) = -c_1 x + \min_{y \geq x} \max_{d_1 \in D_1} \{c_1 y + W(y-d_1) + J_2(y-d_1,(p,d_1))\}. \qquad (4.15)$$

For $1 < t \leq T$ we define

$$G_t(y,p,d) = c_t y + W_t(y-d) + J_{t+1}(y-d,(p,d))$$

and

$$H_t(y,p) = \max_{d \in D_t(p)} \{G_t(y,p,d)\}.$$

For $t = 1$ we define $G_1(y, d)$ and $H_t(y)$ in a similar way. We have the following theorem.

Theorem 4.2.1

(a) For fixed t and $p \in \mathcal{P}_{t-1}$, $H_t(y, p)$ is a convex function of y.

(b) For the first period the optimal policy is a basestock policy. For each period $t > 1$ and each $p \in \mathcal{P}_{t-1}$ there exists a basestock level, $S_{t,p}$ such that it is optimal to order $S_t - x_t$ if $S_t > x_t$ and it optimal not to order when $S_t \leq x_t$ where x_t is the inventory at the beginning of period t.

(c) For $t > 1$ $J_t(x, p)$ is a convex function of x for fixed p and $J_1(x)$ is a convex function of x.

Proof. We prove the theorem by induction. For $t = T$ and fixed $p \in \mathcal{P}_{T-1}$,

$$G_T(y, p, d) = c_T y + W(y - d)$$

is a convex function of y. Since $H_T(y, p)$ is the maximum of a set of convex functions it is also convex. Let $S_T(p)$ be the smaller minimizer of $H_T(y, p)$ for fixed p. Note that $y = x$, and $y = S_T(p)$, minimize $H_t(y, p)$ over the set $\{y : \ y \geq x\}$ when $x > S_T$ and $x \leq S_T(p)$, respectively, and this proves part (b). Convexity of $J_T(x, p)$ with respect to x follows from the fact that $J_T(x) = -cx + H_T(\max(S_T, x), p)$, which is the sum of a linear function and the composition of a convex function with another convex function.

Now assume that theorem is true for $t+1$. Consider $G_t(y,p,d)$ for fixed p and d. Note that first two terms are a constant and a piecewise linear, convex function. Moreover, from the induction hypothesis $J_{t+1}(x,(p,d_t))$ is a convex function of x, so it is also a convex function of y. Consequently, $H_t(y,p)$ is a convex function of y for fixed p.

Part (b) and (c) follows from part (a) and as in the base case. Proof for $t=1$ also follows similarly. ■

Theorem 4.2.1 states that in each period t, for every realization of the past demands we have a different basestock level. The reason for this is that our uncertainty set in each period t, $\mathcal{D}_t(p)$ depends on the vector of past demands, p. For different $p \in \mathcal{P}_{t-1}$, the available demand patterns for the adversary may be different. Therefore, for each policy, the future demand that maximizes cost may be different for different realizations of the past demand which clearly means that optimal policy will be dependent on past demand.

For each period t the demand vector p takes values from $\mathcal{P}_{t-1}$ which may have infinitely many elements. In order to solve the problem, we may have to compute an infinite number of basestock levels for every possible value of p. However, notice that the optimal policy depends on p only through $\mathcal{D}_t(p)$ in each period t. Therefore for any vectors $p_1 \neq p_2$, the optimal policies for these two vectors are the same if $\mathcal{D}_t(p_1) = \mathcal{D}_t(p_2)$. We will use this fact in order to decrease the number of optimal basestocks we have to compute.

To solve the dynamic program in Theorem 4.1.1, we have compute the cost-to-go function in for all values of the state variable which, for each period t, consists of (x,p) such that $x \in \mathbf{R}$ and $p \in \mathcal{P}_{t-1}$. Although we may be able to diminish the size of our state space by using some properties of our uncertainty set, it may still be very large in most cases and directly applying the dynamic programming algorithm will be impractical due to its computational burden. However, the optimality of state dependent basestock policies gives a justification for the use of time and state independent basestock policies as an approximation.

One advantage of using the dynamic approach is that we can incorporate a fixed ordering cost into our model without too much a large increase the computational complexity. In fact, using the dynamic programming approach one can prove that in the fixed cost case, the optimal policy is a two parameter basestock policy (an (S,s) policy) where the basestock parameters again depend on the state variables.

4.3 Risk budgets model

In this section we will show how to solve the dynamic problem for the demand model (2.3)-(2.5).

Let d and $\bar{d}$ two demand patterns in our demand uncertainty sets, and z and $\bar{z}$ the corresponding vectors defined according to the equation (2.3). Let d^t, $\bar{d}^t$, z^t and

$\bar{z}^t$ be the subvectors restricted to periods $1, \ldots, t$ for some $t < T$. If

$$\sum_{i=1}^{t} z_i^t = \sum_{i=1}^{t} \bar{z}_i^t$$

we have $\mathcal{D}_{t+1}(d^t) = \mathcal{D}_{t+1}(\bar{d}^t)$, so $J_{t+1}(x, d^t) = J_{t+1}(x, \bar{d}^t)$ for any x. Therefore we can replace the second argument which is the past demand in the cost-to-go function with the total risk consumed up to that period (which is equal to the sum of z's). As a result the dynamic programming recursion turns into the following form.

$$J_1(x) \quad = \quad \min_{u_1 \geq 0} \max_{d_1 \in D_1} \left\{ c_1 u_1 + W(x + u_1 - d_1) + J_1(x + u_1 - d_1, \left| \frac{d_1 - \mu_1}{\delta_1} \right|) \right\}$$

and for $2 \leq t \leq T$ and $0 \leq k \leq \Gamma_{t-1}$

$$J_t(x, k) \quad = \quad \min_{u_t \geq 0} \max_{d_t \in \tilde{D}_t(k)} \{ c_t u_t + W(x + u_t - d_t) + J_{t+1}(x + u_t - d_t, k + \left| \frac{d_t - \mu_t}{\delta_t} \right|) \}$$

and for $0 \leq k \leq \Gamma_T$

$$J_{T+1}(x, k) = 0.$$

where $\tilde{D}_t(k)$ is defined similarly to $D_t(p)$ which is defined in Section 4.1. Here $J_t(x, k)$ for $t > 1$ gives the optimal total cost for periods $t, \ldots, T$ assuming that the risk used up until period $t - 1$ is k and the starting inventory in period t is x.

