Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0000-0000 | EISSN 0000-0000 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000
© 0000 INFORMS

Multistage Robust Mixed Integer Optimization with Adaptive Partitions

Dimitris Bertsimas, Iain Dunning
Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139, dbertsim@mit.edu, idunning@mit.edu

We present a new partition-and-bound method for multistage adaptive mixed integer optimization (AMIO) problems that extends previous work on finite adaptability. The approach analyzes the optimal solution to a static (non-adaptive) version of an AMIO problem to gain insight into which regions of the uncertainty set are restricting the objective function value. We use this information to construct partitions in the uncertainty set, leading to a finitely adaptable formulation of the problem. We use the same information to determine a lower bound on the fully adaptive solution. The method repeats this process iteratively to further improve the objective until a desired gap is reached. We provide theoretical motivation for this method, and characterize both its convergence properties and the growth in the number of partitions. Using this insights we propose and evaluate enhancements to the method such as warm starts and smarter partition creation. We describe in detail how to apply finite adaptability to multistage AMIO problems to appropriately address nonanticipativity restrictions. Finally we demonstrate in computational experiments that the method can provide substantial improvements over a non-adaptive solution and existing methods for problems described in the literature. In particular, we find that our method produces high-quality solutions versus the amount of computational effort, even as the problem scales in both number of time stages and in the number of decision variables.

Key words: adaptive optimization, robust optimization, integer optimization
History: Submitted November 22, 2014. Revised August 10, 2015.

1. Introduction

Robust optimization is a methodology for addressing uncertainty in optimization problems where *uncertain parameters* are modeled as belonging to *uncertainty sets*, which contrasts with the more traditional approach of modeling uncertainty with probability distributions. We solve a robust optimization (RO) problem by optimizing with respect to the worst-case realization of the uncertain parameters over the uncertainty set Ξ,

$$\begin{aligned} \max_{\mathbf{x}\in\mathcal{X}} \ \min_{\boldsymbol{\xi}\in\Xi} \ & c(\boldsymbol{\xi},\mathbf{x}) \\ \text{subject to } & \mathbf{g}(\boldsymbol{\xi},\mathbf{x}) \leq \mathbf{0} \quad \forall \boldsymbol{\xi}\in\Xi \end{aligned} \tag{1}$$

where $\mathbf{x}$ is the vector of decision variables and $\boldsymbol{\xi}$ is a vector representing the uncertain parameters. For a survey of robust optimization we refer the reader to Bertsimas et al. (2011a) and Ben-Tal et al. (2009). The optimization problem (1) commonly has an infinite number of constraints but is tractable for many uncertainty sets of interest by reformulating the problem to obtain a deterministic optimization problem of finite size, or by generating constraints only as needed until the solution is optimal and feasible with respect to the uncertain constraints (e.g. Fischetti and Monaci (2012), Bertsimas et al. (2015)).

RO was introduced in the context of single-stage problems as a way to address uncertainty in problem data. Since then it has been shown to be a more general methodology to address multistage optimization problems, where the uncertain parameters are revealed over time. In most multistage problems our future *recourse* decisions can be made with knowledge of at least some of the uncertain parameters: those realized by the time those decisions must be made. If we place no restrictions on the functional relationship between uncertain parameters and recourse decisions, which is the ideal case, then the resulting problem is known to be hard in both the computational complexity sense (Ben-Tal et al. 2004), and has proved to be hard to solve in practice. Some work on addressing this fully adaptive case includes a Benders' decomposition-like approach to solve a unit commitment problem in Bertsimas et al. (2013), and the generalized "column-and-constraint generation" framework for two-stage continuous problems of Zeng and Zhao (2013). A goal of the adaptive optimization (AO) literature then is to provide methods to approximate and approach this fully adaptable ideal and to quantify, if possible, how good these approximations are.

Perhaps the most popular approach in the AO literature to date has been to restrict the recourse decisions to be simple functions of the uncertain parameters. Affine adaptability, also known as linear decision rules, was introduced to the RO setting in Ben-Tal et al. (2004) and has proven to be useful in a wide variety of settings including control problems (Goulart et al. 2006), supply chain problems (Ben-Tal et al. 2005), and project management (Chen et al. 2007). The choice of these linear decision rules is primarily one of computational practicality with no real expectation that they will be optimal in general (where optimality would be finding an affinely adaptive policy that performs as well as the fully adaptive policy). Many of the papers cited here demonstrate these shortcomings with both examples and more realistic experiments, and many researchers have characterized the conditions under which, and to what degree, affine adaptability performs well, e.g. Bertsimas et al. (2010), Bertsimas and Goyal (2012), Bertsimas and Bidkhori (2014). More complicated approaches trade computational efficiency for optimality, for example by using polynomial adaptability (Bertsimas et al. 2011b) and deflected linear decision rules (Chen et al. 2008). A major shortcoming of all these approaches is that they do not allow for discrete recourse decisions, significantly reducing their modeling power for adaptive mixed-integer optimization (AMIO) problems. Bertsimas and Caramanis (2007) acknowledge this and propose a decision rule for integer decisions as part of their sampling approach. Bertsimas and Georghiou (2015) propose a cutting-plane method that allows for piecewise linear decision rules for continuous recourse decisions and piecewise constant decision rules for integer recourse decisions. This approach was shown to be able to find good solutions to small problems, but does not scale well to problems with many time stages. A recent extension by Bertsimas and Georghiou (2014) for binary recourse decisions that allows for a reformulation demonstrates substantially improved performance on multistage problems.

An alternative approach is finite adaptability, where instead of focusing on the functional form of the recourse decisions we instead partition the uncertainty set and have different recourse decisions for each partition. This can be viewed as modeling the recourse decisions as piecewise constant functions of the uncertain parameters, with the domain of each piece (or equivalently the uncertainty set partitioning) either fixed, or decided endogenously as part of the optimization problem. One of the key benefits of this approach is that it handles discrete recourse decisions naturally, suggesting it is a good choice for AMIO. The quality of solutions obtained with a finite adaptability approach rests entirely on how the partitions are selected. For example, in Vayanos et al. (2011) the uncertainty set is partitioned ahead of time using hyper-rectangles. A bilinear optimization problem that decides the best two-partition recourse decision is proposed in Bertsimas and Caramanis (2010), and the gap between a fully adaptive solution and a finitely adaptive solution is bounded. Finally, Hanasusanto et al. (2014) propose a general methodology for obtaining optimal K-partition recourse decisions for two-stage binary optimization problems, and characterize when their approach is able to match the fully adaptive solution. They are able to solve a variety of problems, although even problems with as few as three partitions and twenty binary second stage variables cannot be solved to optimality within two hours. While this recent work represents great progress in AMIO, the size of AMIO problems that can be solved in a reasonable amount of time remains small and lags significantly behind the size of tractable continuous AO problems, particularly in the multistage case.

In this paper we will consider multistage AMIO problems of the general form

$$\begin{aligned} z_{full} = \min_{\mathbf{x}\in\mathcal{X}} \max_{\boldsymbol{\xi}\in\Xi} & \sum_{t=1}^{T} \mathbf{c}^t(\boldsymbol{\xi}) \cdot \mathbf{x}^t\left(\boldsymbol{\xi}^1,\ldots,\boldsymbol{\xi}^{t-1}\right) \\ \text{subject to} \quad & \sum_{t=1}^{T} \mathbf{A}^t(\boldsymbol{\xi}) \cdot \mathbf{x}^t\left(\boldsymbol{\xi}^1,\ldots,\boldsymbol{\xi}^{t-1}\right) \le \mathbf{b}(\boldsymbol{\xi}) \quad \forall \boldsymbol{\xi} = \left(\boldsymbol{\xi}^1,\ldots,\boldsymbol{\xi}^T\right) \in \Xi \end{aligned} \tag{2}$$

where $\boldsymbol{\xi}$ represents the uncertain parameters, $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$ define linear constraints and the objective respectively (all themselves affine functions of $\boldsymbol{\xi}$), and $\mathbf{x}^t$ is the vector of decision variables for stage t, which can be a mixture of discrete and continuous variables as captured by the deterministic set of constraints $\mathcal{X}$. For $t=1$ we note that $\mathbf{x}^1$ is a non-adaptive here-and-now decision that does not depend on $\boldsymbol{\xi}$.

The key contribution of this paper is a new "partition-and-bound" method for solving the multistage AMIO problem (2) that scales better to larger instances than alternative methods. In particular, as our method and the ones that we are comparing ourselves with do not necessarily find the fully adaptive solution, the metric by which we assess a method is the quality of the solutions it finds versus the computational effort required to obtain these solutions. The method builds on the finite adaptability approach to AMIO by utilizing the information obtained about the (possibly already partitioned) uncertainty set when solving an AO problem to select a new partitioning scheme. The same information is used to calculate a lower bound on the fully adaptive solution by adapting the sampling proposal of Hadjiyiannis et al. (2011). This process of solving and updating the partitioning can be repeated until a termination criterion in terms of the bound gap, or some other limit (total iterations or computation time) is reached. Our proposal can be extended in many ways: in this paper we show that affine adaptability for continuous recourse decisions can be trivially incorporated into our approach (leading to piecewise affine decision rules), but other extensions such as quadratic objectives and constraints can be also easily incorporated. We also give a detailed treatment to the interactions between finite adaptability and the need to enforce *nonanticipativity* for multistage problems - the natural notion that a decision at time t must be made without certain knowledge of uncertain parameters that will be revealed after time t.

Our approach lies somewhere between those that determine the optimal partitions simultaneously with the values of the decisions (Bertsimas and Caramanis 2010, Hanasusanto et al. 2014) and methods that select a partitioning *a priori* (Vayanos et al. 2011). While finalizing this paper we became of aware of a very recently submitted paper by Postek and Den Hertog (2014) that presents an iterative finite partitioning approach similar to the idea we present in this paper. Throughout this paper we will contrast and compare our approach to this alternative, both from a theoretical and computational point of view.

After the initial submission of this paper Song and Luedtke (2014) proposed an "adaptive partition-based approach" for two-stage stochastic programs (SP) that partitions sets of scenarios. Their approach uses the solutions to these relaxed problems to change the partitioning scheme – a scenario-based analogue to the approach of this paper. An interesting point of difference is the perspective of the two approaches: in the SP case we start with many individual scenarios that we group into partitions, whereas in the RO approach we start with a single set and seek to split apart into partitions. Song and Luedtke (2014) build upon other similar aggregation techniques discussed in the SP literature, especially work aimed to at improving the tractability of sample-average approximation. We refer interested readers to their work for a more comprehensive treatment of the related SP literature.

We have structured the paper as follows:

- In Section 2, we present a theoretical result describing a property that a new partitioning of an uncertainty set must have to improve a solution of an AMIO over an existing partitioning scheme. We use this result to propose a method of constructing partitions using the solution of AMIO problems that is inspired by Voronoi diagrams. In this section, we initially focus on the two-stage case: By combining this with a lower bounding method, we obtain an iterative partition-and-bound method for two-stage AMIO. We explain how affine adaptability can be incorporated into the method, and provide a worked example to aid in implementation.

- In Section 3, we analyze our method from two points of view. We show that our method has a desirable non-increasing property that enables computational "warm starts", but provide an example that demonstrates that there is no general guarantee of convergence for the solutions produced by our method to the fully adaptive solution. We also consider the question of the growth in the number of partitions with increasing partitions. We show that while the number of partitions can grow very large after many iterations, the quality of the solution those partitions lead to for a fixed computational budget is the more relevant quantity. We also present a small modification to the method that can greatly reduce the number of partitions necessary.

- In Section 4, we generalize the method to the multistage case. We first demonstrate the issues that must be considered in choosing the partitioning scheme while satisfying nonanticipativity. We then describe the general multistage partitioning scheme and a simple routine to determine the minimum set of nonanticipativity constraints to add.

- In Section 5, we provide results of computational experiments that show that our method produces good solutions versus total computation time. In particular, we solve instances of a capital budgeting problem and a multistage lot sizing problem, including a comparison with the methods of Hanasusanto et al. (2014), Bertsimas and Georghiou (2015), and Postek and Den Hertog (2014).

- In the final Section 6, we provide some concluding remarks and thoughts on future directions.

Notation

We use lowercase, non-bold face symbols for scalar quantities (e.g., $z \in \mathbb{R}$) and a bold face for vectors (e.g., $\mathbf{x} \in \mathbb{R}^n$). Matrices are denoted by bold face uppercase symbols, e.g., $\mathbf{A} \in \mathbb{R}^{m \times n}$. We use Ξ to denote an uncertainty set and $\boldsymbol{\xi}$ to denote uncertain parameters. Our uncertain parameters are typically a vector-of-vectors, i.e.,

$$\boldsymbol{\xi} = \left(\boldsymbol{\xi}^1, \ldots, \boldsymbol{\xi}^t, \ldots, \boldsymbol{\xi}^T\right), \tag{3}$$

where $\boldsymbol{\xi}^t$ is the vector of uncertain parameters for stage t. The j^{th} component of the uncertain parameters for time stage t is expressed with a subscript, e.g., ξ^t_j. For particular realizations of uncertain parameters we use a hat: For example $\hat{\boldsymbol{\xi}}_i$ is the i^{th} realization and $\hat{\boldsymbol{\xi}}^t_i$ is the vector of uncertain parameters for sample i and time stage t.

2. Partition-and-bound method for two-stage AMIO

In this section, we describe a partition-and-bound method that takes a finite adaptability approach to two-stage AMIO problems. In particular, we consider the problem

$$\begin{aligned} \min_{\mathbf{x} \in \mathcal{X}, z} \quad & z & & (4) \\ \text{subject to} \quad & \mathbf{c}^1(\boldsymbol{\xi}) \cdot \mathbf{x}^1 + \mathbf{c}^2(\boldsymbol{\xi}) \cdot \mathbf{x}^2(\boldsymbol{\xi}) \le z & \forall \boldsymbol{\xi} \in \Xi & \\ & \mathbf{a}^1_i(\boldsymbol{\xi}) \cdot \mathbf{x}^1 + \mathbf{a}^2_i(\boldsymbol{\xi}) \cdot \mathbf{x}^2(\boldsymbol{\xi}) \le b_i(\boldsymbol{\xi}) & \forall \boldsymbol{\xi} \in \Xi,\ i \in \mathcal{I}, & \end{aligned}$$

which has m linear uncertain constraints ($\mathcal{I} = \{1, \ldots, m\}$) and the objective expressed in epigraph form. This is a special case of the multistage problem (2) where $T = 2$. We first demonstrate some properties of AO solutions that motivate the design of our proposed method, then discuss the implementation details.

2.1. Partitioning scheme

Finite adaptability is a simple approach for approximating fully adaptive wait-and-see decisions that can vary with the uncertain parameters. In the context of our two-stage problem (4) we can view finite adaptability as a restriction of the decisions $\mathbf{x}^2(\boldsymbol{\xi})$ to the class of piecewise constant policies, i.e.

$$\mathbf{x}^2(\boldsymbol{\xi}) = \begin{cases} \mathbf{x}^2_1, & \forall \boldsymbol{\xi} \in \Xi_1, \\ \vdots & \\ \mathbf{x}^2_K, & \forall \boldsymbol{\xi} \in \Xi_K, \end{cases} \tag{5}$$

where $\Xi_k, k \in \{1, \ldots, K\}$ defines a partitioning of Ξ and $\mathbf{x}^2_k$ is the implemented decision if the realized value of $\boldsymbol{\xi}$ is in the partition Ξ_k. If the realized value lies on the boundary of two partitions then we may select either decision, as the decision associated with each partition is feasible for all values of the uncertain parameters on the boundary. We will converge in the limit to the fully adaptive solution as the number of partitions grows and the diameter of the largest partition approaches 0. Therefore, there is a natural trade-off between the number of partitions (and the solution time) and how "close" our solution is to the fully adaptive solution, and a goal when using finite adaptability is to identify a partitioning scheme that is near the efficient frontier of this trade-off.

