SCALING-BASED METHODS IN OPTIMIZATION AND CUT GENERATION

A Thesis
Presented to
The Academic Faculty

by

Jeffrey W. Pavelka

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology
May 2017

SCALING-BASED METHODS IN OPTIMIZATION AND CUT GENERATION

Approved by:

Professor Sebastian Pokutta, Advisor
H. Milton Stewart School of
Industrial and Systems Engineering
Georgia Institute of Technology

Professor Chelsea White, Co-Advisor
H. Milton Stewart School of
Industrial and Systems Engineering
Georgia Institute of Technology

Professor Alejandro Toriello
H. Milton Stewart School of
Industrial and Systems Engineering
Georgia Institute of Technology

Professor Santanu Dey
H. Milton Stewart School of
Industrial and Systems Engineering
Georgia Institute of Technology

Professor Marc E. Pfetsch
Department of Mathematics
Technische Universität Darmstadt

Date Approved: 16 February 2017

ACKNOWLEDGEMENTS

I'm lucky to have received so much love, support, guidance, and companionship on my way to completing this thesis.

I owe a great deal to my parents, Richard and Suzanne, who've shown me nothing but support, care, and affection for as long as I can remember. This debt also extends to my brother, Tony, who has had a profound influence on me throughout the years.

I'm grateful to have completed my doctoral studies under the advisement of Sebastian Pokutta, whose insights and ideas always pushed my research forward. It has been an honor to learn under such a well-respected researcher. I must also thank Chip White, Alejandro Toriello, Santanu Dey, and March Pfetsch for graciously serving on my dissertation committee. Along the way, I've benefited from the expertise of several collaborators; many thanks to Turgay Ayer, Murat Kurt, Pierre Le Bodic, and Gabor Braun for their contributions to my research pursuits.

In my time at Tech, I've taken many classes that helped build my analytical toolset and greatly influenced my thought processes and overall research directions. I would like to thank Ton Dieker, Craig Tovey, David Goldberg, George Nemhauser, Anton Kleywegt, Robin Thomas, and Santosh Vempala for delivering these stimulating courses. My academic journey did not begin at Georgia Tech, however - I'd be remiss not to offer a huge thank you to Todd Easton, my advisor while I secured a master's degree at Kansas State, and the person who first introduced me to optimization research.

During the last few years, I've had the pleasure of working with many wonderful people at Macy's Systems and Technology. Thanks to Ramesh Muthiah, Raghu

Krishnamurthy, Tim Kennell, Sue Higgins, Nimish Sevak, Praveen Sameneni, and many others who've made my collaboration with Macy's a pleasant one.

To the my fellow ISyE Ph.D. students, thank you for making these past few years so enjoyable. Along the way you've been classmates, friends, confidants, teammates, hosts, chauffeurs, and many other things. Thanks go out to James Bailey, Brian Kues, Lars Kirkby, Daniel Silva, Alborz Parcham Kashani, Tonya Roberts, Paul Vazquez, Rodrigue Ngueyep Tzoumpe, Camilo Ortiz, Erin Garcia, Ben Johnson, Anthony Bonifonte, Vinod Cheriyan, Daniel Zink, Jikai Zou, Jin Lee, Alvaro Lorca, Andres Iroume, Tim Sprock, Josh Hale, Ben Peters, Tony Yaacoub, Mathias Klapp, Matias Siebert, Aurko Roy, Amy Musselman, Seunghan Lee, Weijun Xie, Jan Vlachy, Mallory Soldner, Simon Chow, and so many others that I've regretfully omitted.