Although our state space has two dimensions now, we still have to compute compute the cost-to-go function for any value of risk usage which is in $[0, \Gamma_t]$ which means that we can only approximate the true cost-to-go by discretization. But once we discretize k, it is possible to show that for a fixed risk consumption the cost-to-go function is piecewise linear in x, and therefore as in the algorithm for the adversarial problem in Chapter 2, we can compute the function by evaluating each of its pieces.

4.3.1 A special case

The dynamic programming algorithm proves to be efficient when uncertainty set is described by the condition $d_t \in [\mu_t - \delta_t, \mu_t + \delta_t]$ (which means $\Gamma_t = t$ in our risk budgets context). This model is the box model that we considered in Section 2.4.1. Notice that we can get rid of the second argument in the cost-to-go function, since no matter what the past demand is, we always have the same uncertainty set for the future periods and $D_t(p) = [\mu_t - \delta_t, \mu_t + \delta_t]$ for every p and t. In fact, due to convexity of the cost function, it is easy to see that in the optimal solution, the adversary sets either $d_t = \mu_t - \delta_t$ or $d_t = \mu_t + \delta_t$ for each t. Therefore, the cost-to-go functions turns out to be as follows. For $1 \leq t \leq T$

$$J_t(x) \quad = \quad \min_{u_t \geq 0} \max_{d_t \in \{\mu_t - \delta_t, \mu_t + \delta_t\}} \{c_t u_t + W(x + u_t - d_t) + J_{t+1}(x + u_t - d_t)\}$$

and $J_{T+1}(x) = 0$.

Using Lemma 2.4.4, we have the following result:

Lemma 4.3.1

(a) For every t, $J_t(x)$ is a piecewise linear convex function with $T - t + 2$ pieces.

(b) The dynamic program for the box model can solved in $O(T^2)$ time.

We want to emphasize the fact that the optimal policy in the dynamic setting is a state dependent basestock policy. The reason for the dependence on state space is due to the dependence of future demands that is picked by the adversary on the past demands. In the box uncertainty case, however, the adversary's behavior behavior is independent from the demand history. Thus, basestock levels will only depend on time, not the state of the system.

4.4 Bursty demand model

In this section we consider the adversarial problem for the bursty demand model given in Section 2.2. We can adapt the dynamic programming recursion used for the risk budgets model as follows.

For each period t, and each integer $1 \leq k < \min\{W, t\}$, let $\Pi_t(x, k)$, denote the minimum cost attainable 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 minimum cost attainable 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 $1 \leq t \leq T$:

$$\Pi_t(x,k) \quad = \quad \min_{u_t \geq 0} \max_{d \in \{\mu_t - \delta_t, \mu_t + \delta_t\}} \left\{ c_t u_t + \mathcal{W}_t(x + u_t - d) + \Pi_{t+1}(x + u_t - d, k+1) \right\},$$

for $1 \leq k < \min\{W-1, t\}$

$$\Pi_t(x, W-1) = \min_{u_t \geq 0} \max_{d \in \{\mu_t - \delta_t, \mu_t + \delta_t\}} \left\{ c_t u_t + \mathcal{W}_t(x + u_t - d) + \Pi_{t+1}(x + u_t - d, 0) \right\},$$

for $W - 1 < t$,

$$\begin{aligned} \Pi_t(x, 0) &= \min_{u_t \geq 0} \max \left\{ \Pi_t^1(x), \Pi_t^0(x) \right\}, \quad \text{where} \\ \Pi_t^1(x) &= \min_{u_t \geq 0} \max_{d \in \{\mu_t - \delta_t, \mu_t + \delta_t\}} \left\{ c_t u_t + \mathcal{W}_t(x + u_t - d) + \Pi_{t+1}(x + u_t - d, 0) \right\}, \\ \Pi_t^0(x) &= c_t u_t + \mathcal{W}_t(x + u_t - P_t) + \Pi_{t+1}(x + u_t - P_t, 1). \end{aligned}$$

To solve the recursions above one can use the same approach with the budgets model. For each k we compute and store the piecewise linear representation of $\Pi_t(x, k)$. Moreover, from Theorem 4.2.1, the optimal policy is a basestock policy that in each period t, depends on the last peak that occurred before t.

Appendix A

An alternative approach for solving P_M

In this section we present an alternative method to solve problem $P_M(\gamma, t^*, t^f, t^e)$, iteratively for different values of γ and t^e. Originally, in Section 3.3.4 we described a dynamic programming method to solve the problem. Note that in the formulation in section 3.3.1 the right hand side values of (3.14)-(3.16) are integral. For simplicity, we re-index the problem and set $\hat{\Gamma}_{i-t^*+1}(\gamma) = \lfloor \Gamma_i - \gamma \rfloor$ for $i \in \{t^*, ..., t^f - 2\}$, $\hat{\Gamma}_{t^f - t^*}(\gamma) = \min\{\lfloor \Gamma_{t^f} - (\gamma + l(\gamma, t^e)) \rfloor, \lfloor \Gamma_{t^f - 1} - \gamma \rfloor\})$, and $\hat{\Gamma}_{i-t^*}(\gamma) = \lfloor \Gamma_i - (\gamma + l(\gamma, t^e)) \rfloor$ for $i \in \{t^f + 1, ..., t^e\}$. If (d^*, z^*) is an optimal solution, then $d^*_t = \mu_t - z_t \delta_t$ if and only if

$$h_t(\sigma - (\mu_t - z_t\delta_t)) + c_{t+1}(\mu_t - z_t\delta_t) \geq b_t(\mu_t + z_t\delta_t - \sigma) + c_{t+1}(\mu_t + z_t\delta_t)$$

for $t \in M \cup F$. We use this to eliminate the d variables and write the cost corresponding to each demand variable as

$$\hat{cost}_t(\sigma, z_t) = \max\{h_t(\sigma - (\mu_t - z_t\delta_t)) + c_{t+1}(\mu_t - z_t\delta_t), b_t(\mu_t + z_t\delta_t - \sigma) + c_{t+1}(\mu_t + z_t\delta_t)\}$$