To motivate our selection of partitioning scheme we will consider a variant of (4) where the second-stage decisions cannot change with $\boldsymbol{\xi}$, i.e. a static policy:

$$\begin{aligned} z_{static} = \min_{\mathbf{x} \in \mathcal{X}, z} \quad & z \\ \text{subject to} \quad & \mathbf{c}^1 \cdot \mathbf{x}^1 + \mathbf{c}^2 \cdot \mathbf{x}^2 \leq z \\ & \mathbf{a}^1_i(\boldsymbol{\xi}) \cdot \mathbf{x}^1 + \mathbf{a}^2_i(\boldsymbol{\xi}) \cdot \mathbf{x}^2 \leq b_i(\boldsymbol{\xi}) \quad \forall \boldsymbol{\xi} \in \Xi, \ i \in \mathcal{I}. \end{aligned} \tag{6}$$

We will initially restrict ourselves to the case where all decision variables are continuous, the uncertainty set is polyhedral (which ensures that a cutting plane method terminates), the objective coefficients are certain, and the problem is a feasible and bounded. Consider using a cutting-plane method to solve this static problem: that is, we iteratively solve a deterministic relaxation of the RO problem with a finite subset of the possible constraints and add violated constraints from the full RO problem until the solution of the relaxation is feasible with respect to all uncertain parameters in the uncertainty set. For each constraint i we define $\mathcal{A}_i$ to be set of *active uncertain parameters* $\hat{\boldsymbol{\xi}}$ corresponding to the generated constraints (cuts) that have zero slack. These sets may be empty, singletons, or even have multiple elements (multiple cuts for i with zero slack). After solving for the static policy we consider partitioning the uncertainty set into two sets Ξ_1 and Ξ_2, that is

$$
\begin{aligned}
z_{part} = \min_{\mathbf{x}\in\mathcal{X},z} \quad & z && (7)\\
\text{subject to} \quad & \mathbf{c}^1\cdot\mathbf{x}^1 + \mathbf{c}^2\cdot\mathbf{x}^2_j \le z \qquad \forall j\in\{1,2\}\\
& \mathbf{a}^1_i(\boldsymbol{\xi})\cdot\mathbf{x}^1 + \mathbf{a}^2_i(\boldsymbol{\xi})\cdot\mathbf{x}^2_j \le b_i(\boldsymbol{\xi}) \quad \forall\boldsymbol{\xi}\in\Xi_j, j\in\{1,2\}, i\in\mathcal{I},
\end{aligned}
$$

which we hope will have a solution that is better than the static policy (that is, $z_{part} < z_{static}$). Let $\mathcal{A} = \bigcup_i \mathcal{A}_i$. We now seek to show that whether this improvement occurs or not depends on the nature of the partitions Ξ_1 and Ξ_2.

THEOREM 1. *If $\mathcal{A}$, the set of all active uncertain parameters for the static problem* (6)*, satisfies either $\mathcal{A}\subset\Xi_1$ or $\mathcal{A}\subset\Xi_2$ then $z_{part} = z_{static}$. Otherwise $z_{part}\le z_{static}$.*

Proof of Theorem 1 Suppose that the set of active uncertain parameters $\mathcal{A}$ is a subset of Ξ_1. Consider solving the problem

$$
\begin{aligned}
z_{half} = \min_{\mathbf{x}\in\mathcal{X},z} \quad & z && (8)\\
\text{subject to} \quad & \mathbf{c}^1\cdot\mathbf{x}^1 + \mathbf{c}^2\cdot\mathbf{x}^2_1 \le z\\
& \mathbf{a}^1_i(\boldsymbol{\xi})\cdot\mathbf{x}^1 + \mathbf{a}^2_i(\boldsymbol{\xi})\cdot\mathbf{x}^2_1 \le b_i(\boldsymbol{\xi}) \quad \forall\boldsymbol{\xi}\in\Xi_1, i\in\mathcal{I},
\end{aligned}
$$

which can be viewed as relaxation of (6) as we must only satisfy the constraints for a subset of Ξ. Despite being a relaxation, we still have $z_{half} = z_{static}$, as every cut with minimal slack in (6) will still be a valid cut for (8) and thus the solution will be equal as we have not relaxed any of the constraints that bound the previous solution. Note that this does not preclude the possibility that the equivalent problem to (8) for the other partition Ξ_2 has an objective function value better than z_{static}, but as z_{part} is taken to be the maximum of the objective function values corresponding to the two partitions in (7) there can be no improvement of objective function value overall (i.e., $z_{half} = z_{part} = z_{static}$). On the other hand, if we have partitions such that $\mathcal{A} \not\subset \Xi_1$ and $\mathcal{A} \not\subset \Xi_2$, then the possibility of improvement exists: at least one zero-slack cut will not be valid for z_{half}, so the solution to (8) is not restricted to having the same solution as (6). $\square$

This result naturally leads us to consider partitioning schemes that will achieve this required condition, and thus enable improvement. A partitioning scheme that achieves this is one in which every $\hat{\boldsymbol{\xi}} \in \mathcal{A}$ lies in its own partition. A simple way (both conceptually and computationally) to construct such a set of partitions is to use a *Voronoi diagram* (see e.g. Aurenhammer (1991) for an overview). Given a set of K points $\left\{\hat{\boldsymbol{\xi}}_1, \ldots, \hat{\boldsymbol{\xi}}_K\right\} \subset \Xi$ the Voronoi diagram associated with these points defines a partition for each $\hat{\boldsymbol{\xi}}_i$ that contains only the $\boldsymbol{\xi} \in \Xi$ such that the Euclidean distance between $\hat{\boldsymbol{\xi}}_i$ and $\boldsymbol{\xi}$ is less than or equal to the distance to any other given point $\hat{\boldsymbol{\xi}}_j$. We now apply this to the problem at hand: as $\mathcal{A}$ is a finite set we can express the partition induced by active uncertain parameter $\hat{\boldsymbol{\xi}}_i \in \mathcal{A}$ as

$$\begin{aligned}\Xi\left(\hat{\boldsymbol{\xi}}_i\right) &= \Xi \cap \left\{\boldsymbol{\xi} \,\middle|\, \left\|\hat{\boldsymbol{\xi}}_i - \boldsymbol{\xi}\right\|_2 \leq \left\|\hat{\boldsymbol{\xi}}_j - \boldsymbol{\xi}\right\|_2 \quad \forall \hat{\boldsymbol{\xi}}_j \in \mathcal{A},\ \hat{\boldsymbol{\xi}}_i \neq \hat{\boldsymbol{\xi}}_j\right\} \\ &= \Xi \cap \left\{\boldsymbol{\xi} \,\middle|\, \left(\hat{\boldsymbol{\xi}}_j - \hat{\boldsymbol{\xi}}_i\right) \cdot \boldsymbol{\xi} \leq \frac{1}{2}\left(\hat{\boldsymbol{\xi}}_j - \hat{\boldsymbol{\xi}}_i\right) \cdot \left(\hat{\boldsymbol{\xi}}_j + \hat{\boldsymbol{\xi}}_i\right) \quad \forall \hat{\boldsymbol{\xi}}_j \in \mathcal{A},\ \hat{\boldsymbol{\xi}}_i \neq \hat{\boldsymbol{\xi}}_j\right\}\end{aligned}$$

which is a set defined by Ξ and $|\mathcal{A}| - 1$ additional linear constraints (and if Ξ is polyhedral then $\Xi\left(\hat{\boldsymbol{\xi}}_i\right)$ will also be polyhedral).

We will now generalize away from the cutting plane method-based reasoning and continuous problems, and we will assume throughout that the uncertainty set is closed and convex but not necessarily polyhedral. For constraint i and a static policy solution $(\bar{\mathbf{x}}^1, \bar{\mathbf{x}}^2)$ we define the set of active uncertain parameters for constraint i as

$$\mathcal{A}_i = \arg\min_{\boldsymbol{\xi} \in \Xi} \left\{b_i(\boldsymbol{\xi}) - \mathbf{a}_i^1(\boldsymbol{\xi}) \cdot \bar{\mathbf{x}}^1 - \mathbf{a}_i^2(\boldsymbol{\xi}) \cdot \bar{\mathbf{x}}^2\right\}, \tag{9}$$

which, as the set of optimal solutions for minimizing a linear function over a convex set is itself convex, may be a convex set of infinite cardinality. Note that we do not require that these $\hat{\boldsymbol{\xi}} \in \mathcal{A}_i$

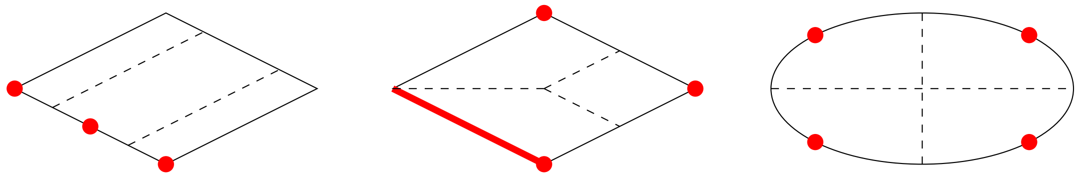


Figure 1 Partitioning of simple uncertainty sets using Voronoi diagrams. The first two diagrams use the polyhedral uncertainty set $\Xi_P = \left\{ \boldsymbol{\xi} \,\middle|\, \left\| \left[\frac{1}{2}\xi_1, \xi_2 \right] \right\|_1 \leq 1 \right\}$, and the third uses an ellipsoidal set $\Xi_E = \left\{ \boldsymbol{\xi} \,\middle|\, \left\| \left[\frac{1}{2}\xi_1, \xi_2 \right] \right\|_2 \leq 1 \right\}$. The bold points represent the active uncertain parameters $\hat{\xi}$ used to create the partitions. In the center example the red line represents an infinite set of possible active uncertain parameters, from which we have selected one arbitrarily.

correspond to constraints that have zero slack, as this may possibly never occur for a problem with integer variables. We can also define an equivalent set $\mathcal{A}_c$ for the objective epigraph constraint. The possibility that these sets are of infinite cardinality presents a problem for our proposed Voronoi diagram scheme, as the natural generalization of Voronoi diagrams for convex sets (instead of just points) can produces nonconvex partitions (Lee and Drysdale 1981). We propose selecting a single point from each set $\mathcal{A}_i$ using either problem-specific knowledge, or a more general technique such as taking the Chebyshev center or random selection. In Figure 1 we demonstrate the partitions that would be induced in a polyhedral uncertainty set for two different sets of uncertain parameters, as well as a partitioning of an ellipsoidal set.

2.2. Description of method

We can now build an iterative method built around this partitioning scheme. The method starts by solving a static-policy version of our adaptive optimization problem to determine a set of active uncertain parameters. We use these to construct a finitely-adaptive version of our problem, and solve that. This in turn produces a new set of active uncertain parameters which we can then use to partition further, ideally improving on the previous solution at each iteration.

We have already described how we collect a finite set of active uncertain parameters given a solution. The other key detail is the way we track the active uncertain parameters across iterations, and how we construct the partitions after the first iteration. We propose nested partitions: the active uncertain parameters $\mathcal{A}$ corresponding to the constraints for a partition at iteration k are used to sub-partition that partition at iteration $k+1$. We can describe the partition construction scheme in terms of a *tree* $\mathcal{T}$ of uncertain parameters, for which we will define the following sets: $\mathit{Leaves}\left(\mathcal{T}\right)$ is the set of leaves of the tree $\mathcal{T}$, $\mathit{Children}\left(\hat{\boldsymbol{\xi}}\right)$ is the set of children of $\hat{\boldsymbol{\xi}}$ in the tree $\mathcal{T}$, $\mathit{Parent}\left(\hat{\boldsymbol{\xi}}\right)$ is the parent of $\hat{\boldsymbol{\xi}}$ in the tree $\mathcal{T}$, and finally $\mathit{Siblings}\left(\hat{\boldsymbol{\xi}}\right)$ is the set of children of the parent $\hat{\boldsymbol{\xi}}$, i.e. $\mathit{Siblings}\left(\hat{\boldsymbol{\xi}}\right)=\mathit{Children}\left(\mathit{Parent}\left(\hat{\boldsymbol{\xi}}\right)\right)$. Each leaf of $\mathcal{T}$ corresponds to a partition of the uncertainty set and each level of the tree corresponds to an iteration of the algorithm. At each iteration we construct a partition for a leaf $\hat{\boldsymbol{\xi}}_i$ as an intersection of partitions

$$\begin{aligned}\Xi\left(\hat{\boldsymbol{\xi}}_i\right)=&\left\{\boldsymbol{\xi}\,\middle|\,\left\|\hat{\boldsymbol{\xi}}_i-\boldsymbol{\xi}\right\|_2\le\left\|\hat{\boldsymbol{\xi}}_j-\boldsymbol{\xi}\right\|_2\quad\forall\hat{\boldsymbol{\xi}}_j\in\mathit{Siblings}\left(\hat{\boldsymbol{\xi}}_i\right)\right\}\\&\cap\left\{\boldsymbol{\xi}\,\middle|\,\left\|\mathit{Parent}\left(\hat{\boldsymbol{\xi}}_i\right)-\boldsymbol{\xi}\right\|_2\le\left\|\hat{\boldsymbol{\xi}}_j-\boldsymbol{\xi}\right\|_2\quad\forall\hat{\boldsymbol{\xi}}_j\in\mathit{Siblings}\left(\mathit{Parent}\left(\hat{\boldsymbol{\xi}}_i\right)\right)\right\}\\&\vdots\\&\cap\Xi,\end{aligned}\tag{10}$$

which terminates when the parent is the root node, which has no siblings and thus is equivalent to the entire uncertainty set. By adding the active uncertain parameters for partition $\Xi\left(\hat{\boldsymbol{\xi}}_i\right)$ as children of $\hat{\boldsymbol{\xi}}_i$ in $\mathcal{T}$, we create subpartitions of $\Xi\left(\hat{\boldsymbol{\xi}}_i\right)$ at the next iteration as shown in Figure 2.

Given this machinery our method for solving problems of the form of (4) proceeds as follows:

Partition-and-bound method for two-stage AMIO

1. **Initialize**. Define $\mathcal{T}^1$ to be the initial tree of uncertain parameters that consists of one root node (any $\boldsymbol{\xi}\in\Xi$). Set iteration counter $k\leftarrow 1$

2. **Solve**. Solve the following partitioned version of (4), with one partition for every $\hat{\boldsymbol{\xi}}_j\in\mathit{Leaves}\left(\mathcal{T}^k\right)$:

$$\begin{aligned}z_{alg}\left(\mathcal{T}^k\right)=\min_{\mathbf{x}\in\mathcal{X},z}\quad & z\\ \text{subject to}\quad & \mathbf{c}^1\left(\boldsymbol{\xi}\right)\cdot\mathbf{x}^1+\mathbf{c}^2\left(\boldsymbol{\xi}\right)\cdot\mathbf{x}^2_j\le z && \forall\boldsymbol{\xi}\in\Xi\left(\hat{\boldsymbol{\xi}}_j\right),\forall\hat{\boldsymbol{\xi}}_j\in\mathit{Leaves}\left(\mathcal{T}^k\right)\\ & \mathbf{a}^1_i\left(\boldsymbol{\xi}\right)\cdot\mathbf{x}^1+\mathbf{a}^2_i\left(\boldsymbol{\xi}\right)\cdot\mathbf{x}^2_j\le b_i\left(\boldsymbol{\xi}\right) && \forall\boldsymbol{\xi}\in\Xi\left(\hat{\boldsymbol{\xi}}_j\right),\forall\hat{\boldsymbol{\xi}}_j\in\mathit{Leaves}\left(\mathcal{T}^k\right),i\in\mathcal{I},\end{aligned}\tag{11}$$

where $\Xi\left(\hat{\boldsymbol{\xi}}_j\right)$ is the set defined in (10), and $\mathbf{x}^2_j$ is the set of recourse decisions corresponding to the partition induced by parameters $\hat{\boldsymbol{\xi}}_j$.