And Tonya, well... I think you're special enough to deserve another mention. Thank you for being part of my journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The work in this thesis aims to extend the body of knowledge on the topic of integer optimization, addressing both theoretical and practical concerns. In Chapter 2, we explore solving integer programs by use of an augmentation oracle - an oracle that, when given polytope $P \subseteq \mathbb{R}^n$, cost vector $c \in \mathbb{R}^n$, and feasible solution $\hat{x} \in P \cap \mathbb{Z}^n$, returns an improved solution $x \in P \cap \mathbb{Z}^n$ with $c^T x > c^T \hat{x}$ (or asserts no such x exists). By cleverly scaling the objective vector, one can make use of such an oracle to efficiently recover the optimal solution to an integer program. We study two known optimization techniques that use augmentation as a subroutine, called *bit scaling* and *geometric scaling*, focusing on the number of augmentations necessary to solve an integer program under these schemes. For geometric scaling we improve the best-known upper bound, which matches the performance of bit scaling for 0/1 polytopes. For bit scaling, we give a family of instances for which the algorithm exhibits behavior matching the known upper bound. Furthermore, this same example shows that bit scaling may require arbitrarily more augmentation steps that geometric scaling on the same problem. Lastly, we study the effectiveness of these augmentation approaches for solving integer programs in practice. Our results reveal that scaling methods can successfully help close the integrality gap on hard instances.

Chapter 3 addresses questions regarding Chvátal-Gomory (CG) cuts for 0/1 polytopes. A CG cut of a polyhedron P is any inequality of the form $c^T x \leq \lfloor \delta \rfloor$ with $c \in \mathbb{Z}^n$ and $c^T x \leq \delta$ valid for all $x \in P$. The *CG closure* of P, denoted P', is the intersection of P with all of its CG cuts. The work follows two streams of research. First, for any $0/1$ polytope P and integral vector c, it is known that iteratively obtaining the CG closure of P will yield the integer hull of P after k rounds, with $k \in O(n^2 \log n)$. Meanwhile, the best known lower bound is currently $\Omega(n^2)$. The first motivating question is whether we can close this gap. In this document, we present an improved upper bound which, while still $O(n^2 \log n)$ in the general case, represents an improvement for certain classes of polyhedra. The second stream of work regards determining the complexity of the separation problem over mod-k cuts, a class of cuts related to CG cuts. We are able to prove NP-completeness for this problem, mirroring a similar recent result for CG cuts.

Lastly, in Chapter 4 we study an inventory problem inspired by a collaboration with a large online retailer. In this problem, we control inventory in a scenario where the replenishment schedule is unknown - instead, the times between replenishments are governed by some random process. We develop a basic stochastic model for this scenario, analyze optimal decisions under this model, and demonstrate its effectiveness via a simulation study. Further, through data provided by our collaborators, we are able to test the usefulness of the model in real-life scenarios. Of particular interest here is the use of data-driven prediction techniques to tune model parameters. We demonstrate that predictions culled from sophisticated machine learning techniques (e.g. neural network regression) can provide a boost in performance as compared to simpler, classical techniques (e.g. moving averages).

CHAPTER I

INTRODUCTION

Broadly speaking, an optimization problem seeks to determine the most efficient way to complete a task. One of the earliest successes in the field of optimization was the development of linear programming (LP), in which we seek to maximize a linear cost function subject to a set of linear constraints. (Mixed) integer programming (MIP or IP) extends linear programming by stipulating that some or all variables must take on integer values. This is a powerful extension, and a wide array of real-life problems can be modeled under this framework. This modeling power comes at a cost, however: while linear programming is famously solvable in polynomial time, integer programs are NP-hard in general and hence efficient algorithms cannot exist unless P=NP. Despite this negative theoretical result, MIP is still a powerful tool helping users in many industries make better decisions.

Optimization problems are at the core of this thesis, including both practical and theoretical concerns. The first two-thirds of this work are dedicated to the theory of integer programming, rooted in two areas of active research: cutting planes and primal methods for integer program. The remainder of the thesis covers novel work within the field of inventory control, an area rich in applied optimization techniques. Integer programming is still involved here, alongside topics in stochastic processes and machine learning.