Then the problem turns into the following form.

$$\begin{array}{rll} \hat{P_M}(\gamma, t^*, t^f, t^e) = Max & \sum_{i=1}^{t^e - t^*} \hat{cost}_t(\sigma, z_t) & \\ s.t. & \sum_{j=t^*}^{i} z_i \le \hat{\Gamma}_i(\gamma) & 1 \le i \le t^e - t^* \\ & 0 \le z_i \le 1 & 1 \le i \le t^e - t^* \end{array}$$

Lemma A.0.1 *For any $\gamma \in \mathbf{R}$, t^*, t^f, and t^e, there exists an integral optimal solution, $z_1, ..., z_{t^e - t^*}$, to $\hat{P_M}(\gamma, t^*, t^f, t^e)$.*

Proof. Note that the objective function is a convex function. Since we are maximizing an objective function over a polyhedron, there exist an optimal solution that is an extreme point. Suppose that z is the optimal extreme point and suppose that it is not integral. Let t_1 be the smallest time period for which z_{t_1} is fractional. There are two cases to consider. If there is no other time period for which the risk is fractional, $z_1 + z_2 + ... + z_t < \hat{\Gamma}_t(\gamma)$ for all $t \in \{t_1, ..., T\}$. We form two solutions, by setting $z_{t_1} = 1$ and $z_{t_1} = 0$. z can be written as a convex combination of these to solutions which is a contradiction. If there is more than one fractional variable, we consider the second smallest index, t_2, for which z_{t^2} is fractional. Note that $z_1 + z_2 + ... + z_t < \hat{\Gamma}_t(\gamma)$

for all $t \in \{t_1, ..., t_2 - 1\}$. Therefore we can form two solutions by perturbing z_{t^1} and z_{t^2} and write z as a linear combination of these two solutions. ■

The discussion of our algorithm in Section 3.3.4 implies that we can solve $\hat{P_M}(k + f_{(j)}, t^*, t^f, t^e)$ for every possible value of $k + f_{(j)}$ to solve the adversarial problem. Our purpose in this section is to show a method to derive the solution to $P_M(.)$ for $k + f_{(j+1)}$ from the solution for $k + f_{(j)}$ so that we will not have to solve $P_M(.)$ each time from scratch. Notice that as we increase j to $j + 1$ two things may happen:

1. $\hat{\Gamma}_m(k + f_{(j+1)}) = \hat{\Gamma}_m(k + f_{(j)}) - 1$ for some $m \in \{1, 2, ..., t^f - t^*\}$ and $\hat{\Gamma}_t(k + f_{(j+1)}) = \hat{\Gamma}_t(k + f_{(j)})\ \forall t \in \{1, 2, ..., t^e - t^*\} \backslash \{m\}$.

2. $\hat{\Gamma}_t(k + f_{(j+1)}) = \hat{\Gamma}_m(k + f_{(j)}) - 1\ \forall t \in \{t^f - t^*, ..., t^e - t^*\}$ and $\hat{\Gamma}_t(k + f_{(j+1)}) = \hat{\Gamma}_t(k + f_{(j)})\ \forall t \in \{1, 2, ..., t^f - t^* - 1\}$

The following Lemma suggests a method to derive the solution for $\hat{P_M}(k+f_{(j+1)}, t^*, t^f, t^e)$ from the solution for $\hat{P_M}(k + f_{(j)}, t^*, t^f, t^e)$.

Lemma A.0.2 *Let $\gamma_1 \leq \gamma_2$. Suppose that $\hat{\Gamma}_t(\gamma_1) = \hat{\Gamma}_t(\gamma_2)+1$ for some $1 \leq t \leq t^e - t^*$ and $\hat{\Gamma}_i(\gamma_1) = \hat{\Gamma}_i(\gamma_2)\ \forall i \in \{1, 2, ..., t^e - t^*\} \backslash \{t\}$. Let z^1 be the optimal solution to $\hat{P_M}(\gamma_1, t^*, t^f, t^e)$. Then, there exist an optimal solution z^2 to $\hat{P_M}(\gamma_2, t^*, t^f, t^e)$ and such that there exist two periods $t_1 \leq t \leq t_2$ for which $z^2_{t_1} \leq z^1_{t_1}$, $z^2_{t_2} \geq z^1_{t_2}$ and $z^2_i = z^1_i$ for the rest of the indices.*

Proof. Note that if $\sum_{i=1}^t z^1_i < \hat{\Gamma}_t(\gamma_1)$ then z^1 is also optimal for $\hat{P_M}(\gamma_2, t^*, t^f, t^e)$. Therefore we assume that $\sum_{i=1}^t z^1_i = \hat{\Gamma}_t(\gamma_1)$ Let z^2 be an arbitrary optimal solution to $\hat{P_M}(\gamma_2, t^*, t^f, t^e)$. Since $\sum_{i=1}^{t} z_i^2 \leq \hat{\Gamma}_t(\gamma_2) = \hat{\Gamma}_t(\gamma_1) - 1$, there exists $1 \leq i \leq t$ such that $z_i^1 > z_i^2$. Let

$$t_1 = \max\{i \in \{1, ..., t\} | z_i^1 > z_i^2\}.$$

We consider the following two cases.