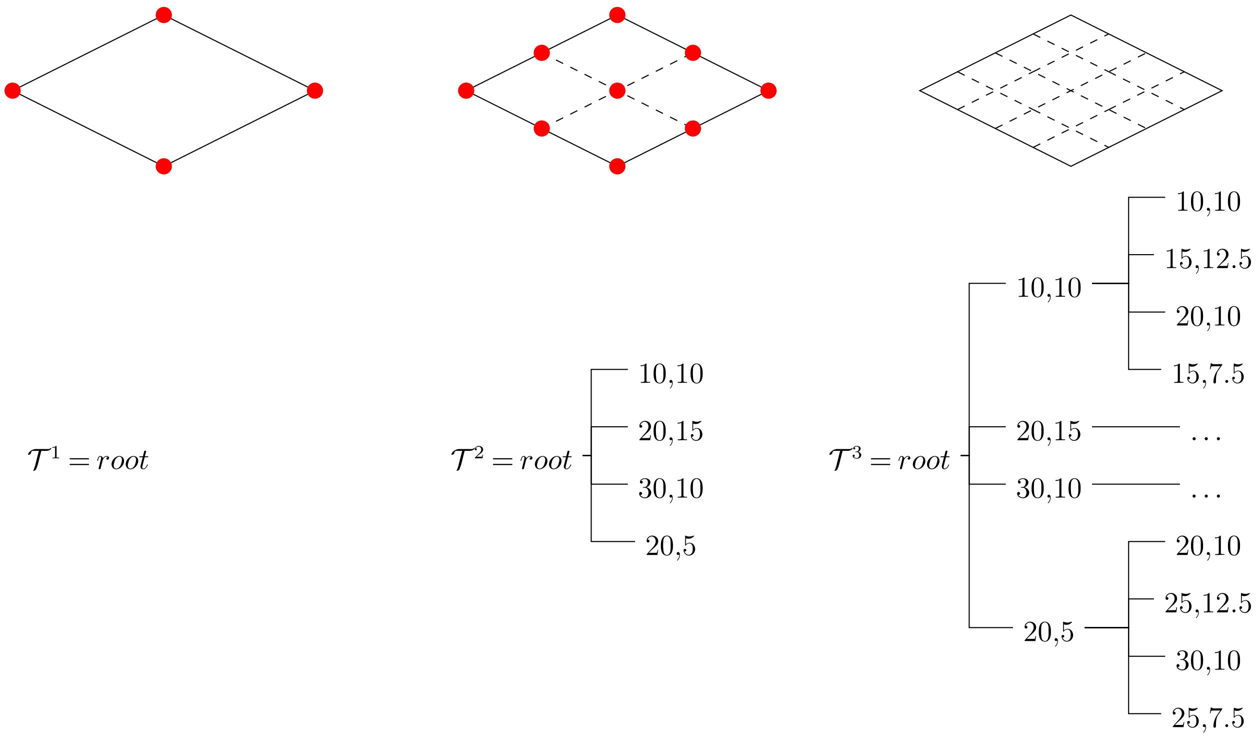


Figure 2 Visualization of the partitions and active uncertain parameter tree across multiple iterations. The uncertainty set is $\Xi = \left\{ (\xi_1, \xi_2) \,\middle|\, \frac{|\xi_1 - 20|}{10} + \frac{|\xi_2 - 10|}{5} \leq 1 \right\}$, and we assume that at each iteration all the extreme points of the set/partitions are active uncertain parameters (red points) in the solved partitioned problem.

3. **Grow**. Initialize $\mathcal{T}^{k+1} \leftarrow \mathcal{T}^k$. For every leaf $\hat{\boldsymbol{\xi}}_j \in Leaves\left(\mathcal{T}^{k+1}\right)$, add children to that leaf for each $\hat{\boldsymbol{\xi}}$ in the set of active uncertain parameters $\mathcal{A}$ for the partition $\Xi\left(\hat{\boldsymbol{\xi}}_j\right)$ and the solution to (11), selecting a finite subset if there is an infinite set of active uncertain parameters to choose from.

4. **Bound**. Calculate a lower bound $z_{lower}\left(\mathcal{T}^{k+1}\right)$ (see Section 2.3) of the fully adaptive solution, and terminate if the bound gap

$$\frac{(z_{alg} - z_{lower})}{|z_{lower}|} \tag{12}$$

is less than ϵ_{gap}, or the iteration or time limit has been reached. Otherwise set $k \leftarrow k+1$ and go to Step 2.

A possibility not directly addressed by this method is the case where the initial static solution is infeasible. For example, consider the feasibility problem

$$\begin{aligned} z_{feas} = \min_{x^2} \quad & 0 && (13)\\ \text{subject to} \quad & x^2(\xi) \le \xi + \epsilon \quad \forall \xi \in [0,1] \\ & x^2(\xi) \ge \xi - \epsilon \quad \forall \xi \in [0,1], \end{aligned}$$

which has no feasible static policy (but does have the solution $x^2(\xi) = \xi$, as well as piecewise constant solutions). If infeasibility does occur, then one potential remedy is to use problem-specific knowledge to identify a near-feasible solution or to initially relax problematic constraints so that active uncertain parameters can be collected for partitioning. Failing that, sampling (random or otherwise) of the uncertainty set can serve the same purpose. Once a feasible solution is identified the algorithm can proceed as before.

Our method is similar to that of Postek and Den Hertog (2014) for the case of two-stage problems. The primary difference is the method of constructing partitions: while our method creates multiple child partitions at each iteration, the method of Postek and Den Hertog (2014) creates only two child partitions per iteration per partition. These two child partitions are defined by a single separating hyperplane between a pair of active uncertain parameters, with the primary selection heuristic employed in that paper being to choose the pair with the greatest distance between them. We will explore the implications of this difference further in the Section 3 and in the computational experiments.

We also note that, while we have focused here on mixed-integer linear optimization, other problem classes can be solved using the same approach. For example, we could consider quadratic terms in the objective, or second-order cone constraints. The key primitive we require is the ability to extract active uncertain parameters after solving a partitioned problem in order to further improve the solution at the next iteration, and the ability to be partition the uncertainty set.

2.3. Calculating lower bounds

Our method ideally produces successfully closer approximations to the fully adaptive problem (4). The most natural termination criterion then, apart from an iteration- or time-limit criterion, is to terminate when our approximation (an upper bound, as we are minimizing) is sufficiently close to the fully adaptive solution. As we do not know the fully adaptive solution, we propose using the set of uncertain parameters available at the end of iteration k to obtain a lower bound. That is, after the solution of the partitioned problem at iteration k, we grow our tree $\mathcal{T}$ and use all parameters in $\mathcal{T}$ to construct and solve a deterministic problem whose objective provides the lower bound. This is essentially the same as the "scenario based bound" of Hadjiyiannis et al. (2011), which also utilizes "binding" uncertainty realizations, much like our active uncertain parameters. We restate this bound here in the context of our method.

Proposition 1. *Consider the solution of the deterministic optimization problem*

$$\begin{aligned} z_{lower}\left(\mathcal{T}\right) = \min_{\mathbf{x}\in\mathcal{X},z} \quad & z && (14)\\ \textit{subject to} \quad & \mathbf{c}^1\left(\hat{\boldsymbol{\xi}}_j\right)\cdot\mathbf{x}^1 + \mathbf{c}^2\left(\hat{\boldsymbol{\xi}}_j\right)\cdot\mathbf{x}^2_j \le z && \forall \hat{\boldsymbol{\xi}}_j \in \mathcal{T}\\ & \mathbf{a}^1_i\left(\hat{\boldsymbol{\xi}}_j\right)\cdot\mathbf{x}^1 + \mathbf{a}^2_i\left(\hat{\boldsymbol{\xi}}_j\right)\cdot\mathbf{x}^2_j \le b_i\left(\hat{\boldsymbol{\xi}}_j\right) && \forall \hat{\boldsymbol{\xi}}_j \in \mathcal{T}, i \in I \end{aligned}$$

where $\mathcal{T}$ is a set of uncertain parameters $\hat{\boldsymbol{\xi}} \in \Xi$. Then $z_{lower}\left(\mathcal{T}\right) \le z_{full}$, where z_{full} is the solution to (4).

Proof of Proposition 1 Problem (14) is a relaxation of (4) as we only require the solution to satisfy the constraints for a finite subset of uncertain parameters. Any solution to the fully adaptive problem (4) has a trivial mapping to a feasible solution for (14) that preserves the objective function value, so $z_{lower}\left(\mathcal{T}\right) \le z_{full}$. □

A useful property of this bound is that as the tree grows the lower bound will tend to approach the fully adaptive solution. At the same time we are improving our upper bound by decreasing the size of each partition, and thus closing the bound gap from both sides. Care must be taken if solving (14) approximately: for example if (14) is a MIO problem and we allow the solver to terminate when the gap between the best known integer solution and the continuous relaxation falls below a tolerance, as is common in practice, we will obtain a solution that is larger than the true lower bound or even the fully adaptive solution and thus understate the bound gap.

2.4. Worked example: two-stage inventory control problem

Example 1. We now demonstrate our approach by solving a two-stage inventory control problem. We must satisfy all of the uncertain demand, and there are three ways to order stock to do so: ordering any amount here-and-now at a unit cost of 50, ordering a fixed "lot" of size 25 at a unit cost of 60, and ordering a fixed lot of size 25 at a unit cost of 75. Only one lot of each size may be ordered, but the decision to order them can be made after the demand is known. We must pay a high unit cost of 65 for the storage and disposal of any of the remaining stock after the demand is satisfied. We can formulate this as a two-stage AMIO problem

$$\begin{aligned} \min \quad & z && (15) \\ \text{subject to} \quad & 50x^1 + 65I^2(\xi) + 1500y^2_A(\xi) + 1875y^2_B(\xi) \le z && \forall \xi \in \Xi \\ & I^2(\xi) \ge 0 && \forall \xi \in \Xi \\ & x^1 \ge 0,\ y^2_A(\xi), y^2_B(\xi) \in \{0,1\} && \forall \xi \in \Xi, \end{aligned}$$

where x^1 is the here-and-now continuous ordering decision, $y^2_A(\xi)$ and $y^2_B(\xi)$ are the wait-and-see binary ordering decisions, $I^2(\xi) = x^1 - \xi + 25y^2_A(\xi) + 25y^2_B(\xi)$, and ξ is the uncertain demand that is drawn from the uncertainty set $\Xi = \{\xi \,|\, 5 \le \xi \le 95\}$. We will initialize our tree $\mathcal{T}^1$ (step 1) with a root $\hat{\xi} = 50$, leading to a problem with a single partition (the entire set) and with y^2_A and y^2_B unable to vary with ξ (static). Solving this single-partition version of (15) (step 2) gives us the initial solution $z = 10600$, $x^1 = 95$, $y^2_A = 0$, and $y^2_B = 0$. The active uncertain parameter value for the objective constraint is when the demand is low ($\hat{\xi} = 5$, which leaves 90 units of stock) and the worst-case for the non-negative inventory constraint is when the demand is high ($\hat{\xi} = 95$, leaving no stock). We grow the tree (step 3) to obtain $\mathcal{T} = \{50 : \{5, 95\}\}$, which we then use to determine a lower bound on the fully adaptive solution ($z_{lower} = 5625$) for an initial gap of approximately 88%.

We now solve the two-partition version of (15) at iteration 2. The first partition is

$$\begin{aligned}\Xi\left(\hat{\xi}_1 = 5\right) &= \{\xi \,|5 \le \xi \le 95\} \cap \{\xi \,|\|5-\xi\|_2 \le \|95-\xi\|_2\} \\ &= \{\xi \,|5 \le \xi \le 50\}\end{aligned}$$

and the second is $\Xi\left(\hat{\xi}_2 = 95\right) = \{\xi \,|50 \le \xi \le 95\}$. The objective function value for this problem is substantially lower ($z = 7925$) than at iteration 1 and the stock bought initially is lower ($x^1 = 70$). For the first partition we will order neither lot ($y^2_{A,1} = y^2_{B,1} = 0$), but if the demand falls in the second partition we will purchase the first, cheaper lot ($y^2_{A,2} = 1$, $y^2_{B,2} = 0$). Once again the active uncertain parameters lie at the extremes of the partitions, which leads to the tree $\mathcal{T}^3 = \{50 : \{5 : \{5, 50\}, 95 : \{50, 95\}\}\}$. Our lower bound is unchanged, as we were already using the values 5, 50, and 95, but our gap shrinks substantially to 41%, thanks to the reduced upper bound.

We now begin our third and final iteration, by solving the partitioned problem induced by $\mathcal{T}^3$. In the interest of space we only detail the construction of the partition for the leaf $\hat{\xi} = 50$ that is the child of $\hat{\xi} = 5$:

$$\begin{aligned}\Xi\left(\hat{\xi} = 50\right) &= \Xi \cap \{\xi \,|\|5-\xi\|_2 \le \|95-\xi\|_2\} \cap \{\xi \,|\|50-\xi\|_2 \le \|5-\xi\|_2\} \\ &= \{\xi \,|\xi \le 50\} \cap \left\{\xi \,\middle|\, \xi \ge 27\frac{1}{2}\right\} \\ &= \{\xi \,|5 \le \xi \le 27.5\}.\end{aligned}$$

Solving this four-partition problem produces the best solution yet ($z = 7375$). The solution purchases substantially less stock up front ($x^1 = 47\frac{1}{2}$), and we can best describe the values of y_A and y_B as piecewise constant functions of ξ, i.e.,

$$y^2_A(\xi) = \begin{cases} 0, & 5 \le \xi < 72\frac{1}{2}, \\ 1, & 72\frac{1}{2} \le \xi \le 95, \end{cases} \tag{16}$$

and

$$y^2_B(\xi) = \begin{cases} 0, & 5 \le \xi < 27\frac{1}{2}, \\ 1, & 27\frac{1}{2} \le \xi \le 95, \end{cases} \tag{17}$$

respectively. Growing the tree with the new active uncertain parameters leads to a new lower bound of $z_{lower} = 5912\frac{1}{2}$, resulting in a final gap of 26%. Although it cannot be determined at this iteration, our upper bound is very good as the fully adaptive solution has an objective of $z_{full} = 7250$ (as determined by exhaustively applying this method).