1.1 *Solving integer programs*

As so much of this thesis is rooted in integer programming, we open this introductory chapter with a brief review of the basics of MIP. In particular, to motivate the research in the rest of the thesis, we will briefly overview how integer programs

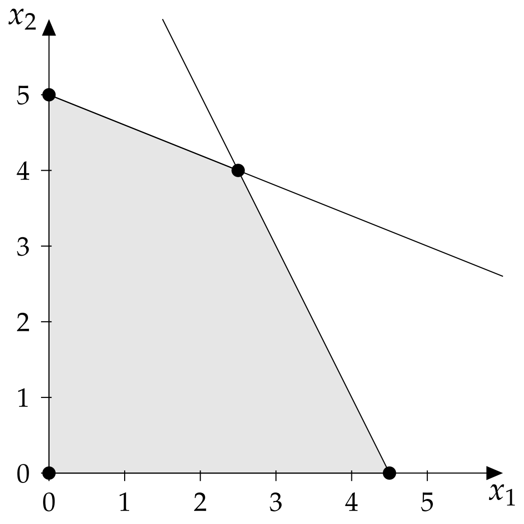


Figure 1: A polyhedron in $\mathbb{R}^2$ with extreme points highlighted. The polyhedron is shaded in gray, bounded by the constraints $x_1 \geq 0$, $x_2 \geq 0$, $2x_1 + 5x_2 \leq 25$, and $2x_1 + x_2 \leq 9$.

are solved in practice. To simplify the discussion, suppose the MIPs we work with are *pure* integer programs, i.e. all variables are forced to take integer values.

A *polyhedron* (pluralized as *polyhedra*) is any set of the form $\{x \in \mathbb{R}^n : Ax \leq b\}$ with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. A bounded polyhedron is also known as a *polytope*. The polyhedron is the basic geometric object of linear programming; an example polyhedron in $\mathbb{R}^2$ is illustrated in Figure 1. Given a *cost* or *objective* vector $c \in \mathbb{R}^n$, the goal of linear programming is to identify $x \in \operatorname{argmax}\{c^T x : x \in P\}$. An important result in linear programming is that if an optimal solutions exist, then there is some optimal solution which is an *extreme point* (also called a *vertex*) of the polyhedron; a point $x \in P$ which cannot be written as a convex combination of other elements of P.

Linear programming is known to be solvable in polynomial time, e.g. via the *ellipsoid method* [45] or *Karmarkar's algorithm* [43]. In practice, the most commonly-used and best-performing LP algorithm is still the *simplex method* [21], which interestingly has no known polynomial-time implementation. An important characteristic of the simplex method is that if an optimal solution exists, the method always return a solution which is an extreme point of the polyhedron.

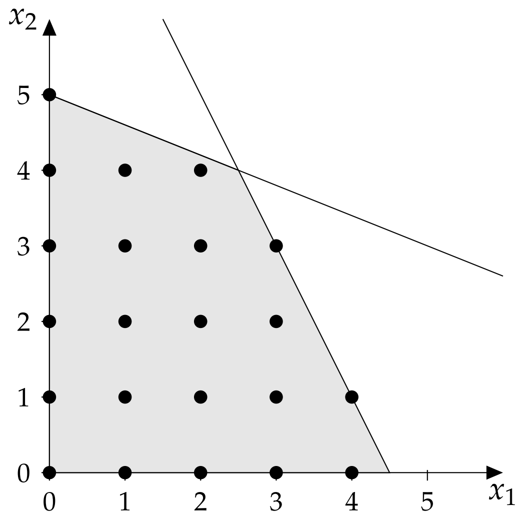


Figure 2: The integer feasible solutions in the polyhedron from Figure 1.

In contrast to linear programming, the set of feasible solutions in an integer program is of the form $P \cap \mathbb{Z}^n$ for some polyhedron P, as illustrated in Figure 2. Given cost vector $c \in \mathbb{R}^n$, the goal of integer programming is to identify $x \in \operatorname{argmax}\{c^T x : x \in P \cap \mathbb{Z}^n\}$. To find an optimal solution, IP solvers typically start with an inequality description $Ax \leq b$ of P, and proceed by first considering the problem's *linear programming relaxation*, i.e. the linear program that results from dropping all integrality restrictions. Clearly, the set of feasible solutions to the IP is a subset of the solutions to the LP relaxation. So if the solver finds an optimal LP solution x that also happens to be integral, then x must also be optimal for the IP.