Case 1: $t = t^e - t^*$

We form a new solution $\overline{z}^1$ to $\hat{P_M}(\gamma_1, t^*, t^f, t^e)$, by setting $\overline{z}_{t_1} = z_{t_1}^1 = 1$ and $\overline{z}_i^1 = z_i^2$ for $i \in \{1, ..., t^e - t^* - 1\}$. To prove the feasibility of $\overline{z}^1$ note that for $1 \leq k \leq t_1 - 1$

$$\sum_{i=1}^{k} \overline{z}_i^1 = \sum_{i=1}^{k} z_i^2 \leq \hat{\Gamma}_k(\gamma_2) = \hat{\Gamma}_k(\gamma_1).$$

For t_1

$$\sum_{i=1}^{t_1} \overline{z}_i^1 \quad = \quad \sum_{i=1}^{t_1-1} z_i^2 + 1 \leq \sum_{i=1}^{t_1-1} z_i^1 + 1 \leq \hat{\Gamma}_{t_1}(\gamma_1)$$

To see that the first inequality is true, suppose that $\sum_{i=1}^{t_1-1} z_i^2 > \sum_{i=1}^{t_1-1} z_i^1$. Then from the definition of t_1 we have $\sum_{t_1}^{t} z_i^2 + 1 \geq \sum_{t_1}^{t} z_i^1$. Therefore, we have

$$\hat{\Gamma}_t(\gamma_1) - 1 = \hat{\Gamma}_t(\gamma_2) \geq \sum_{i=1}^{t} z_i^2 > \sum_{i=1}^{t} z_i^1 - 1 = \hat{\Gamma}_t(\gamma_1) - 1$$

which is a contradiction. For $t_1 + 1 \leq k \leq t$

$$\sum_{i=1}^{k} \overline{z}_i^1 \quad = \quad \sum_{i=1}^{t_1-1} z_i^2 + 1 + \sum_{t_1+1}^{t} z_i^2 \leq \sum_{i=1}^{t_1-1} z_i^1 + 1 + \sum_{t_1+1}^{t} z_i^1 \leq \hat{\Gamma}_{t_1}(\gamma_1).$$

The first inequality above follows similarly to the previous case.

The cost of this solution is equal to

$$\sum_{i=1}^{t^e-t^*} \hat{cost}_i(\sigma, \overline{z}_i^1) = \sum_{i\in\{1,...t^e-t^*\}\setminus\{t_1\}} \hat{cost}_i(\sigma, z_i^2) + \hat{cost}_{t_1}(\sigma, z_{t_1}^1) \leq \sum_{i\in\{1,...t^e-t^*\}} \hat{cost}_i(\sigma, z_i^1)$$

$$\Rightarrow \sum_{i\in\{1,...t^e-t^*\}\setminus\{t_1\}} \hat{cost}_i(\sigma, z_i^2) \leq \sum_{i\in\{1,...t^e-t^*-1\}} \hat{cost}_i(\sigma, z_i^1) \tag{A.1}$$

Now consider the solution $\overline{z}^2$ to $\hat{P_M}(\gamma_2, t^*, t^f, t^e)$, formed by setting $\overline{z}_{t_1}^2 = z_{t_1}^2 = 0$ and $\overline{z}_i^2 = z_i^1$ for $\{1, ..., t^e - t^* - 1\}$. For $1 \leq k \leq t_1 - 1$

$$\sum_{i=1}^{k} \overline{z}_i^2 = \sum_{i=1}^{k} z_i^1 \leq \hat{\Gamma}_k(\gamma_1) = \hat{\Gamma}_k(\gamma_2).$$

For $k \geq t_1$

$$\sum_{i=1}^{k} \overline{z}_i^2 = \sum_{i=1}^{k} z_i^1 - 1 \leq \hat{\Gamma}_k(\gamma_1) - 1 \leq \hat{\Gamma}_{t_1}(\gamma_2).$$

The cost of this solution satisfies

$$\sum_{i=1}^{t^e-t^*} \hat{cost}_i(\sigma, \overline{z}_i^2) = \sum_{i\in\{1,...t^e-t^*\}\setminus\{t_1\}} \hat{cost}_i(\sigma, z_i^1) + \hat{cost}_{t_1}(\sigma, \overline{z}_{t_1}^2) \geq \sum_{i\in\{1,...,t^e-t^*\}} \hat{cost}_i(\sigma, z_i^2)$$

Therefore, $(\overline{d}^2, \overline{z}^2)$ is optimal.

Case 2: $t < t^e - t^*$

We define

$$t_2 = \min\{i \in \{t+1, ..., t^e - t^*\} | z_i^1 < z_i^2\}$$

Such an index exists since otherwise

$$\sum_{i=1}^{k} z_i^2 < \hat{\Gamma}_t(\gamma_1) - 1 + \sum_{i=t+1}^{k} z_i^2 \leq \hat{\Gamma}_t(\gamma_1) - 1 + \sum_{i=t+1}^{k} z_i^1 \leq \hat{\Gamma}_k(\gamma_1) - 1 < \hat{\Gamma}_k(\gamma_2)$$

for $t+1 \leq k \leq t^e - t^*$ and this means that z^2 is not optimal.