2.5. Incorporating affine adaptability

In the introduction we discussed the popularity and success of affine adaptability, also known as linear decisions rules. We can express this type of adaptability in the context of our two-stage problem (4) with the substitution

$$\mathbf{x}^2(\boldsymbol{\xi}) = \mathbf{F}\boldsymbol{\xi} + \mathbf{g}, \tag{18}$$

where $\mathbf{F}$ and $\mathbf{g}$ are now the decision variables instead of $\mathbf{x}^2$ itself. There are two problems with substituting (18) directly into (4). The first is that if $\mathbf{a}_i^2(\boldsymbol{\xi})$ and $\mathbf{c}^2(\boldsymbol{\xi})$ are not constant with respect to the uncertain parameters then we will have quadratic products of uncertain parameters, which is not generally tractable (Ben-Tal et al. 2004). The second problem is that if a variable $x_j^2(\boldsymbol{\xi})$ is integer then $\mathbf{F}_j$ must necessarily be zero for any non-trivial uncertainty set and so there is no adaptability.

We will thus separate our second stage decisions into two sets: continuous decisions $\mathbf{x}^2$ and integer decisions $\mathbf{y}^2$, and apply our affine substitution only to the continuous decisions. This leads to an affine adaptability form of the fully adaptive problem (4):

$$\begin{aligned}
\min_{\mathbf{F},\mathbf{g},\mathbf{x}^1,\mathbf{y},z} \quad & z && (19)\\
\text{subject to} \quad & \mathbf{c}^1(\boldsymbol{\xi})\cdot\mathbf{x}^1 + \mathbf{c}_x^2\cdot(\mathbf{F}\boldsymbol{\xi}+\mathbf{g}) + \mathbf{c}_y^2(\boldsymbol{\xi})\cdot\mathbf{y}^2(\boldsymbol{\xi}) \le z && \forall \boldsymbol{\xi}\in\Xi\\
& \mathbf{a}_i^1(\boldsymbol{\xi})\cdot\mathbf{x}^1 + \mathbf{a}_{x,i}^2\cdot(\mathbf{F}\boldsymbol{\xi}+\mathbf{g}) + \mathbf{a}_{y,i}^2(\boldsymbol{\xi})\cdot\mathbf{y}^2(\boldsymbol{\xi}) \le b_i(\boldsymbol{\xi}) && \forall \boldsymbol{\xi}\in\Xi,\ i\in\{1,\dots,m\}\\
& \mathbf{F}\in\mathbb{R}^{n_x\times n_\xi}, \mathbf{g}\in\mathbb{R}^{n_x}, \mathbf{y}\in\mathbb{Z}^{n_y}.
\end{aligned}$$

A key observation with respect to our finite partitioning method is that $\mathbf{F}$ and $\mathbf{g}$ are themselves decisions that can be deferred until the uncertainty has been realized, and so they too are second stage variables. This allows use to combine finite adaptability and affine adaptability, as we can associate a different affine policy $(\mathbf{F}_j, \mathbf{g}_j)$ for each partition. Doing so produces a piecewise linear policy: for example, for two partitions we would have

$$\mathbf{x}^2(\boldsymbol{\xi}) = \begin{cases} \mathbf{F}_1\boldsymbol{\xi} + \mathbf{g}_2, & \boldsymbol{\xi} \in \Xi\left(\hat{\boldsymbol{\xi}}_1\right), \\ \mathbf{F}_2\boldsymbol{\xi} + \mathbf{g}_2, & \boldsymbol{\xi} \in \Xi\left(\hat{\boldsymbol{\xi}}_2\right), \end{cases} \tag{20}$$

for continuous second stage decisions, and a piecewise constant policy

$$\mathbf{y}^2(\boldsymbol{\xi}) = \begin{cases} \mathbf{y}_1^2, & \boldsymbol{\xi} \in \Xi\left(\hat{\boldsymbol{\xi}}_1\right), \\ \mathbf{y}_2^2, & \boldsymbol{\xi} \in \Xi\left(\hat{\boldsymbol{\xi}}_2\right), \end{cases} \tag{21}$$

for integer decisions. We demonstrate combining affine adaptability with finite adaptability in our mulitstage lot sizing computational example in Section 5.2.

3. Analysis of the partition-and-bound method

In this section, we characterize the partition-and-bound method proposed above from a theoretical point of view. In particular, we seek to understand how the algorithm approaches the ideal fully adaptive solution, and to what degree the number of partitions grows at each iteration.

3.1. Convergence Properties

We now aim to understand how $z_{alg}(\mathcal{T}^k)$ varies with the iteration count k.

Proposition 2. *The upper bound $z_{alg}(\mathcal{T}^k)$ will never increase as k increases.*

Proof of Proposition 2 This follows from the "nested" nature of our partitions. Consider the solution of the partitioned problem at iteration k, e.g. $\bar{\mathbf{x}}^1, \bar{\mathbf{x}}_1^2, \bar{\mathbf{x}}_2^2, \ldots$. At iteration $k+1$ we will further subpartition each of these partitions, resulting in new second stage variables. We can construct a feasible solution for iteration $k+1$ by setting the value of each of these new variables for each subpartition to the same value as the decision variables associated with their respective parent partition. This new solution is feasible by construction and has the same objective value, so we will at least match the objective of iteration k at iteration $k+1$. $\square$

This property has computational implications, as it allows us to provide an initial feasible solution at iteration $k+1$ using the solution at iteration k. This can allow integer optimization solvers to truncate portions of the search tree faster, and reach lower integrality gaps earlier.

We have established that we will never get worse, and it is also possible to identify families of problems (in particular, problems with only continuous decision variables) where we will smoothly converge to the fully adaptive solution. Instead of identifying these, we instead provide the following counterexample that demonstrates that there does not exist a general guarantee that we will improve the bound gap for any finite number of iterations, even if we start at far from the lower bound, for linear AMIO problems. To demonstrate this, consider the parametric family of problems

$$\begin{aligned} z(\epsilon) = \min_{x^2\in\{0,1\},y^2\in\{0,1\},z} \quad & z && (22)\\ \text{subject to} \quad & x^2(\xi) + y^2(\xi) \le z && \forall \xi \in [0,1] \\ & x^2(\xi) \ge \frac{\epsilon - \xi}{\epsilon} && \forall \xi \in [0,1] \\ & y^2(\xi) \ge \frac{\epsilon + \xi - 1}{\epsilon} && \forall \xi \in [0,1], \end{aligned}$$

where $\epsilon \in [0,1]$. This problem has a fully adaptive solution of $z = 1$ and

$$x^2(\xi) = \begin{cases} 1, & 0 \le \xi \le \epsilon, \\ 0, & \epsilon < \xi \le 1, \end{cases} \qquad y^2(\xi) = \begin{cases} 0, & 0 \le \xi \le \epsilon, \\ 1, & \epsilon < \xi \le 1. \end{cases}$$

Note that the static policy sets both $x^2 = y^2 = 1$, and thus $z = 2$. The initial bound gap is 1 as any random $\hat{\xi}$ used to calculate the lower bound will correctly identify it to be $z_{lower} = 1$. Both variables share the same "breakpoint" of ϵ, which means a simple two-partition finitely adaptive solution could match the fully adaptive solution, if only such a partition could be identified.

Proposition 3. *The partition-and-bound method will never find a solution equivalent to the fully adaptive solution for* (22)*, or decrease the bound gap, unless* $\epsilon = q \times 2^{-p}$ *for some* $p, q \in \mathbb{Z}^+$.

Proof of Proposition 3 First note that the two extreme points of each partition are always the active uncertain parameters (the low end from the second constraint, and the high end from the third). Thus at the beginning of iteration k we will have 2^{k-1} partitions of width 2^{1-k}. For all partitions $\Xi(\xi) = [a,b]$ such that $b < \epsilon$ the corresponding decision variable values are $x^2 = 1$ and $y^2 = 0$, with the corresponding lower bound of 1 on the objective z. For all partitions such $a > \epsilon$ we have decision variable values $x^2 = 0$ and $y^2 = 1$, also with the corresponding lower bound of 1 on the objective z. The remaining partition then has the property that $a \le \epsilon \le b$. If the inequalities are strict, then we must set both $x^2 = y^2 = 1$, and thus do not improve over the static policy. However if $\epsilon = a$ or $\epsilon = b$ then we can match the fully adaptive solution because the "breakpoint" is correct. As a and b are always of the form $q \times 2^{-p}$ for nonnegative integers p and q, this condition can only be satisfied if ϵ is also drawn from this family. Any other choice of ϵ, such as 1/3, will never be matched for any finite number of iterations. $\square$

This result also applies to the method of Postek and Den Hertog (2014), using similar reasoning. While this result is a negative one, we believe that such structures are uncommon in problems of interest, and may be avoidable by making different modeling choices when they do occur.

3.2. Growth in Partitioning Scheme

For a problem with m uncertain constraints and an uncertain objective the number of partitions with our method is at most $(m+1)^{k-1}$ at iteration k (and may be much lower if the same uncertain parameter is active for multiple constraints). In contrast, the proposal of Postek and Den Hertog (2014) will have at most 2^{k-1}, but may require more iterations to achieve a solution of the same quality. Here we show by example that the growth rate of the size partitioning scheme is of secondary concern to the relationship between computational effort and quality of solution.

Consider the following example problem, which was presented in Bertsimas and Goyal (2010):

$$\begin{aligned} z(n) = \min_{\mathbf{x}^2 \ge 0, z} \quad & z \\ \text{subject to} \quad & \mathbf{e} \cdot \mathbf{x}^2(\boldsymbol{\xi}) \le z \quad \forall \boldsymbol{\xi} \in \Xi \\ & \mathbf{x}^2(\boldsymbol{\xi}) \ge \boldsymbol{\xi} \quad \forall \boldsymbol{\xi} \in \Xi, \end{aligned} \tag{23}$$

where $\Xi = \left\{ \boldsymbol{\xi} \,\middle|\, \sum_{i=1}^n \xi_i \le 1, \boldsymbol{\xi} \ge 0 \right\}$. The initial static policy has an objective value of n compared to the fully adaptive policy objective value of 1. The upper bound provided by our method in the second iteration is $\frac{n+1}{2}$, which is obtained using n partitions. On the other hand, the method of Postek and Den Hertog (2014) will have 2^{k-1} partitions at iteration k. If we assume that time to solve this problem is proportional to the number of partitions then in the same time it takes to solve the second iteration for our method (n time units), we can do approximately $\log_2(n)$ iterations of Postek and Den Hertog (2014)'s method. However we can show (Appendix A) that it requires n iterations of that method (and thus 2^{n-1}) partitions to achieve the same objective as our method, demonstrating the need to focus on the solution quality versus time rather than iteration or partition counts.

In the above example all partitions had the same objective contribution, i.e., the overall objective was equal to the objective value for each partition. This is not necessarily always the case, and it is trivial to find cases where this doesn't occur. Recall that the motivation for our method, as presented in Theorem 1, was to identify active uncertain parameters that were "binding" the solution and to separate them. A similar approach can be taken with partitions themselves: if a partition's objective is not restricting the overall objective (the partition is not *active*), then we will not sub-partition it at the next iteration. To identify these partitions we need to encourage a solver to find the best possible solution for each partition. We can achieve this by adding a very small term to the objective, e.g.,

$$\begin{aligned}
z_{alg}\left(\mathcal{T}^k\right) = \min_{\mathbf{x}\in\mathcal{X},z} \quad & z + \epsilon \sum_j \tilde{z}_j && (24)\\
\text{subject to} \quad & \tilde{z}_j \le z && \forall \hat{\boldsymbol{\xi}}_j \in \mathit{Leaves}\left(\mathcal{T}^k\right)\\
& \mathbf{c}^1(\boldsymbol{\xi})\cdot\mathbf{x}^1 + \mathbf{c}^2(\boldsymbol{\xi})\cdot\mathbf{x}^2_j \le \tilde{z}_j && \forall \boldsymbol{\xi}\in\Xi\left(\hat{\boldsymbol{\xi}}_j\right), \forall \hat{\boldsymbol{\xi}}_j \in \mathit{Leaves}\left(\mathcal{T}^k\right)\\
& \mathbf{a}^1_i(\boldsymbol{\xi})\cdot\mathbf{x}^1 + \mathbf{a}^2_i(\boldsymbol{\xi})\cdot\mathbf{x}^2_j \le b_i(\boldsymbol{\xi}) && \forall \boldsymbol{\xi}\in\Xi\left(\hat{\boldsymbol{\xi}}_j\right), \forall \hat{\boldsymbol{\xi}}_j \in \mathit{Leaves}\left(\mathcal{T}^k\right), i\in\mathcal{I},
\end{aligned}$$

where $\tilde{z}_j$ is the objective for partition j and we say a partition is *active* if $\tilde{z}_j = z$.

Consider the following variation on (23) that weights one of the components of the decision above the rest:

$$z(M,n) = \min_{\mathbf{x}^2\ge 0,z} \quad z \qquad (25)$$

$$\text{subject to} \quad (M-1)x_1^2 + \mathbf{e}\cdot\mathbf{x}^2(\boldsymbol{\xi}) \le z \quad \forall \boldsymbol{\xi}\in\Xi$$
$$\mathbf{x}^2(\boldsymbol{\xi}) \ge \boldsymbol{\xi} \quad \forall \boldsymbol{\xi}\in\Xi,$$

where the uncertainty set is the same as in the previous example. The static policy now has an objective value of $M+n-1$ compared to the fully adaptive solution's M. If we now apply our partitioning scheme once, the partition corresponding to $i=1$ will have objective $M+\frac{n-1}{2}$ and all other partitions will have the smaller objective $\frac{M+n}{2}$. At the next iteration this is further reduced to $M+\frac{n-1}{4}$ for the active partition and $\frac{M+n+2}{4}$ for all other partitions. Consider the specific case of $M=n=10$: the initial static solution has objective 19. After partitioning we have one partition with objective 14.5 and the remainder have objective 10, which is equal to the fully adaptive objective. Thus, we should only add children parameters for the active partition, as further subpartitions of the inactive partitions will not have any further effect on the overall objective. If we continue with further iterations we find that the number of partitions grows linearly with the number of iterations in this case. We implement this approach in our computation experiments in Section 5.

4. Partition-and-bound method for multistage AMIO

We now generalize our partitioning method described above for two-stage AMIO to the full multistage case of (2). In the interests of clarity, the order of events in our model is as follows: The here-and-now decision $\mathbf{x}^1$ are made before any uncertainty is realized, and then the first stage uncertain parameters $\boldsymbol{\xi}^1$ are revealed. We make the decision $\mathbf{x}^2$ with the benefit of knowing $\boldsymbol{\xi}^1$, but with no other knowledge of the uncertainty parameters to be revealed later except what is implied by knowing $\boldsymbol{\xi}^1$. This then proceeds until we make the decision $\mathbf{x}^T$ with the benefit of knowing $\boldsymbol{\xi}^1,\dots,\boldsymbol{\xi}^{T-1}$, and the final uncertain parameters $\boldsymbol{\xi}^T$ are then revealed (if applicable). Dependencies between uncertain parameters both within and across time stages can be modeled in the uncertainty set with constraints.

While this model of decision making across time is simple enough to state, modeling these *nonanticipativity restrictions* - that a decision made now cannot be made by using exact knowledge of the future - is the primary complication that we address in this section as we extend our approach to the multistage case. We note that enforcing these restrictions is trivial for some classes of adaptability. For example, for affine adaptive policies we can restrict the policy to be a function of only the appropriate components of $\boldsymbol{\xi}$. It is less obvious how to proceed with finite adaptability: The following example demonstrates how a naive application of the method developed in Section 2 will either produce overly-optimistic solutions if we fail to recognize a need for nonanticipativity constraints, or will produce pessimistic solutions where the choice of partitioning scheme and nonanticipativity constraints combine to result in less adaptability than we may like.