How lucky must we be for the LP solution to actually be integral? Can we improve our luck? It should be clear from a glance at Figure 2 that there are many other polyhedra whose feasible integer points coincide exactly with those in the figure. Indeed, what if instead of using the polyhedron of Figure 2, we give the solver the polyhedron in Figure 3? Every extreme point of this polyhedron is integral, so if we use the simplex method to solve the LP relaxation, we are guaranteed to be given an integer solution.

In integer programming theory, the polyhedron in Figure 3 is known as the

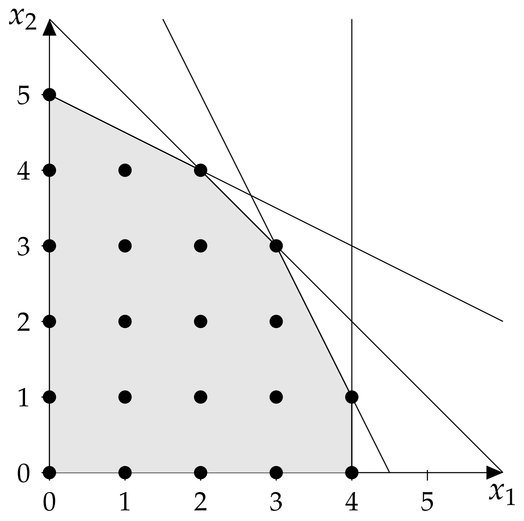


Figure 3: A polyhedron with the same integer feasible solutions as in Figure 2, but with the property that every extreme point is integral.

integer hull of the polyhedron in Figure 2. In general, for a polyhedron P, the integer hull of P is the set $P_I := \text{conv}\,(P \cap \mathbb{Z}^n)$, which is itself a polyhedron. Unfortunately, it is usually not straightforward to pass from an inequality description of P to a description of its integer hull. Consequently, searching for a description of P_I is not the chosen solution method for successful IP solvers.

When a solver optimizes the LP relaxation with nonintegral solution x', there are two common paths forward; branching and cutting planes. A *branching* technique creates from P and x' two or more sub-polyhedra $P_1, P_2, \ldots, P_k$ with the properties that $x' \notin \bigcup_{i=1}^{k} P_i$ and $P \cap \mathbb{Z}^n = \bigcup_{i=1}^{k}(P_i \cap \mathbb{Z}^n)$. Once the integer program corresponding to each sub-polyhedron is solved, we may compare their solutions to find the optimal solution for P. Of course, solving the IP over the sub-polyhedra may involve recursively branching many more times, and in general exponentially many sub-problems may be created.

Perhaps the most common branching strategy involves identifying a coordinate i for which $x'_i \notin \mathbb{Z}$, then creating two sub-polyhedra P_1 and P_2. The inequalities describing P_1 are the original inequalities $Ax \leq b$, plus the added inequality $x_i \leq \lfloor x'_i \rfloor$. Similarly, the inequalities describing P_2 are $Ax \leq b$ and $x_i \geq \lceil x'_i \rceil$. See Figure 4