As in the previous case we form a new solution, $\overline{z}^1$ for $\hat{P_M}(\gamma_1, t^*, t^f, t^e)$. We set $\overline{z}^1_{t_1} = z^1_{t_1} = 1$, $\overline{z}^1_{t_2} = z^1_{t_2} = 0$ and $\overline{z}^1_i = z^2_i$ for $\{1, ..., t^e - t^* - 1\}$. For $1 \leq k \leq t$ the proof for $\sum_{i=1}^k \overline{z}^1_i \leq \hat{\Gamma}_k(\gamma_1)$ follows from the same argument used in the previous case. For $t < k < t_2$

$$\begin{aligned}\sum_{i=1}^k \overline{z}^1_i = \sum_{i=1}^t \overline{z}^1_i + \sum_{i=t+1}^k z^2_i &\leq \hat{\Gamma}_t(\gamma_1) + \sum_{i=t+1}^k z^1_i \\ &= \sum_{i=1}^t z^1_i + \sum_{i=t+1}^k z^1_i \leq \hat{\Gamma}_k(\gamma_1)\end{aligned}$$

and for $k \geq t^2$

$$\sum_{i=1}^k \overline{z}^1_i = \sum_{i=1}^k \overline{z}^1_i = \sum_{i=1}^k z^2_i \leq \hat{\Gamma}_k(\gamma_2) = \hat{\Gamma}_k(\gamma_1)$$

The cost corresponding to that solution is

$$\sum_{i \in \{1, ... t^e - t^*\} \setminus \{t_1, t_2\}} c\hat{o}st_i(\sigma, z^2_i) + c\hat{o}st_{t_1}(\sigma, z^1_{t_1}) + c\hat{o}st_{t_2}(\sigma, z^1_{t_2}) \leq \sum_{i \in \{1, ... t^e - t^*\}} c\hat{o}st_i(\sigma, z^1_i)$$

$$\Rightarrow \sum_{i \in \{1, ... t^e - t^*\} \setminus \{t_1, t_2\}} c\hat{o}st_i(\sigma, z^2_i) \leq \sum_{i \in \{1, ... t^e - t^*\} \setminus \{t_1, t_2\}} c\hat{o}st_i(\sigma, z^1_i)$$

Consider the solution, $\overline{z}^2$ to $\hat{P_M}(\gamma_2, t^*, t^f, t^e)$ such that $\overline{z}^2_{t_1} = z^2_{t_1} = 0$, $\overline{z}^2_{t_2} = z^2_{t_2} = 1$ and $\overline{z}^2_i = z^1_i$ for $\{1, ..., t^e - t^* - 1\}$. The proof for feasibility of $\overline{z}^2$ is as follows. For $1 \leq k \leq t$ same argument in case 1 holds. For $t < k < t_2$

$$\sum_{i=1}^k \overline{z}^2_i = \sum_{i=1}^k z^1_i - 1 \leq \hat{\Gamma}_k(\gamma_1) - 1 \leq \hat{\Gamma}_{t_1}(\gamma_2)$$

and for $k \geq t_2$

$$\sum_{i=1}^k \overline{z}^2_i = \sum_{i=1}^k z^1_i \leq \hat{\Gamma}_k(\gamma_1) = \hat{\Gamma}_k(\gamma_2).$$

The cost of this solution satisfies

$$\begin{aligned}\sum_{i=1}^{t^e-t^*} \hat{cost}_i(\sigma, \overline{z}_i^2) &= \sum_{i\in\{1,\ldots t^e-t^*\}\setminus\{t_1,t_2\}} \hat{cost}_i(\sigma, z_i^1) + \hat{cost}_{t_1}(\sigma, z_{t_1}^2) + \hat{cost}_{t_2}(\sigma, z_{t_2}^2) \\ &\geq \sum_{i\in\{1,\ldots,t^e-t^*\}} \hat{cost}_i(\sigma, z_i^2)\end{aligned}$$

Therefore, $\overline{z}^2$ is optimal. ■

Lemma A.0.2 states that if right hand side of one of the budget constraints, say the t^{th} constraint, drops by one we can compute the new solution from the old solution greedily. To construct the solution for $\hat{P_M}(k + f_{(j+1)}, t^*, t^f, t^e)$, from the solution $\hat{P_M}(k + f_{(j)}, t^*, t^f, t^e)$ for the given k, j, t^*, t^f and t^e, we consider the indices for which $\hat{\Gamma}(k + f_{(j)}) = \hat{\Gamma}(k + f_{(j+1)}) + 1$. Starting from the smallest such index we apply our greedy method.

Appendix B

NP-completeness proofs

B.1 Proof of Theorem 3.4.1

Theorem 3.4.1 states that the adversarial problem for the bursty demand model with basestock policies is NP-hard. However, it also states that it is not strongly NP-hard and for any given $\epsilon > 0$, an ϵ-approximate solution can be computed in time that is polynomial in T and $1/\epsilon$.

Part (a): We will use the following problem for to obtain the NP-hardness reduction.

Problem B.1.1 *Suppose we have a set of n positive integers $A = \{a_1, a_2, ..., a_n\}$ and a positive integer K. Is there a subset of A whose sum is exactly K?*

Given an instance of Problem B.1.1 we will transform it into an instance of our adversarial problem as follows:

We assume that $w = 0$, $\mu_t = 0$ and $\delta_t = 0$ for every t. The problem has $n+4$ periods. In period i, $1 \leq i \leq n$, $p_i = a_i$ (so in such periods the demand will be a_i if there is a peak, or zero if there is no peak). Also, in these periods all costs are zero. Moreover,

- in period $t = n + 1$, $p_t = 0$,
- in period $t = n + 2$, $p_t = 1$,
- in period $t = n + 3$, $p_t = K - 1$,
- in period $t = n + 4$, $p_t = 0$.

Also, in period $n+1$, the inventory holding cost is 1. In period $n+4$, the production cost is a large value M such that $M > \sum_i a_i$. All other costs are equal to 0. The initial inventory equals the sum of all a_i. Finally, $\sigma = K - 1$.

Now, we have the following.