Example 2. Consider a three-stage ($T=3$) version of the inventory problem of Section 2.4. The problem proceeds exactly as before, except there are two rounds of demand, two rounds of continuous order decisions, and two rounds of lot ordering decisions. The holding cost is paid on the stock remaining after both rounds of demand and all parameters are the same as before, resulting in the three-stage AMIO

$$\begin{aligned}
\min \quad & z \qquad\qquad (26)\\
\text{subject to} \quad & 50x^1 + 65I^2\left(\xi^1\right) + 1500y_A^2\left(\xi^1\right) + 1875y_B^2\left(\xi^1\right) + \\
& 50x^2\left(\xi^1\right) + 65I^3\left(\xi^1,\xi^2\right) + 1500y_A^3\left(\xi^1,\xi^2\right) + 1875y_B^3\left(\xi^1,\xi^2\right) \le z \qquad \forall \boldsymbol{\xi} \in \Xi \\
& I^2\left(\xi^1\right), I^3\left(\xi^1,\xi^2\right), x^1, x^2\left(\xi^1\right) \ge 0 \qquad \forall \boldsymbol{\xi} \in \Xi \\
& y_A^2\left(\xi^1\right), y_B^2\left(\xi^1\right), y_A^3\left(\xi^1,\xi^2\right), y_B^3\left(\xi^1,\xi^2\right) \in \{0,1\} \quad \forall \boldsymbol{\xi} \in \Xi,
\end{aligned}$$

where

$$\begin{aligned}
I^2\left(\xi^1\right) &= x^1 - \xi^1 + 25y_A^2\left(\xi^1\right) + 25y_B^2\left(\xi^1\right), \\
I^3\left(\xi^1,\xi^2\right) &= I^2\left(\xi^1\right) + x^2\left(\xi^1\right) - \xi^2 + 25y_A^3\left(\xi^1,\xi^2\right) + 25y_B^3\left(\xi^1,\xi^2\right),
\end{aligned}$$

and $\Xi = \{5 \le \xi^1, \xi^2 \le 95\}$. We begin by solving the static policy version of this problem (no partitions), and obtain a solution with objective value 27050 and three active uncertain parameters: $\hat{\boldsymbol{\xi}}_1 = (5,5)$ (for the objective constraint), $\hat{\boldsymbol{\xi}}_2 = (95,5)$ (for the non-negativity constraint on I^2) and

$\hat{\boldsymbol{\xi}}_3 = (95, 95)$ (for the non-negativity constraint on I^3). We now construct partitions exactly as before, ignoring the multistage nature of the uncertain parameters, resulting in the three partitions

$$\Xi\left(\hat{\boldsymbol{\xi}}_1 = (5,5)\right) = \left\{\boldsymbol{\xi} \,\middle|\, 5 \leq \xi^1 \leq 50, \xi^1 + \xi^2 \leq 100\right\},$$

$$\Xi\left(\hat{\boldsymbol{\xi}}_2 = (95,5)\right) = \left\{\boldsymbol{\xi} \,\middle|\, 50 \leq \xi^1 \leq 95, 5 \leq \xi^2 \leq 50\right\},$$

$$\Xi\left(\hat{\boldsymbol{\xi}}_3 = (95,95)\right) = \left\{\boldsymbol{\xi} \,\middle|\, 5 \leq \xi^1 \leq 95, \xi^1 + \xi^2 \geq 100\right\},$$

which are visualized in Figure 4. As this is a multistage problem we now consider nonanticipativity. We proceed by creating a set of variables for each partition, for example x_1^2, x_2^2, and x_3^2 for the second-stage decision $x^2(\xi^1)$. Recall that the interpretation of finite adaptability is that if the realized value of $\boldsymbol{\xi}$ falls in $\Xi\left(\hat{\boldsymbol{\xi}}_1\right)$, then we will select x_1^2 as the implemented decision, and similarly for the other partitions. With this in mind, we consider the case when $\xi^1 = 15$. This value of ξ^1 may fall in either $\Xi\left(\hat{\boldsymbol{\xi}}_1\right)$ or $\Xi\left(\hat{\boldsymbol{\xi}}_3\right)$, but we cannot determine which decision to implement without knowing ξ^2, which is yet to occur. We must then add the nonanticipativity restriction that $x_1^2 = x_3^2$. Similar logic applies to $\xi^1 = 85$, which may be in either $\Xi\left(\hat{\boldsymbol{\xi}}_2\right)$ or $\Xi\left(\hat{\boldsymbol{\xi}}_3\right)$, which requires us to add the restriction $x_2^2 = x_3^2$. These restrictions combine to leave us with no adaptability at all in our second-stage decisions ($x_1^2 = x_2^2 = x_3^2$). The third-stage variables are not affected by these problems as, by that time, we know both ξ^1 and ξ^2 and can thus unambiguously identify the partition and the decision we will implement.

4.1. Multistage partitions and nonanticipativity constraints

A naive application of our existing partitioning scheme will result in overly conservative solutions after the application of nonanticipativity constraints. We thus seek a minimal modification of our existing scheme that retains as much of the benefit of partitioning as possible when these constraints are enforced. We will again have a partition for each leaf of our tree $\mathcal{T}$, and we will use the same tree operators and terminology we established previously. The changes are in how we convert the leaves into partitions, and are coupled with how we identify the required nonanticipativity constraints.

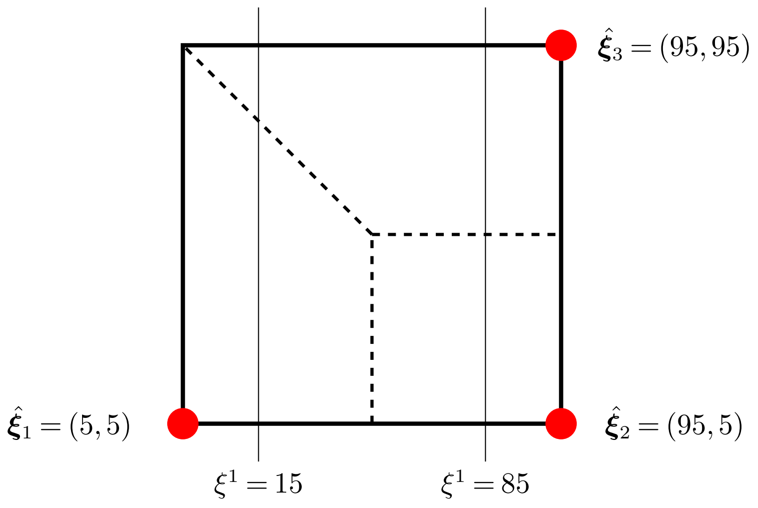

$\hat{\boldsymbol{\xi}}_3 = (95,95)$
$\hat{\boldsymbol{\xi}}_1 = (5,5)$
$\hat{\boldsymbol{\xi}}_2 = (95,5)$
$\xi^1 = 15$
$\xi^1 = 85$

Figure 3 Partitioning of the uncertainty set in Example 2 using the two-stage partitioning scheme without any adjustments for the multistage case. The thin solid lines for $\xi^1 = 15$ and $\xi^1 = 85$ demonstrate that knowing both components ξ^1 and ξ^2 is required to fully identify a partition, requiring the addition of nonanticipativity constraints that remove many of the benefits of partitioning scheme.

Our scheme is very similar to the two-stage version presented previously, but with restrictions at each layer of the tree to only certain elements of the uncertain parameters. In particular, the partition induced by a leaf $\hat{\boldsymbol{\xi}}_i$ is defined to be

$$\begin{aligned}\Xi\left(\hat{\boldsymbol{\xi}}_i\right) = \quad & \left\{\boldsymbol{\xi} \;\middle|\; \left\|\hat{\boldsymbol{\xi}}_i^{t_{i,j}} - \boldsymbol{\xi}^{t_{i,j}}\right\|_2 \le \left\|\hat{\boldsymbol{\xi}}_j^{t_{i,j}} - \boldsymbol{\xi}^{t_{i,j}}\right\|_2 \quad \forall \hat{\boldsymbol{\xi}}_j \in Siblings\left(\hat{\boldsymbol{\xi}}_i\right)\right\} \\ & \cap \left\{\boldsymbol{\xi} \;\middle|\; \left\|Parent\left(\hat{\boldsymbol{\xi}}_i\right)^{t'_{i,j}} - \boldsymbol{\xi}^{t'_{i,j}}\right\|_2 \le \left\|\hat{\boldsymbol{\xi}}_j^{t'_{i,j}} - \boldsymbol{\xi}^{t'_{i,j}}\right\|_2 \quad \forall \hat{\boldsymbol{\xi}}_j \in Siblings\left(Parent\left(\hat{\boldsymbol{\xi}}_i\right)\right)\right\} \\ & \cdots \cap \Xi.\end{aligned} \tag{27}$$

We define $t_{i,j}$ for a pair of uncertain parameters $\hat{\boldsymbol{\xi}}_i$ and $\hat{\boldsymbol{\xi}}_j$ as $\arg\min_t \left\{\hat{\boldsymbol{\xi}}_i^t \neq \hat{\boldsymbol{\xi}}_j^t\right\}$, that is the first time stage for which the uncertain parameters differ. $t'_{i,j}$ is the same but for the one level up the tree, i.e., the first time stage t such that $Parent\left(\hat{\boldsymbol{\xi}}_i\right)^t \neq \hat{\boldsymbol{\xi}}_j^t$ for $\hat{\boldsymbol{\xi}}_j^t \in Siblings\left(Parent\left(\hat{\boldsymbol{\xi}}_i\right)\right)$, and so on.

We must also present a complementary scheme for enforcing nonanticipativity. We claim the following for two partitions of the uncertainty set $\Xi\left(\hat{\boldsymbol{\xi}}_i\right)$ and $\Xi\left(\hat{\boldsymbol{\xi}}_j\right)$ induced by the leaf uncertain parameters $\hat{\boldsymbol{\xi}}_i$ and $\hat{\boldsymbol{\xi}}_j$.

Proposition 4. *If there exists $\boldsymbol{\psi} = (\boldsymbol{\psi}^1, \dots, \boldsymbol{\psi}^T) \in \Xi\left(\hat{\boldsymbol{\xi}}_i\right)$ and $\boldsymbol{\phi} = (\boldsymbol{\phi}^1, \dots, \boldsymbol{\phi}^T) \in \Xi\left(\hat{\boldsymbol{\xi}}_j\right)$ such that $\boldsymbol{\psi}^s = \boldsymbol{\phi}^s\ \forall s \in \{1, \dots, t-1\}$, and at least one of $\boldsymbol{\psi} \in int\left(\Xi\left(\hat{\boldsymbol{\xi}}_i\right)\right)$ and $\boldsymbol{\phi} \in int\left(\Xi\left(\hat{\boldsymbol{\xi}}_j\right)\right)$ holds, then we must enforce nonanticipativity constraints for the corresponding decisions at time stage t as the two partitions can not be distinguished with the uncertain parameters realized by that time stage. Otherwise, we do not need to enforce any restrictions at time stage t for this pair.*

Proof of Proposition 4 If the condition holds, then there exists a partial realization of the uncertain parameters which could lie in two different partitions, based on known information, so we must restrict the decisions for those partitions to be the same. If the equality condition holds, but only for a point on the boundary of both partitions, then both decisions are valid and interchangeable and we gain no benefit from future knowledge separating the two anyway. If the equality condition does not hold then the partitions are distinguishable and no constraint is required. □

Given this condition, we can now propose a simple routine for determining which constraints to add. For the sake of exposition we will focus on polyhedral uncertainty sets, although a similar routine could be constructed for more general sets. In particular, for each pair of leaf parameters $\hat{\boldsymbol{\xi}}_i$ and $\hat{\boldsymbol{\xi}}_j$ and their corresponding induced partitions, expressed as

$$\Xi\left(\hat{\boldsymbol{\xi}}_i\right) = \{\boldsymbol{\xi} \in \Xi \,|\, \mathbf{F}\boldsymbol{\xi} \leq \mathbf{f}\} \quad \text{and} \quad \Xi\left(\hat{\boldsymbol{\xi}}_j\right) = \{\boldsymbol{\xi} \in \Xi \,|\, \mathbf{G}\boldsymbol{\xi} \leq \mathbf{g}\},$$

we can formulate the optimization problem

$$\begin{aligned} z(t) = \max_{\boldsymbol{\psi},\boldsymbol{\phi},\alpha\geq 0,\beta\geq 0} \quad & \alpha + \beta \\ \text{subject to } & F\boldsymbol{\psi} + \alpha\mathbf{e} \leq \mathbf{f} \\ & G\boldsymbol{\phi} + \beta\mathbf{e} \leq \mathbf{g} \\ & \boldsymbol{\psi}^s = \boldsymbol{\phi}^s \quad \forall s \in \{1, \dots t-1\}, \end{aligned} \tag{28}$$

the solution to which informs us if nonanticipativity constraints are necessary at time stage t. If the problem is infeasible then they are not required, as the first two of constraints can always be satisfied but the third may not necessarily be satisfiable. If there is a feasible solution but the objective value is 0 then all feasible solutions are on the boundaries of the uncertainty sets, which does not require the constraints. If the objective value is positive then we must enforce the nonanticipitavity constraints as there is an interior point for the partitions that is identical up to time t.

Finally, we present a simple alternative rule by which the need for nonanticipativity constraints can be determined. This rule is not a general method for enforcing nonanticipativity in finite adaptability as it relies on the particular structure produced by our partitioning scheme. Given two partitions $\Xi\left(\hat{\boldsymbol{\xi}}_i\right)$ and $\Xi\left(\hat{\boldsymbol{\xi}}_j\right)$, the rule for whether we need to enforce the nonanticipativity constraints $\mathbf{x}_i^t = \mathbf{x}_j^t$ is as follows:

1. Let $\hat{\boldsymbol{\xi}}_A \leftarrow \hat{\boldsymbol{\xi}}_i$ and $\hat{\boldsymbol{\xi}}_B \leftarrow \hat{\boldsymbol{\xi}}_j$.
2. If $\hat{\boldsymbol{\xi}}_A$ and $\hat{\boldsymbol{\xi}}_B$ **are** siblings, then let $t^{A,B} = \arg\min_s \left\{\hat{\boldsymbol{\xi}}_A^s \neq \hat{\boldsymbol{\xi}}_B^s\right\}$. If $t > t^{A,B}$, then we **do not** need to enforce the nonanticipativity constraints. If $t \leq t^{A,B}$, then we **do**.
3. Otherwise, if $\hat{\boldsymbol{\xi}}_A$ and $\hat{\boldsymbol{\xi}}_B$ **are not** siblings, let $\hat{\boldsymbol{\xi}}_A \leftarrow Parent\left(\hat{\boldsymbol{\xi}}_A\right)$ and $\hat{\boldsymbol{\xi}}_B \leftarrow Parent\left(\hat{\boldsymbol{\xi}}_B\right)$ and go to Step 2.

This rule is simple to implement, but can be overconservative as it may declare nonanticipativity constraints as required when they are not needed. Despite this shortcoming, it could be used to quickly to determine the cases where the constraints are not required before the above optimization approach can definitively determine whether they are truly required. We provide proof of the correctness of the rule, as well as an example demonstrating overconservatism, in Appendix B.