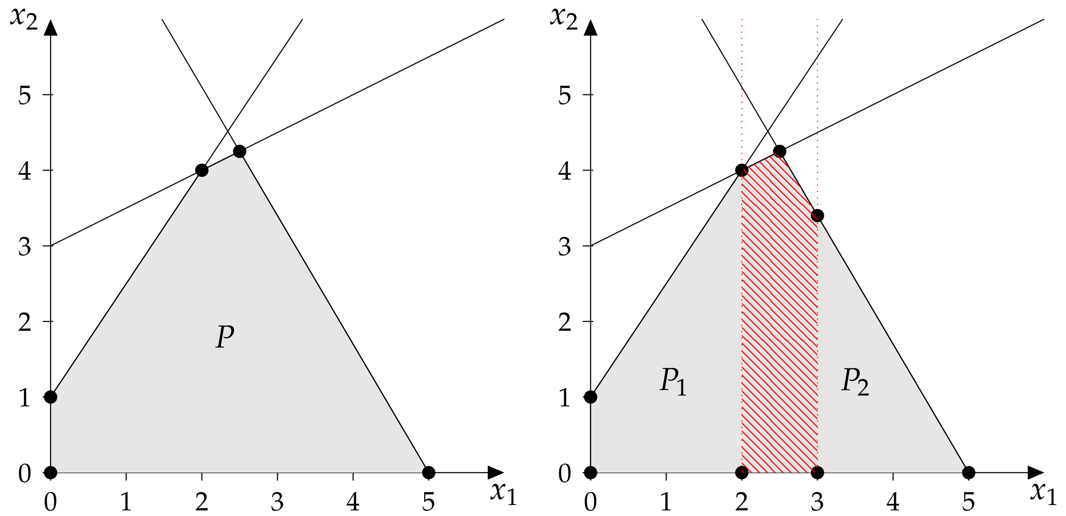


Figure 4: An example branching step. An IP solver trying to maximize x_2 over the polyhedron P will first solve the LP relaxation problem, finding solution (2.5, 4.25). This solution is not integral, so we branch on one of the non-integral coordinates - suppose we choose x_1. We create sub-polyhedron P_1 by adding the inequality $x_1 \leq 2$, and for sub-polyhedron P_2 we add $x_1 \geq 3$. The hatched region in the right-most image is not a part of $P_1 \cup P_2$, which is fine as it contains no integral solutions.

for an illustration.

The goal of a *cutting plane method* is to create from P a new polyhedron Q that more closely represents P_I (i.e. $P_I \subseteq Q \subsetneq P$). An IP solver employing a cutting plane technique takes the inequality description $Ax \leq b$ and non-integral LP relaxation solution x' to derive $a_0 \in \mathbb{R}^n$, $b_0 \in \mathbb{R}$ satisfying that the inequality $a_0^T x \leq b_0$ is valid for all $x \in P_I$, but $a_0^T x' > b_0$. The inequality $a_0^T x \leq b_0$ is called a *cutting plane* or simply a *cut*. The most desirable cuts are the *facet-defining* cuts, which are in some sense the strongest cuts available. For a cut $a_0^T x \leq b_0$ to be facet defining, the dimension of $P_I \cap \{x \in \mathbb{R}^n : a_0^T x = b_0\}$ must be one less than the dimension of P_I. Importantly, P_I is given exactly by the intersection of all of its facet-defining inequalities. See Figure 5 for an illustration of cutting planes.

The method then proceeds with the polyhedron defined by $Ax \leq b$ and $a_0^T x \leq b_0$, repeating until the associated LP relaxation returns an integral solution. Though it may not be immediately obvious, several cutting plane schemes (see e.g. [35] or [4]) have been devised that are guaranteed to terminate at an optimal solution