Lemma B.1.2 *Suppose there is a subset S of $1, 2, ..., n$ such that $\sum_{i \in S} a_i = K$. Then there is a demand pattern whose cost equals $K + M * (K - 1)$.*

Proof. For each $i \notin S$, we set the demand to a_i, and for all other $1 \leq i \leq n$ we set the demand to zero. In period $n + 1$ we set the demand to zero, in period $n + 2$ we set it to 1, in period $n + 3$ we set it to $K - 1$, and in period $n + 4$ we set it to zero.

By definition of the set S, we have that $x_1 - \sum_{i=1}^{n} d_i = K$ and $x_1 - \sum_{i=1}^{t} d_i \geq K$ for each $1 \leq t \leq n$. So the inventory holding cost in period $n + 1$ equals K. So

$x_1 - \sum_{i=1}^{n+1} d_i = K$, and thus $x_1 - \sum_{i=1}^{n+2} d_i = K - 1 (= \sigma)$. So $x_1 - \sum_{i=1}^{n+3} d_i = 0$, and in period $n+4$ we incur a production cost of $M * (K-1)$. ■

We also have,

Lemma B.1.3 *Suppose we consider a demand pattern such that $x_1 - \sum_{i=1}^{n} d_i \leq K - 1$. Then its cost is at most $K - 1 + M * (K-1)$.*

Proof. Since $x_1 - \sum_{i=1}^{n} d_i \leq K - 1$, the inventory holding cost in period $n+1$ is less than or equal to $K-1$. And, also, since $\sigma = K - 1$, we have that $x_1 - \sum_{i=1}^{n+2} d_i \leq K - 1$, and so $x_1 - \sum_{i=1}^{n+3} d_i \geq 0$, and so the production cost paid in period $n+4$ is at most $M * (K-1)$. ■

Lemma B.1.4 *Suppose we consider a demand pattern such that $x_1 - \sum_{i=1}^{n} d_i \geq K + 1$. Then its cost is less than $K + M * (K-1)$.*

Proof. The inventory holding cost paid in period $n+1$ is at most $x_1 - \sum_{i=1}^{n} d_i$ which is at most $\sum_i a_i$.

Since the demand in $n+1$ is always zero, $x_1 - \sum_{i=1}^{n+1} d_i = x_1 - \sum_{i=1}^{n} d_i \geq K + 1$, and so $x_1 - \sum_{i=1}^{n+2} d_i \geq K (> \sigma)$. So $x_1 - \sum_{i=1}^{n+3} d_i \geq 1$, and the production cost paid in period $n+4$ is at most $M * (K-2)$.

In summary, the total cost is at most $\sum_i a_i + M * (K-2)$ which is strictly less than $K + M * (K-1)$ by definition of M. ■

Finally,

Lemma B.1.5 *Suppose we consider a demand pattern such that $x_1 - \sum_{i=1}^{n} d_i = K$. Then there is a subset of the a_i with sum K.*

Proof. Since $\sigma = K - 1$, this follows easily. Use the subset defined by the time periods (up to n) where the peaks are not used. ■

In summary, there is a subset S of the A with sum K if and only if there is a demand pattern with cost $K + M * (K - 1)$.

B.2 Proof of Theorem 3.6.1

We will prove the theorem using the maximum stable set problem in the complement of the graph $G = (V, E)$. Let G' denote the complement. For an arbitrary graph G we construct the following robust basestock problem with $|V| + 1$ periods:

- For periods $t = 1, 2, ..., |V|$, we set $h_t = 0$, $b_t = 1$, $c_t = 0$ and $0 \leq d_t \leq 2$,
- for period $t = |V| + 1$, we set $h_t = M$, $b_t = M$, $c_t = 0$ and $d_t = 1$,
- for each edge (i, j) in graph G', we have $(d_i + d_j)/2 \leq 1$,
- we set $x_1 = 0$.

Notice that for M large enough ($M > |V| + 1$), $\sigma = 1$ gives the optimal solution. We have the following Lemma that gives a characterization of the demand pattern that gives the maximum cost for $\sigma = 1$.

Lemma B.2.1 *Let $\sigma = 1$. Then, there exist an optimal solution to the adversarial problem such that $d_t \in \{0, 2\}$ for $1 \leq t \leq |V|$.*

Proof. Notice that in each period starting inventory is at least 1, since $\sigma = 1$. If for $1 \leq t \leq |V|$, $d_t \leq 1$, then we can set d_t to 0 without changing the cost of the solution, since $h_t = 0$ for such periods. Therefore, we can assume w.l.o.g that $d_t > 1$ or $d_t = 0$ for $1 \leq t \leq |V|$. Suppose that for some period $1 \leq t \leq |V|$, $d_t > 1$. Notice that for every period t' that is a neighbor of t in G' the $d_{t'} < 1$, due to the constraint $(d_t + d_{t'})/2 \leq 1$. Therefore we can assume that $d_{t'} = 0$ for such periods. Since at the end of period t, we incur a backlogging cost of $b_t(d_t - 1)$, we can increase the cost by setting $d_t = 2$ which contradicts with the optimality of d. ■

Let κ be the clique number of graph G (the cardinality of the maximum stable set in G'). It follows from Lemma B.2.1 that when $\sigma = 1$ the value of the adversarial problem is equal to κ. Furthermore, it is not hard to show that if M is large enough, the optimal basestock is indeed $\sigma^* = 1$

Remark B.2.2 *Our proof shows that computing the optimal cost of the basestock problem for general polyhedral case is strongly NP-hard. On the other hand, one may consider approximating the optimal basestock without having to approximate the cost. We want to stress the fact that this is impossible and using the idea in our NP-completeness proof it can be shown that unless P = NP, we can not approximate the optimal basestock value within a constant factor.*

Appendix C

Algorithms for the discrete budgets model

C.1 Proof of Theorem 3.6.4

We remind the reader that the discrete budgets case 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

$$d_t \;=\; \mu_t + k_t\, \alpha_i,$$

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. (We assume that Γ_T is polynomial in T).