4.2. Multistage method and implementation

In the interests of space we will not present the multistage method in full, as it suffices to simply take the definition of the method for the two-stage case, but use the multistage partitioning scheme and add nonanticipativity constraints wherever they are required. We can also extend our lower bound approach to the multistage case, in particular

$$\begin{aligned} z_{lower}\left(\mathcal{T}\right) = \min_{\mathbf{x}\in\mathcal{X},z} \quad & z \qquad\qquad (29)\\ \text{subject to} \quad & \sum_{t=1}^{T} \mathbf{c}^t\left(\hat{\boldsymbol{\xi}}_j\right)\cdot \mathbf{x}_j^t \leq z \qquad \forall \hat{\boldsymbol{\xi}}_j \in \mathcal{T} \end{aligned}$$

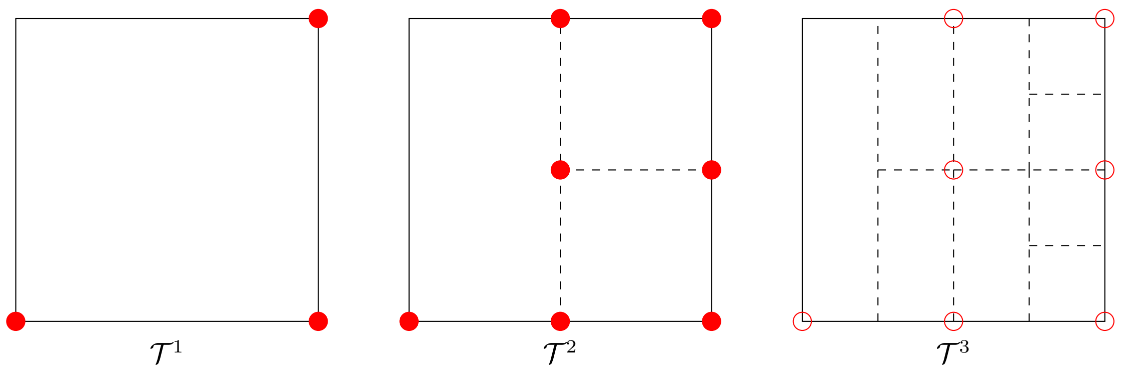


Figure 4 **Partitioning of the uncertainty set in Example 2 using the multistage partitioning scheme. Note that at the second iteration we have two distinct second-stage decisions, unlike in Figure 4 where we had only one. When we partition again we grow to four distinct second-stage decisions, and nine distinct third-stage decisions.**

$$\sum_{t=1}^{T} \mathbf{a}_i^t\left(\hat{\boldsymbol{\xi}}_j\right)\cdot \mathbf{x}_j^t \le b_i\left(\hat{\boldsymbol{\xi}}_j\right) \qquad \forall \hat{\boldsymbol{\xi}}_j \in \mathcal{T}, i \in I$$

$$\mathbf{x}_j^t = \mathbf{x}_k^t \qquad \forall \hat{\boldsymbol{\xi}}_j, \hat{\boldsymbol{\xi}}_k \in \mathcal{T}, \forall t : \hat{\boldsymbol{\xi}}_j^{1,\ldots,t-1} = \hat{\boldsymbol{\xi}}_k^{1,\ldots,t-1},$$

where the final set of constraints enforces the nonanticipativity restrictions. If two samples $\hat{\boldsymbol{\xi}}_j$ and $\hat{\boldsymbol{\xi}}_k$ are identical up to and including time $t-1$, then the corresponding decisions must be restricted to be equal at time t as we would not have any extra information to inform our decision. For the edge case $t = 1$ we enforce that all decisions are the same, as no uncertainty has been revealed yet.

We provide a worked example of applying the multistage algorithm to the lot sizing problem in Appendix C. This example demonstrates the substantial difference between our method and the method of Postek and Den Hertog (2014), which must employ various heuristics to avoid becoming overconservative. For example, applying the multistage method of Postek and Den Hertog (2014) to the three-stage Example 2 would create two partitions instead of three, and would allow for no adaptability in the second-stage, regardless of the number of iterations. In contrast, our method still allows for three partitions at the first iteration and is able to retain adaptability, as visually demonstrated in Figure 4.

5. Computational Experiments

The goal of this section is to demonstrate that, for problems of interest from the literature, the methods described in this paper are practical and valuable. To do so we consider a two-stage binary capital budgeting problem (Section 5.1) and a mulitstage lot sizing problem (Section 5.2). All methods were implemented in the Julia programming language using the JuMP (Lubin and Dunning 2015) and JuMPeR (Dunning 2015) packages, with Gurobi 6.0.4 used as the solver (Gurobi Optimization 2015).

5.1. Capital Budgeting

Our first example is the two-stage capital budgeting problem described by Hanasusanto et al. (2014). We must decide which of N projects to pursue to maximize profits, but are constrained by a known budget B. The cost c_i and profit r_i for each project i are affine functions of uncertain parameters $\boldsymbol{\xi}$,

$$c_i\left(\boldsymbol{\xi}\right)=\left(1+\frac{1}{2}\boldsymbol{\Phi}_i\cdot\boldsymbol{\xi}\right)c_i^0 \qquad \text{and} \qquad r_i\left(\boldsymbol{\xi}\right)=\left(1+\frac{1}{2}\boldsymbol{\Psi}_i\cdot\boldsymbol{\xi}\right)r_i^0.$$

We have the option of pursuing a project either before or after these uncertain parameters are observed. If we pursue a project after they are observed then we generate only a fraction $\theta\in[0,1)$ of the profit - a penalty for delaying the decision. This can be expressed as the adaptive binary optimization problem

$$\begin{aligned}
\max_{z,\mathbf{x}} \quad & z \\
\text{subject to} \quad & \mathbf{r}\left(\boldsymbol{\xi}\right)\cdot\left(\mathbf{x}^1+\theta\mathbf{x}^2\left(\boldsymbol{\xi}\right)\right)\geq z && \forall\boldsymbol{\xi}\in\Xi \\
& \mathbf{c}\left(\boldsymbol{\xi}\right)\cdot\left(\mathbf{x}^1+\mathbf{x}^2\left(\boldsymbol{\xi}\right)\right)\leq B && \forall\boldsymbol{\xi}\in\Xi \\
& \mathbf{x}^1+\mathbf{x}^2\left(\boldsymbol{\xi}\right)\leq\mathbf{e} && \forall\boldsymbol{\xi}\in\Xi \\
& \mathbf{x}^1,\mathbf{x}^2\left(\boldsymbol{\xi}\right)\in\{0,1\}^N && \forall\boldsymbol{\xi}\in\Xi,
\end{aligned}$$

where $\mathbf{x}^1$ is the first-stage decision to pursue now, and $\mathbf{x}^2$ is the second-stage decision to pursue later. The uncertainty set is the hypercube $\Xi=\left\{\boldsymbol{\xi}\,\middle|\,\boldsymbol{\xi}\in[-1,1]^4\right\}$. We used the same distributions for the parameters as Hanasusanto et al. (2014), and set Gurobi's integrality gap tolerance to 1%

for all instances. All robust optimization problems were solved by reformulation to a deterministic problem.

We analyze our results from both the perspective of the decrease in gap versus total time, but also by "improvement" versus total time, where we use the same definition of improvement as Hanasusanto et al. (2014):

$$\frac{z_{alg}\left(\mathcal{T}^k\right) - z_{alg}\left(\mathcal{T}^1\right)}{z_{alg}\left(\mathcal{T}^1\right)}.$$

Before comparing results with Hanasusanto et al. (2014) for larger problem sizes we first tried to understand the effects of only subpartitionining active partitions, which we define for this problem as the partitions that have an objective value that is within 10^{-2} of the worst-case objective value. As can be seen in Figure 5, for 100 instances of size $N = 5$ we see that while subpartitioning all partitions leads to the number of partitions growing exponentially, focusing on only the active partitions leads to a near-linear increase in partition count. As discussed in Section 3.2, more important than partition count is the solution quality versus time. The right side of Figure 5 demonstrates this clearly, as four and eight iterations of each approach both take approximately 0.1 seconds but the active partition-only approach delivers superior solutions on average.

For the purpose of comparison we considered 50 random instances of each size $N \in \{5, 10, 20, 30\}$, and in Figure 6 we plot improvement and gap versus time for each instance. We found that there is little variability across instances in performance for $N \geq 10$, and that as N increases a pattern emerges where substantial initial improvement is made, before a multiple-iteration stall, followed by further improvement. It seems reasonable to suggest that this pattern is driven by the ratio of N to the fixed dimension of the uncertainty set, which is why the behaviour for $N = 5$ differs substantially from the rest.

For instances of size $N = 30$ Hanasusanto et al. (2014) found an improvement of approximately 65% with their two-partition solution, 90% with their three-partition solution, and 110% with their four-partition solution. However, even with only $N = 20$ almost all instances failed to solve to optimality using their method within 2 hours. In contrast we are able to consistently achieve an 80%

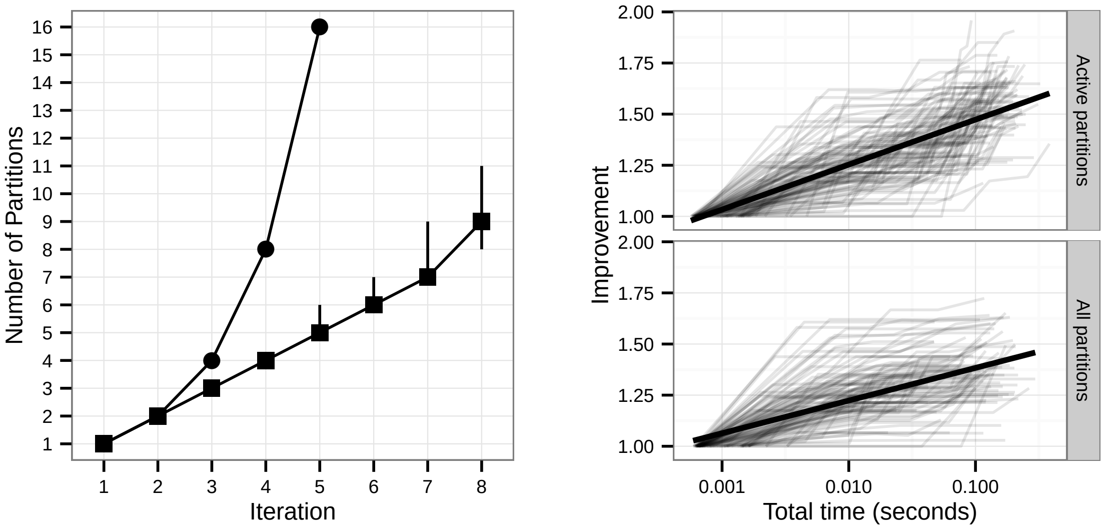


Figure 5 **Left: median number of partitions versus iteration number if all partitions are subpartitioned (circles) or only active partitions are (squares). The top of the vertical line is the 90^{th} percentile, and the bottom is the 10^{th} percentile; there is no variation when all partitions are subpartitioned. Right: improvement versus time for the two approaches. Each semi-transparent line represents the improvement across time for a single instance, and the solid black line is the line of best fit. Partitioning only on active partitions delivers better quality solutions on average than partitioning all partitions for the same computation time.**

improvement for instances of size $N = 30$ in under 1 second (which corresponds to a 30% bound gap). We suspect that this drastic difference in performance arises from the structure of the integer optimization problems being solved by the two methods. Our approach has a very simple block structure (one block per partition) not too dissimilar from the deterministic equivalent, whereas Hanasusanto et al. (2014) must introduce "big-M" constraints to allow for the partition structure to be included in the optimization, creating a computationally difficult problem. The two approaches are complementary, however: the method of Hanasusanto et al. (2014) would be preferable if having the best possible or a "simple" solution was important, such as when the second-stage decisions must be implemented as-is and we cannot reoptimize.

5.2. Multistage lot sizing

The multistage lot sizing problem described in Bertsimas and Georghiou (2015) is a challenging AMIO problem that features a mix of continuous and binary decisions across multiple time stages.

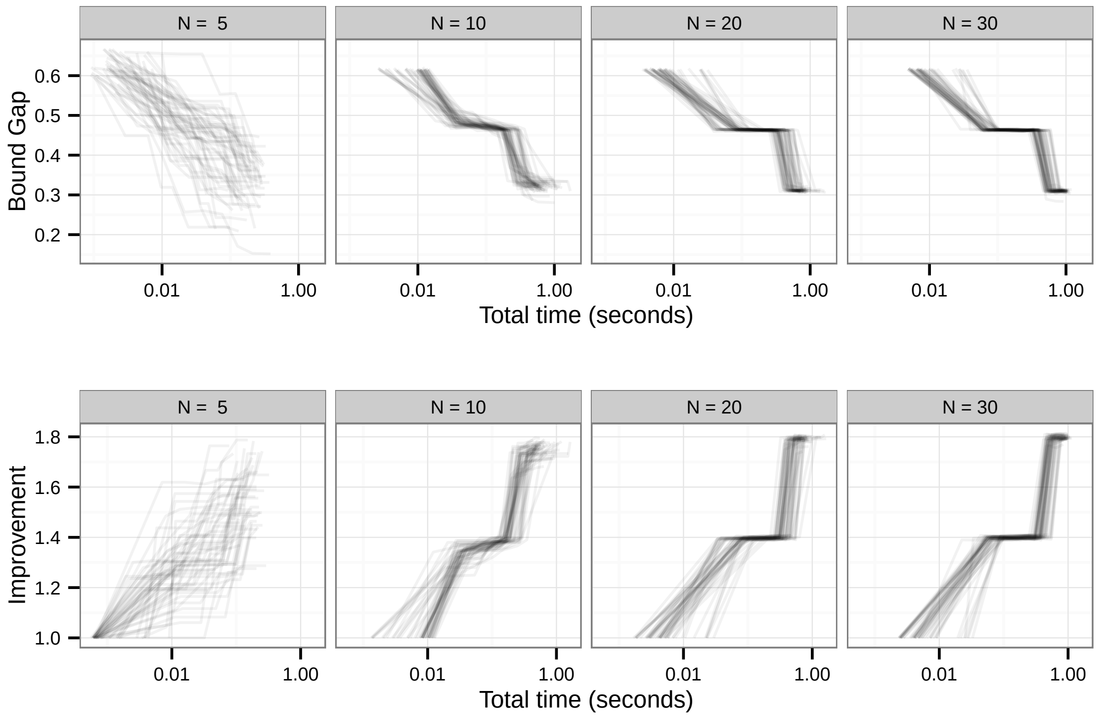


Figure 6 **Improvement and bound gap versus time for 50 instances of size $N \in \{5, 10, 20, 30\}$. Each instance is represented by a semi-transparent line, and darker areas correspond to many overlapping lines.**

We have used smaller versions of this problem throughout this paper as a motivating example. The full formulation is provided in Bertsimas and Georghiou (2015), but we repeat the description here for clarity. Our goal is to minimize the cost of stock ordered plus holding costs, subject to the need to satisfy uncertain demand. We must decide x^t, the (continuous, non-negative) amount of stock ordered at time stage t for delivery at time stage $t+1$ at a unit cost of c_x, and y_n^t, a binary decision indicating whether to order a fixed amount q_n of stock for immediate delivery at time t at a unit cost of c_n. There are N different lot sizes to select from. We do not allow stock levels to become negative, and we impose a unit holding cost c_h for any stock remaining at the end of stage t. The uncertainty set is a box uncertainty set $\Xi = \{\boldsymbol{\xi} \,|\, \xi^1 = 1,\ l^t \le \xi^t \le u^t \quad \forall t \in \{2, \dots, T\}\}$. For the continuous decisions $\mathbf{x}$ we start with an affine policy instead of a static policy, leading to a piecewise affine policy after partitioning (as described in Section 2.5). In order to aid understanding of the implementation of our method for this problem we provide a worked example in Appendix C.