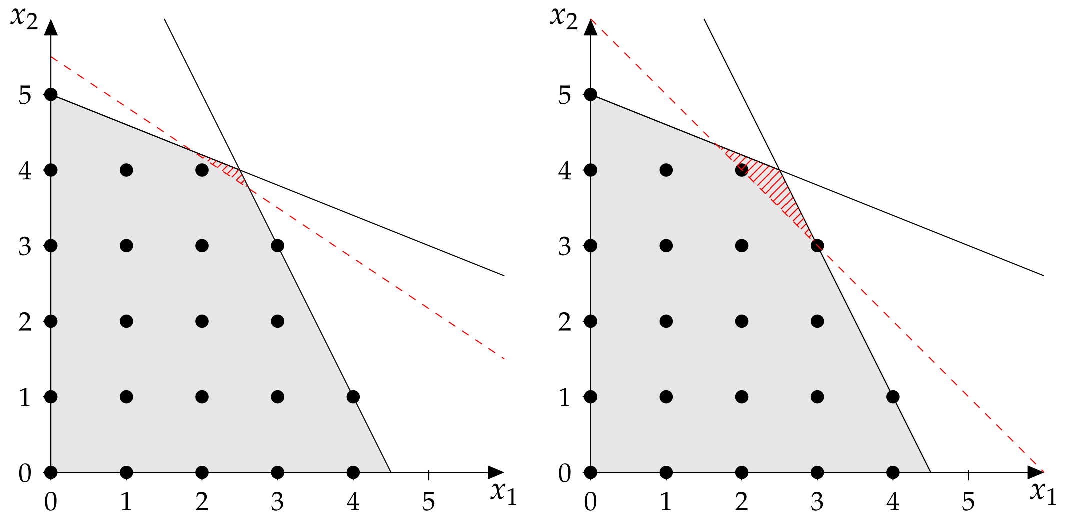


Figure 5: Cutting planes for the polyhedron from Figure 1. The dashed line in the left image illustrates a cutting plane $4x_1 + 6x_2 \leq 33$ that penetrates the polyhedron P, but leaves all integral solutions on the feasible side. This cut is not as tight as the facet-defining cut $x_1 + x_2 \leq 6$ on the right. The line defining this inequality intersects the 2-dimensional figure P_I on the 1-dimensional line segment between the points (2, 4) and (3, 3).

(though perhaps after adding exponentially many cuts), and may indeed be able to recover a description of P_I itself.

Of course, an IP solver need not be either purely branching-based or purely cut-based. Indeed, today's most successful solvers employ a combination of both in a technique called *branch and cut*, where either cuts or branches are added at various times in the algorithm.

1.2 Outline of topics

We now present the topics covered in this thesis, with motivations for each section along with a preview of the results found within.

1.2.1 Solving MIPs via Scaling-based Augmentation

Chapter 2 adds to the literature of so-called *primal* or *augmenting* methods for solving integer programs. These methods differ from traditional IP solving techniques in a key aspect. While both branching and cutting plane methods first find infeasible solutions then iteratively refine the search space, augmenting methods only consider moving from one feasible solution to another. Such a move is only taken if the new solution is an improvement to the previous according to some objective. To recover a good algorithm, we change this objective intelligently throughout the method. Thus a key difference between primal and more traditional methods is that primal methods alter the objective function as the method progresses, where the traditional methods alter the feasible region.

The techniques we address here are known as the *bit scaling* and *geometric scaling* methods. The "scaling" part of the names comes from the fact that the successively updated objective functions can be seen as progressively scaled versions of the original cost function. Both of these methods are known to terminate after only polynomially many *augmentation steps*, i.e. moving from one feasible solution to an improved one. We make theoretical contributions to the analysis of each of these algorithms. In the case of geometric scaling, we are able to give improvements in the worst-case analysis of the algorithm. In the case of bit scaling, we outline a family of examples whose worst-case behavior meet the known upper bounds (up to constant factors) for the number of augmentations necessary in the algorithm. Furthermore, the example shows that bit scaling can perform arbitrarily worse than geometric scaling.

Also included are a set of computational results, where the algorithms are implemented using standard IP solvers to carry out the augmentation steps. Here, we find that the augmentation methods are successful in decreasing optimality gaps for a class of difficult test problems.

1.2.2 CG and mod-k Cuts in the 0/1 Cube

In Chapter 3 we discuss various theoretical aspects of popular cutting plane techniques when the polyhedron P is such that $P \subseteq [0,1]^n$. In particular, we study the famous *Chvátal-Gomory (CG) cuts*, as well as a related class of cuts called *mod-k cuts*.

A CG cut of a polyhedron P is any inequality of the form $c^T x \leq \lfloor \delta \rfloor$ with $c \in \mathbb{Z}^n$ and $c^T x \leq \delta$ valid for all $x \in P$. The *CG closure* of P, denoted P', is the intersection of P with all of its CG cuts. It is already known that iteratively obtaining the CG closure of $P \subseteq [0,1]^n$ will yield the integer hull of P after k rounds, with $k \in O(n^2 \log n)$. We are able to tighten this bound slightly, but importantly the new proof allows for improved analyses for certain classes of polyhedra. The bounds obtained for these classes were unachievable from previous results.