In this Section we will show that the basestock problem with time variant basestock levels under discrete budgets model can be solved in polynomial time for constant C. For notational simplicity we assume that $C = 1$ throughout this Section.

We model the problem as a longest path problem on a directed graph with

$$\sum_{t=1}^{T} \Gamma_t + T + 2$$

vertices and without any directed cycles. We highlight the fact that on such a graph the longest path problem can be solved in polynomial. (For background on the longest path problem see [ATO93] and [BH01]). We construct the graph as follows.

- We have a starting vertex, denoted by v_0 and a terminal vertex, denoted by v_{T+1}.
- For each time period $1 \leq t \leq T$ and each possible level of budget consumption, $0 \leq k \leq \Gamma_{t-1}$, we have a vertex denoted by $v_{t,k}$. (We assume that $\Gamma_0 = 0$).
- We have an edge from v_0 to every other vertex and from every other vertex to v_{T+1}.

- Moreover, from each vertex v_{t_1,k_1} such that $1 \leq t_1 \leq T$ and $0 \leq k_1 \leq \Gamma_{t_1-1}$, we have an edge into every vertex v_{t_2,k_2} such that $t_1 < t_2 \leq T$ and $k_1 \leq k_2 \leq \Gamma_{t_2-1}$.

We use the edges to locate the time periods where the on-hand inventory drops below the basestock level. The following is the interpretation of the edges that are defined above.

1. For any $1 \leq t \leq T$ and $0 \leq k \leq \Gamma_{t-1}$, the edge from v_0 to $v_{t,k}$ represents the case in which $x_j \geq \sigma_j$ for $0 \leq j < t$ and $x_t \leq \sigma_t$, and k units of budget is consumed in periods $1, 2, ..., t-1$. The cost for such an edge is given by

$$\begin{array}{rll}
\max & \sum_{i=1}^{t-1} W_i(x_1 - \sum_{j=1}^{i} d_j) + c_t(\sigma_t - (x_1 - \sum_{j=1}^{t-1} d_j)) & \\
s.t. & x_1 - \sum_{j=1}^{i-1} d_j \geq \sigma_i & 2 \leq i \leq t-1 \\
 & x_1 - \sum_{j=1}^{t-1} d_j \leq \sigma_t & \\
 & d_i = \mu_i \; + \; k_i \; \alpha & 1 \leq i \leq t-1 \\
 & \sum_{j=1}^{i} |k_j| \; \leq \; \Gamma_i & 1 \leq i \leq t-1 \\
 & l^i \leq k_i \leq u^i & 1 \leq i \leq t-1 \\
 & \sum_{i=1}^{t-1} |k_i| \leq k \;\; \text{and} \;\; k_i \;\; \text{integral} &
\end{array}$$

2. For any $1 \leq t \leq T$ and $0 \leq k \leq \Gamma_{t-1}$, the edge from $v_{t,k}$ to v_{T+1} represents the case in which $x_t < \sigma_t$ and $x_j \geq \sigma_j$ for $t+1 \leq j \leq T$. The cost is given by

$$\begin{array}{lll} \max & \sum_{i=t}^{T} W_i(\sigma_t - \sum_{j=t}^{i} d_j) & \\ s.t. & \sigma_t - \sum_{j=t}^{i-1} d_j \geq \sigma_i & t+1 \leq i \leq T \\ & d_i = \mu_i \; + \; k_i \; \alpha & \\ & \sum_{j=t}^{i} |k_j| \; \leq \; \Gamma_i - k & t \leq i \leq T \\ & l^i \leq k_i \leq u^i \;\; \text{and} \;\; k_i \;\; \text{integral.} & \end{array}$$

3. Any edge from v_{t_1,k_1} to v_{t_2,k_2} represents the case in which the inventory is below the basestock level at the beginning of periods t_1 and t_2, and above basestock level for periods $t_1 < t < t_2$. The cost for such edges is given by

$$\begin{array}{lll} \max & \sum_{i=t_1}^{t_2-1} W_i(\sigma_{t_1} - \sum_{j=t_1}^{i} d_j) + c_{t_2}(\sigma_{t_2} - (\sigma_{t_1} - \sum_{j=1}^{t_2-1} d_j)) & \\ s.t. & \sigma_{t_1} - \sum_{j=t_1}^{i-1} d_j \geq \sigma_i & t_1+1 \leq i \leq t_2-1 \\ & \sigma_{t_1} - \sum_{j=t_1}^{t_2-1} d_j \leq \sigma_{t_2} & \\ & d_i = \mu_i \; + \; k_i \; \alpha & \\ & \sum_{j=t_1}^{i} |k_j| \; \leq \; \Gamma_i - k_1 & t_1 \leq i \leq t_2-1 \\ & l^i \leq k_i \leq u^i & \\ & \sum_{j=t_1}^{t_2-1} |k_j| \leq k_2 - k_1 \;\; \text{and} \;\; k_i \;\; \text{integral.} & \end{array}$$

4. The edge (v_0, v_{T+1}) represents the the case in which inventory on hand is greater than the basestock level for the entire horizon. The cost for this is computed

similar to previous cases.

Each vertex of the form $v_{t,k}$ represents the case such that the inventory is below σ_t at the beginning of period t and the risk consumption up to period t is at most k. Notice that each path from v_0 to $v_{t,k}$ corresponds to a demand pattern that achieves this and the length of the path is equal to the total cost in periods $1, 2, \ldots, t-1$ plus the ordering cost in period t. Similarly any path from v_0 to v_{T+1} corresponds to a solution to the adversarial problem with the length of the path corresponding to the cost of the solution. Therefore the longest path in our graph gives the optimal solution to the adversarial problem.