The problem size may grow rapidly, proportional to T^k where k is the iteration count, if our method is applied naively. We address this by only subpartitioning the active partitions. Additionally many of the constraints only contain some the uncertain parameters and as a result the set of active uncertain parameters is a set of infinite cardinality. To select a single active uncertain parameter per constraint we randomly weight the components and solve a small optimization problem to find a random extreme point of the set. We again use reformulation to convert the robust optimization problems to deterministic MIO problems. Gurobi's integrality gap tolerance was set to 1% for all instances, and the time limit was set to $20T$ seconds for instances with T time stages.

We implemented both our method and the method of Postek and Den Hertog (2014), and evaluated them over 50 randomly generated instances for $T \in \{4,6,8,10\}$ for $N=3$ (with results for $N=2$ being qualitatively similar). The method of Postek and Den Hertog (2014) has multiple heuristics and parameters that can be selected: we used the same as were used in that paper's computational experiments, with the exception of the lower bound calculation where we used all found active uncertain parameters (which is what our method does). To make the running times of the two methods roughly equivalent we used 5 iterations of the method of Postek and Den Hertog (2014), and 3 iterations of our method. We also sought to compare our results with Bertsimas and Georghiou (2015), who use a different definition of a bound gap than us:

$$\frac{z_{alg}\left(\mathcal{T}^k\right) - z_{lower}\left(\mathcal{T}^k\right)}{2\left(z_{alg}\left(\mathcal{T}^k\right) + z_{lower}\left(\mathcal{T}^k\right)\right)}. \tag{30}$$

We have summarized the results in two ways. Firstly in Table 1 we compare our method with the methods of Postek and Den Hertog (2014) and Bertsimas and Georghiou (2015). Our method finds superior solutions in only a fraction of the time of Bertsimas and Georghiou (2015), suggesting that while our method of approximating the optimal piecewise affine policy is a far more efficient strategy that trying to directly optimize the pieces. Comparisons with Postek and Den Hertog (2014) are hard to make in this table format, as to be fair we must compare total computation time versus solution quality, not iteration count version solution quality. We visualize the full comparison, with quality measured by bound gap and improvement, in Figure 7. Here we can see that the curves for

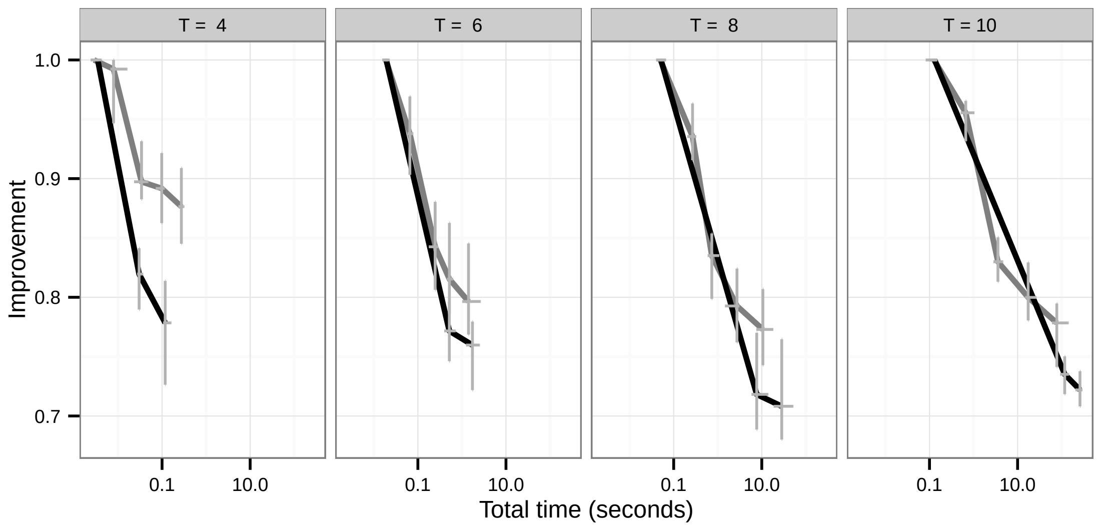
		T			
Method		4	6	8	10
Our method (2 iter.)	Gap (%)	13.0	10.3	11.6	14.9
	Time (s)	0.0	0.5	7.7	108.6
Our method (3 iter.)	Gap (%)	11.4	9.3	11.3	14.9
	Time (s)	0.2	2.0	52.4	309.3
Postek and Den Hertog (2014)	Gap (%)	11.5	14.1	15.7	15.7
	Time (s)	0.4	1.6	10.8	77.8
Bertsimas and Georghiou (2015)	Gap (%)	17.2	34.5	37.6	-
	Time (s)	3381	9181	28743	-

Table 1 Comparison of our method (with two or three iterations) versus the methods of Postek and Den Hertog (2014) and Bertsimas and Georghiou (2015) for the multistage lot sizing problem. "Gap" is calculated as described in Bertsimas and Georghiou (2015), not as in this paper. Time for our method and Postek and Den Hertog (2014) is total time to solve all upper and lower bound problems up to and including that iteration; both gap and time are medians for these two methods. Times for Bertsimas and Georghiou (2015) are obtained from a different computer but with roughly comparable performance, on instances from the same family.

our method in general lie close to or below the curves for Postek and Den Hertog (2014) across all instance sizes, suggesting that our method will produce better solutions on average for the same amount of computational time invested.

6. Concluding Remarks

In this paper, we presented a novel partition-and-bound method for solving adaptive mixed integer optimization problems. We use finite partitioning to approximate the fully adaptable solution, and then use information from the solution of the finitely partitioned problem to further improve the partitioning and calculate lower bounds. The particular Voronoi diagram-like partitioning scheme we employ is inspired by the observation that, under some assumptions, a finite partitioning of the uncertainty set can only improve the solution quality if the partitioning separates regions of the uncertainty set that correspond to binding constraints. We find that our method delivers high-quality solutions for the time spent, even as the problem scales in both number of time stages and in the number of decision variables. The time per iteration does increase with the number of time



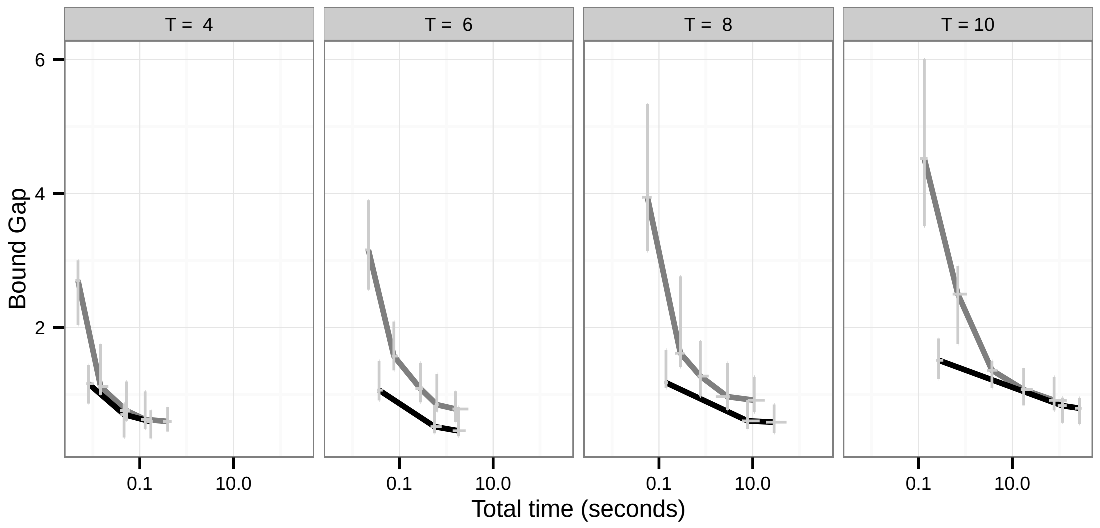


Figure 7 **Time versus improvement and bound gap for the multistage lot sizing problem. The black line and dark grey line correspond to our method and the method of Postek and Den Hertog (2014) respectively. All quantities are medians over 50 instances at each iteration, with the light grey error bars showing the 40^{th} and 60^{th} percentiles. The black curve is generally below the dark grey curve, indicating that our method obtains better results for the same amount of computational effort.**

stages, although we observed that the number of partitions created typically grows much slower than the worst-case by focusing on active partitions. Our method should be viewed as one option among many for AMIO. For example if "good" solutions are required quickly, our method excels. If a near-fully adaptive solution is truly needed, other approaches may be more appropriate. We do note that, in a situation where we can reoptimize our decision at each time stage, perfectly identifying the fully adaptive solution may be unnecessary as the only important decision is the decision that must be made here-and-now.

Finally, we can draw analogies with the branch-and-bound technique for integer optimization. In branch-and-bound we use the linear relaxation of an integer optimization problem to provide a lower bound on the true integer solution, use the information obtained from the solution to branch on fractional variables to obtain upper bounds, and repeat until we are satisfied with the bound gap. In integer optimization the development of new cutting planes and presolve techniques has led to substantial improvements in tightening these bounds gaps faster, and we believe there is exciting future work in developing new techniques for AMIO as extensions of this partition-and-bound approach.

Appendix A: Comparison of partitioning schemes for a large gap example

Consider the following problem, which was presented in Bertsimas and Goyal (2010) to demonstrate large gaps between static and fully adaptive policies:

$$\begin{aligned} z(n) = \min_{\mathbf{x}^2 \geq 0, z} \quad & z \\ \text{subject to} \quad & \mathbf{e} \cdot \mathbf{x}^2(\boldsymbol{\xi}) \leq z \quad \forall \boldsymbol{\xi} \in \Xi \\ & \mathbf{x}^2(\boldsymbol{\xi}) \geq \boldsymbol{\xi} \quad \forall \boldsymbol{\xi} \in \Xi, \end{aligned}$$

where $\Xi = \left\{ \boldsymbol{\xi} \,\middle|\, \sum_{j=1}^n \xi_j \leq 1, \boldsymbol{\xi} \geq 0 \right\}$. The initial static policy solution to this problem has objective value n, compared to the fully adaptive solution 1, and the active uncertain parameter for each component i of $\mathbf{x}$ is the case where $\hat{\xi}_i = 1$ and all other components are zero.

Our method Our method creates n partitions, one for each component i. The partition for $i = 1$ would be

$$\Xi\left(\hat{\boldsymbol{\xi}}_1\right) = \left\{ \boldsymbol{\xi} \,\middle|\, \sum_{i=1}^n \xi_i \leq 1, \boldsymbol{\xi} \geq 0, \xi_1 \geq \xi_2, \ldots, \xi_1 \geq \xi_n \right\},$$

and similarly for the other partitions. If we consider the corresponding solution for this partition we obtain $\mathbf{x}_1^2 = \left(1, \frac{1}{2}, \ldots, \frac{1}{2}\right)$, as while for the first component the worst-case is unchanged ($\hat{\xi}_1 = 1$), the value of ξ_1 must be at least as large as ξ_i which limits the largest value any other component can take to $\frac{1}{2}$. Thus the problem at iteration 2 has n partitions and an objective of $1 + \frac{n-1}{2} = \frac{n+1}{2}$, approximately half the initial objective for $n \gg 1$.

Method of Postek and Den Hertog (2014) The method of Postek and Den Hertog (2014) splits each partition into two sub-partitions at each iteration, by selecting the two active uncertain parameters furthest apart. At the end of the first iteration we can select the active uncertain parameters for components $j=1$ and $j=2$ without loss of generality. This results in two partitions

$$\Xi\left(\hat{\boldsymbol{\xi}}_1\right)=\left\{\boldsymbol{\xi}\,\middle|\,\sum_{i=1}^{n}\xi_i\le 1,\boldsymbol{\xi}\ge 0,\xi_1\ge\xi_2\right\},$$

and

$$\Xi\left(\hat{\boldsymbol{\xi}}_2\right)=\left\{\boldsymbol{\xi}\,\middle|\,\sum_{i=1}^{n}\xi_i\le 1,\boldsymbol{\xi}\ge 0,\xi_1\le\xi_2\right\}.$$

The corresponding solutions are $\mathbf{x}_1^2=\left(1,\frac{1}{2},1,\ldots,1\right)$ and $\mathbf{x}_2^2=\left(\frac{1}{2},1,\ldots,1\right)$ respectively, giving an objective of $n-\frac{1}{2}$. At each subsequent iteration another of the components will be restricted to be no more than $\frac{1}{2}$. Thus the objective at iteration k is $n-\frac{k-1}{2}$ and the the number of partitions is 2^{k-1}. At iteration n the objective will match that of our method $n-\frac{n-1}{2}=\frac{n+1}{2}$ and will have 2^{n-1} partitions.

Appendix B: Admissability of rule for enforcing nonanticipativity

We now demonstrate the rule described in Section 4.1 is sufficient to enforce nonanticipativity, but then show by example that it is overconservative: it will enforce nonanticipativity constraints needlessly in some cases.

Proposition 5. *If $\mathbf{x}_i^t$ and $\mathbf{x}_j^t$ are the decisions at time stage $t\ge 2$ corresponding to the partitions $\Xi\left(\hat{\boldsymbol{\xi}}_i\right)$ and $\Xi\left(\hat{\boldsymbol{\xi}}_j\right)$ respectively, then enforcing $\mathbf{x}_i^t=\mathbf{x}_j^t$ (or not) according to the above decision rule ensures nonanticipativity.*

Proof Consider the case where $\hat{\boldsymbol{\xi}}_i$ and $\hat{\boldsymbol{\xi}}_j$ are siblings. $\mathbf{x}^t$ is a function of the uncertain parameters $\boldsymbol{\xi}^1,\ldots,\boldsymbol{\xi}^{t-1}$. If $t\le t^{A,B}$ then $\hat{\boldsymbol{\xi}}_i^{1,\ldots,t-1}=\hat{\boldsymbol{\xi}}_j^{1,\ldots,t-1}$, and thus we cannot distinguish from the revealed uncertain parameters between $\Xi\left(\hat{\boldsymbol{\xi}}_i\right)$ and $\Xi\left(\hat{\boldsymbol{\xi}}_j\right)$, and so must enforce the nonanticipativity constraints. If $t>t^{A,B}$ then a realization of the uncertain parameters $\boldsymbol{\xi}^1,\ldots,\boldsymbol{\xi}^{t-1}$ can be "assigned" to either one of the partitions (or neither) as a hyperplane based on the $t^{A,B}$ component of $\boldsymbol{\xi}$ separates the two, and so we do not need to enforce nonanticipativity constraints.

If $\hat{\boldsymbol{\xi}}_i$ and $\hat{\boldsymbol{\xi}}_j$ are not siblings, we let $\hat{\boldsymbol{\xi}}_A\leftarrow Parent\left(\hat{\boldsymbol{\xi}}_i\right)$ and $\hat{\boldsymbol{\xi}}_B\leftarrow Parent\left(\hat{\boldsymbol{\xi}}_j\right)$, and update $t_{A,B}$. If $\hat{\boldsymbol{\xi}}_A$ and $\hat{\boldsymbol{\xi}}_B$ are now siblings, then we know that all child partitions are separated by a constraint involving the terms $\boldsymbol{\xi}^{t^{A,B}}$. If $t\le t^{A,B}$, then, using same logic as above, we must conservatively add enforce the nonanticipativity constraints (even though they may be distinguished at a lower level). If $t > t^{A,B}$ then can be assigned to one of the child partitions of either for the same reasons as above. If they are not siblings, we repeat and apply the same logic. □

As suggested in this proof, we may add constraints we do not need due to missing that two leafs might be separated in a way not easily determined just by the tree's structure.