To show the efficiency of our algorithm we have to show that we can compute the cost of each edge in polynomial time. We will show how to do so in the case of the edges for the third type above; the other cases will follow similarly.

We use the following forward dynamic program to compute the costs. For each period $t_1 \leq t \leq t_2$ we have a set, denoted by M_t, that consists of the triples (x, k, c). Here, the first two components of the triple give the inventory at the beginning of period t and the risk consumption until period t. The third component gives the maximum cost corresponding to x and k. We set $M_{t_1} = \{(\sigma_{t_1}, k_1, 0)\}$ and for each $t_1 < t \leq t_2$, we construct M_t as follows: For each triple $(x, k, c) \in M_{t-1}$ and $l_t \leq j \leq u_t$, we add $(x - (\mu_{t-1} + j\alpha), k + |j|, c + W_{t-1}(x - (\mu_{t-1} + j\alpha)))$ to M_t if $x - (\mu_{t-1} + j\alpha) \geq \sigma_t$ for $t_1 < t < t_2$, if $x - (\mu_{t-1} + j\alpha) \leq \sigma_t$ for $t = t_2$ and

$k + |j| \leq \Gamma_{t-1}$. If there exist two triples (x, k, c_1) and (x, k, c_2) such that $c_1 \leq c_2$, we delete (x, k, c_1). For each triple of the form $(x, k_2, c) \in M_{t_2}$, we compute the cost of the triple as $c + c_{t_2}(\sigma_{t_2} - x)$. The cost of the triple with the maximum cost is assigned to the edge.

Notice that the complexity of the algorithm depends on the sizes of the sets M_t for each t which in turn depends on the number of attainable values of the first and the second components of the triples. The second component is nonnegative and it is at most Γ_T which is assumed to be polynomial in T. Moreover, for each triple (x, k, c) in M_t, $x = \sigma_{t_1} - (\sum_{i=t_1}^{t} \mu_i + \rho\alpha)$ for some integer ρ. It is easy to see that ρ can only take values from the interval $[-\Gamma_t, \Gamma_t]$. Therefore size of M_t is $O(\Gamma_T^2)$.

C.2 Proof of Theorem 3.6.5

In this Section we develop a polynomial time algorithm to solve the adversarial problem with the intervals model. The intervals model has the same assumptions with the discrete budgets model except for the part (d) of the description given in previous section. Instead, we have the following model for the budget constraints.

(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 $i_k \leq t \leq j_k$.

In this proof we use the same approach as in the proof of Theorem 3.6.4. We construct a similar directed graph and solve a longest path problem on it. The vertex set of the graph is produced as follows:

- Similar to the previous Theorem, we have the starting and terminal vertices denoted by v_0 and v_{T+1}.
- Let $1 \leq t \leq T$ and $[i_{i_1}, j_{i_1}], \ldots, [i_{i_{n_t}}, j_{i_{n_t}}]$ be the intervals that are in the basis of $\mathcal{I}$ and that cover t. For each n_t+1-tuple $(t, k_1, k_2, \ldots, k_{n_t})$ such that $1 \leq t \leq T$ and $k_1 \leq k_2 \ldots \leq k_{n_t}$, we add a vertex to our graph.

Here the last n_t components of the (n_t+1)-tuple represents the the risk consumption between periods $i_{i_1}, \ldots, i_{i_{n_t}}$ and t.

The edges of the graph and their costs are obtained using a technique similar to that in Theorem 3.6.4. However, one has to be careful in calculating the costs of the edges, since the intervals from the bases that covers t may be different for different periods.

To calculate the costs on the edges we use dynamic programming and produce sets similar to the M_t used above for each period t. In this case the sets consist of (n_t+1)-tuples $(x, k_1, \ldots, k_{n_t}, c)$ in which the first component corresponds to the inventory at the beginning of period t, the next n_t components correspond to the risk consumptions in the intervals that includes t, and the last component is the maximum cost in periods $1, \ldots, t$ corresponding to x and $k_1, \ldots, k_{n_t}$. Since again the complexity of the DP depends on the size of the sets that we construct, it is polynomial for fixed w and C.

Bibliography

[ATO93] R. K. Ahuja, T. L. Magnanti and J. B. Orlin. Network Flows (1993), Prentice-Hall Inc, New Jersey.

[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.

[AHM51] K. Arrow, T. Harris and J. Marschak, Optimal inventory policy, *Econometrica* **19** (1951), 250 – 272.

[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).

[BRW84] I. Barany, T.J. Van Roy and L. A. Wolsey, Uncapacitated lot sizing: The convex hull of solutions, *Mathematical Programming Study* **22** (1984) 32-43.

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

[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.

[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.

[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.

[BT04] D. Bertsimas and A. Thiele, A robust optimization approach to supply chain management, *Lecture Notes in Computer Science* **3064** (2004), 86 – 100.

[BH01] A. Bjrklund and T. Husfeldt. Finding long paths in directed graphs is hard. Manuscript, 2001.

[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.

[DKW52] A. Dvoretzky, J. Kiefer and J. Wolfowitz, The inventory problem, *Econometrica* **20** (1952), 187 – 222.

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

[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.

[GW74] S.J. Gartska and R.J.B. Wets, On decisions rules in stochastic programming. Mathematical Programming **7** (1974) 117-143.

[GSc71] J. Glover and F. Schweppe. Control of linear dynamic systems with set constrained disturbances. IEEE Transactions on Automatic Control **16(6)** (1971) 766-767.

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

[H13] F. Harris, How many parts to make at once, *Factor, The Magazine of Management* **10** (1913), 135 – 136, 152.

[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).

[NW99] J. Nocedal and S. J. Wright. Numerical Optimization, (1999) Springer Series in Operations Research.

[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.

[WW58] H. M. Wagner and T. M. Whitin, Dynamic version of the economic lot size model, *Managemnet Science* **5** (1958) 89-96.

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