Example 3. Consider the two-stage uncertainty set $\Xi = \{(\xi^1, \xi^2) \,|\, 0 \le \xi^1, \xi^2 \le 10\}$, with tree $\mathcal{T}$

$$\mathcal{T} = root \begin{cases} (10,0), & \begin{cases} (0,0) & : A \\ (10,0) & : B \end{cases} \\ & \\ (10,10), & \begin{cases} (0,10) & : C \\ (10,10) & : D \end{cases} \end{cases}$$

where we have labelled each leaf A through D. The uncertainty sets corresponding to each leaf are

$$\begin{aligned} \Xi\left(\hat{\boldsymbol{\xi}}_A\right) &= \left\{\left(\xi^1, \xi^2\right) | 0 \le \xi^1 \le 5,\ 0 \le \xi^2 \le 5\right\} \\ \Xi\left(\hat{\boldsymbol{\xi}}_B\right) &= \left\{\left(\xi^1, \xi^2\right) | 5 \le \xi^1 \le 10,\ 0 \le \xi^2 \le 5\right\} \\ \Xi\left(\hat{\boldsymbol{\xi}}_C\right) &= \left\{\left(\xi^1, \xi^2\right) | 0 \le \xi^1 \le 5,\ 5 \le \xi^2 \le 10\right\} \\ \Xi\left(\hat{\boldsymbol{\xi}}_D\right) &= \left\{\left(\xi^1, \xi^2\right) | 5 \le \xi^1 \le 10,\ 5 \le \xi^2 \le 10\right\} \end{aligned}$$

which we can manually inspect to determine that we must enforce the nonanticipativity constraints $\mathbf{x}^2_A = \mathbf{x}^2_C$ and $\mathbf{x}^2_B = \mathbf{x}^2_D$. However, consider the application of the decision rule above to this problem for $t = 2$, that is $\mathbf{x}^2\left(\xi^1\right)$. $\hat{\boldsymbol{\xi}}_A$ and $\hat{\boldsymbol{\xi}}_D$ are not siblings, so we must compare their parents (which are). The parents first differ in the second time stage, so we add a nonanticipativity constraint (as they appear indistinguishable knowing only ξ^1). However, this constraint is not required, because only $\xi^1 = 5$ could be in either $\Xi\left(\hat{\boldsymbol{\xi}}_A\right)$ or $\Xi\left(\hat{\boldsymbol{\xi}}_D\right)$, but that is a boundary point. Thus the rule is overconservative in this case. □

Appendix C: Comparison of partitioning schemes for multistage lot sizing

The multistage lot sizing problem of Bertsimas and Georghiou (2015) provides a good case study to understand the differences between the method in this paper and the proposal of Postek and Den Hertog (2014). We will walk through two iterations of both methods for $T = 4$, using the randomly generated uncertainty set $\Xi = \{\xi^1 = 1, 21 \le \xi^2 \le 89, 13 \le \xi^3 \le 87, 4 \le \xi^4 \le 100\}$. The other parameters are not directly relevant to the partitioning schemes, so we will not list them here.

We begin by solving the unpartitioned problem, which has an objective value of $1830\frac{2}{3}$. The unique active uncertain parameters we extract are

$$\hat{\boldsymbol{\xi}}_1 = (1,21,13,4), \quad \hat{\boldsymbol{\xi}}_3 = (1,89,13,4), \quad \text{and } \hat{\boldsymbol{\xi}}_5 = (1,89,87,100).$$
$$\hat{\boldsymbol{\xi}}_2 = (1,21,87,4), \quad \hat{\boldsymbol{\xi}}_4 = (1,89,87,4),$$

At this point the two methods diverge, as our method creates five partitions while the other creates two.

Our method

Iteration 1 Our method considers each pair of active uncertain parameters, pairwise, when constructing partitions. We now detail the construction for $\hat{\boldsymbol{\xi}}_3$. Consider first $\hat{\boldsymbol{\xi}}_3$ and $\hat{\boldsymbol{\xi}}_1$: $t_{3,1}$, the first time stage they differ, is 2. Thus the first constraint is $\xi^2 \ge 55$, with the same constraint produced for $\hat{\boldsymbol{\xi}}_2$ as well. For $\hat{\boldsymbol{\xi}}_4$ and $\hat{\boldsymbol{\xi}}_5$ we find $t_{3,4} = t_{3,5} = 3$, giving the constraint $\xi^3 \le 50$. We can now express all five partitions:

$$\Xi\left(\hat{\boldsymbol{\xi}}_1\right) = \left\{\xi^1 = 1, 21 \le \xi^2 \le 55, 13 \le \xi^3 \le 50, 4 \le \xi^4 \le 100\right\},$$
$$\Xi\left(\hat{\boldsymbol{\xi}}_2\right) = \left\{\xi^1 = 1, 21 \le \xi^2 \le 55, 50 \le \xi^3 \le 87, 4 \le \xi^4 \le 100\right\},$$
$$\Xi\left(\hat{\boldsymbol{\xi}}_3\right) = \left\{\xi^1 = 1, 55 \le \xi^2 \le 89, 13 \le \xi^3 \le 50, 4 \le \xi^4 \le 100\right\},$$
$$\Xi\left(\hat{\boldsymbol{\xi}}_4\right) = \left\{\xi^1 = 1, 55 \le \xi^2 \le 89, 50 \le \xi^3 \le 87, 4 \le \xi^4 \le 52\right\},$$
$$\Xi\left(\hat{\boldsymbol{\xi}}_5\right) = \left\{\xi^1 = 1, 55 \le \xi^2 \le 89, 50 \le \xi^3 \le 87, 52 \le \xi^4 \le 100\right\}.$$

The final consideration is non-anticipativity constraints. We must enforce the following constraints, all of which are relatively intuitive as all our constraints include only a single time stage: $x_1^2 = x_2^2$, $x_3^2 = x_4^2 = x_5^2$, $x_4^3 = x_5^3$, with similar constraints on $\mathbf{y}$. Naturally the decision x^1 is the same for all partitions as well.

Iteration 2 We improve the objective from $1830\frac{2}{3}$ to 1672 thanks to our partitioning scheme. The "contributions" to the objective from each partition are $1426\frac{1}{3}$, 1672, 1627, $1530\frac{2}{3}$, and 1672 respectively. Only two of these partitions are "active partitions", so we will further divide only these two. Here we consider partition 2 only for the sake of exposition. This partition has the active uncertain parameters $\hat{\boldsymbol{\xi}}_{2,1} = (1,21,87,4)$, $\hat{\boldsymbol{\xi}}_{2,2} = (1,21,50,4)$, $\hat{\boldsymbol{\xi}}_{2,3} = (1,55,87,4)$, and $\hat{\boldsymbol{\xi}}_{2,4} = (1,21,50,100)$. These produce the following partitions, all of which are sub-partitions of the initial partition:

$$\Xi\left(\hat{\boldsymbol{\xi}}_{2,1}\right) = \left\{\xi^1 = 1, 21 \le \xi^2 \le 38, 68.5 \le \xi^3 \le 87, 4 \le \xi^4 \le 100\right\},$$

$$\Xi\left(\hat{\boldsymbol{\xi}}_{2,2}\right) = \left\{\xi^1 = 1, 21 \le \xi^2 \le 38, 50 \le \xi^3 \le 68.5, 4 \le \xi^4 \le 52\right\},$$
$$\Xi\left(\hat{\boldsymbol{\xi}}_{2,3}\right) = \left\{\xi^1 = 1, 38 \le \xi^2 \le 55, 50 \le \xi^3 \le 87, 4 \le \xi^4 \le 100\right\},$$
$$\Xi\left(\hat{\boldsymbol{\xi}}_{2,4}\right) = \left\{\xi^1 = 1, 21 \le \xi^2 \le 38, 50 \le \xi^3 \le 68.5, 52 \le \xi^4 \le 100\right\},$$

For nonanticipativity, we must first enforce the constraints of the first iteration. As we are not further partitioning partition 1 at this time, this translates to $x^2_1 = x^2_{2,1} = x^2_{2,2} = x^2_{2,3} = x^2_{2,4}$. We must also add new non-anticipativity constraints, namely $x^2_{2,1} = x^2_{2,2} = x^2_{2,4}$ and $x^3_{2,2} = x^3_{2,4}$

Method of Postek and Den Hertog (2014)

Iteration 1 The first step is to identify the t for which we will be using in the split. As described in §5.1 of Postek and Den Hertog (2014), we aim to find the component of the uncertain parameters that has maximal "dispersion" for a t such that $t_{max} \le t \le t_{max} + q$. The initial t_{max} for the set is 0, and the authors used $q = 2$ in their computational experiments. We are left with just $t = 2$ as as the uncertainty set is fixed for $t = 1$.

We now proceed to creating a "2-SH". Using "Heuristic 1" of §5.2 we select $\hat{\boldsymbol{\xi}}_2$ and $\hat{\boldsymbol{\xi}}_5$ as having maximal pairwise distance over the the selected time components, giving the hyperplane $\xi^2 = 55$. Each partition has $t_{max} = 2$, and no non-anticipativity constraints are required apart from the trivial one on x^1.

Iteration 2 The objective value remains at $1830\frac{2}{3}$, with both partitions individually also "contributing" this objective. We will focus on the partition that was defined by the constraint $\xi^2 \ge 55$.

This partition has the active uncertain parameters $\hat{\boldsymbol{\xi}}_{2,1} = (1, 55, 13, 4)$, $\hat{\boldsymbol{\xi}}_{2,2} = (1, 89, 13, 4)$, $\hat{\boldsymbol{\xi}}_{2,3} = (1, 89, 87, 4)$, and $\hat{\boldsymbol{\xi}}_{2,4} = (1, 89, 13, 100)$. We now search for the t to partition on, between $t_{max} = 2$ and $t_{max} + q = 4$. The maximum "dispersion" occurs for $t = 4$, so we will construct a "4-SH". The two two $\hat{\boldsymbol{\xi}}$ with maximum pairwise distance are $\hat{\boldsymbol{\xi}}_{2,3}$ and $\hat{\boldsymbol{\xi}}_{2,4}$, so the hyperplane we construct is $-74\xi^3 + 96\xi^4 = 1292$. We would thus need ξ^4 to distinguish which partition we are in, so we need nonanticipativity constraints $x^2_{2,3} = x^2_{2,4}$ and $x^3_{2,3} = x^3_{2,4}$.

Acknowledgments

We would like to thank the three anonymous reviewers for their helpful comments, Angelos Georghiou for his feedback and encouragement, and Miles Lubin and Boris Braverman for their thoughts on partitioning schemes.

References

Aurenhammer, Franz. 1991. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)* **23**(3) 345–405.

Ben-Tal, Aharon, Laurent El Ghaoui, Arkadi Nemirovski. 2009. *Robust optimization*. Princeton University Press.

Ben-Tal, Aharon, Boaz Golany, Arkadi Nemirovski, Jean-Philippe Vial. 2005. Retailer-supplier flexible commitments contracts: a robust optimization approach. *Manufacturing & Service Operations Management* **7**(3) 248–271.

Ben-Tal, Aharon, Alexander Goryashko, Elana Guslitzer, Arkadi Nemirovski. 2004. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming* **99**(2) 351–376.

Bertsimas, Dimitris, Hoda Bidkhori. 2014. On the performance of affine policies for two-stage adaptive optimization: a geometric perspective. *Mathematical Programming* 1–18.

Bertsimas, Dimitris, David B Brown, Constantine Caramanis. 2011a. Theory and applications of robust optimization. *SIAM review* **53**(3) 464–501.

Bertsimas, Dimitris, Constantine Caramanis. 2007. Adaptability via sampling. *Decision and Control, 2007 46th IEEE Conference on*. 4717–4722.

Bertsimas, Dimitris, Constantine Caramanis. 2010. Finite adaptability in multistage linear optimization. *Automatic Control, IEEE Transactions on* **55**(12) 2751–2766.

Bertsimas, Dimitris, Iain Dunning, Miles Lubin. 2015. Reformulation versus cutting-planes for robust optimization. *Computational Management Science* doi:10.1007/s10287-015-0236-z.

Bertsimas, Dimitris, Angelos Georghiou. 2014. Binary decision dules for multistage adaptive mixed-integer optimization. *Optimization Online* .

Bertsimas, Dimitris, Angelos Georghiou. 2015. Design of near optimal decision rules in multistage adaptive mixed-integer optimization. *Operations Research* .

Bertsimas, Dimitris, Vineet Goyal. 2010. On the power of robust solutions in two-stage stochastic and adaptive optimization problems. *Mathematics of Operations Research* **35**(2) 284–305.

Bertsimas, Dimitris, Vineet Goyal. 2012. On the power and limitations of affine policies in two-stage adaptive optimization. *Mathematical programming* **134**(2) 491–531.

Bertsimas, Dimitris, Dan Iancu, Pablo Parrilo. 2010. Optimality of affine policies in multistage robust optimization. *Mathematics of Operations Research* **35**(2) 363–394.

Bertsimas, Dimitris, Dan Iancu, Pablo Parrilo. 2011b. A hierarchy of near-optimal policies for multistage adaptive optimization. *Automatic Control, IEEE Transactions on* **56**(12) 2809–2824.

Bertsimas, Dimitris, Eugene Litvinov, Xu Andy Sun, Jinye Zhao, Tongxin Zheng. 2013. Adaptive robust optimization for the security constrained unit commitment problem. *Power Systems, IEEE Transactions on* **28**(1) 52–63.

Chen, Xin, Melvyn Sim, Peng Sun. 2007. A robust optimization perspective on stochastic programming. *Operations Research* **55**(6) 1058–1071.

Chen, Xin, Melvyn Sim, Peng Sun, Jiawei Zhang. 2008. A linear decision-based approximation approach to stochastic programming. *Operations Research* **56**(2) 344–357.

Dunning, Iain. 2015. JuMPeR: JuMP extension for robust optimization. URL `https://github.com/IainNZ/JuMPeR.jl`.

Fischetti, Matteo, Michele Monaci. 2012. Cutting plane versus compact formulations for uncertain (integer) linear programs. *Mathematical Programming Computation* **4**(3) 239–273.

Goulart, Paul J, Eric C Kerrigan, Jan M Maciejowski. 2006. Optimization over state feedback policies for robust control with constraints. *Automatica* **42**(4) 523–533.

Gurobi Optimization, Inc. 2015. Gurobi optimizer reference manual. URL `http://www.gurobi.com`.

Hadjiyiannis, M.J., P.J. Goulart, D. Kuhn. 2011. A scenario approach for estimating the suboptimality of linear decision rules in two-stage robust optimization. *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. 7386–7391.

Hanasusanto, Grani Adiwena, Daniel Kuhn, Wolfram Wiesemann. 2014. Two-stage robust integer programming. *Optimization Online* .

Lee, Der-Tsai, Robert L Drysdale, III. 1981. Generalization of voronoi diagrams in the plane. *SIAM Journal on Computing* **10**(1) 73–87.

Lubin, Miles, Iain Dunning. 2015. Computing in operations research using julia. *INFORMS Journal on Computing* **27**(2) 238–248.

Postek, Krzysztof, Dick Den Hertog. 2014. Multi-stage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *CentER Discussion Paper Series* .

Song, Yongjia, James Luedtke. 2014. An adaptive partition-based approach for solving two-stage stochastic programs with fixed recourse .

Vayanos, Phebe, Daniel Kuhn, Berç Rustem. 2011. Decision rules for information discovery in multi-stage stochastic programming. *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 7368–7373.

Zeng, Bo, Long Zhao. 2013. Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters* **41**(5) 457–461.