Also of interest to researchers is the hardness of separation for various classes of cuts. Recently, it has been shown that the problem of finding a CG cut that separates a given point x from a polytope $P \in [0,1]^n$ is an NP-complete problem. In this document, we are able to prove NP-completeness for separation over a related class of cuts, called *mod-k cuts*, for any $k \in \mathbb{Z}_+$.

1.2.3 Opportunistic Replenishments in Inventory Modeling

In Chapter 4, we turn away from integer programming theory and study an application of optimization to inventory control. We build a model for the scenario where replenishment opportunities are not known in advance, and instead must be acted upon opportunistically as they arise.

As a first step, we devise and analyze a simple stochastic model for use in this scenario, and study the cost of uncertainty in its parameters. We then extend the model, making it more applicable for use in real-world processes. We discuss how to make replenishment decisions using the model, in both single-SKU and constrained multi-SKU settings.

Lastly, we test the applicability of the model via a simulation study. We simulate with synthetically generated data, as well as using real-world data provided to us by a large retailer. After establishing the usefulness of the model in the case that accurate order predictions are available, we turn to the question of determining order volumes from historical data. We find that using sophisticated machine learning techniques to drive the model can be more successful than relying on classical forecasting techniques.

CHAPTER II

SOLVING MIPS VIA SCALING-BASED AUGMENTATION

2.1 *Introduction*

Mixed integer programs (MIPs) are most often solved via a combination of branching and cutting plane techniques. An alternative path to solving MIPs is via *primal augmentation* approaches. The idea here is to start from a feasible solution, then iteratively move to new solutions with improved objective values by means of numerous *augmentation steps*.

The augmentation methods considered in this chapter are termed as *scaling* methods. The name comes from the fact that the objective function we improve against is progressively scaled throughout the algorithm. A key insight is that via appropriate scaling, only a polynomial number of augmentation steps is needed before the optimal solution is found. Hence, if these augmentation steps could be performed efficiently, one obtains an efficient algorithm for solving MIPs. For example, this augmentation can be performed very fast for network flows, which, in fact, motivated several scaling approaches for MIPs in the first place (see e.g., [70], [57]).

This work provides insights to the theory of scaling-based augmentation, as well as an exploration into the computational feasibility and performance of such methods.

2.1.1 Related work

Primal augmentation approaches in the context of MIPs have been well-studied, both from an algebraic point of view using test sets, but also in the context of solving linear programs and mixed-integer (nonlinear) programs exactly and approximately. Test sets (or Graver bases), i.e. the sets of feasible integer directions, are studied in [37], which give rise to a natural converging augmentation algorithm; see also [65]. Algebraic approaches (see e.g [24, 26] and the references contained therein) are usually based on an algebraic characterization of test sets; then an improving direction is used for augmentation.

Augmentation methods have recently become important in mixed-integer nonlinear problems (MINLPs), see, e.g., [40] and [56] for an overview. Here, test sets are used to solve or approximate MINLPs; some selected references are [25, 41, 42, 49, 50].

In [10, 11], among other approaches, an exponential penalty function framework is considered for (approximately) solving linear programs. Interestingly, this approach can be considered somewhat dual to the approximate LP solving framework via multiplicative weight updates in [59] for fractional packing and covering problems (see also [2] and [33] for similar applications). In [51], the authors consider an integrated augment-and-branch-and-cut framework for mixed $0/1$ programs. A proximity search heuristic is considered in [31], where the objective function is replaced by a proximity function to explore the neighborhood around a feasible solution.

The work of this chapter focuses on two classic scaling algorithms. First is the *bit scaling* method, which is applicable to solving MIPs over polyhedra P with $P \subseteq [0, 1]^n$. The method was introduced in [68] and based on [27]. Second is *geometric scaling*, which is valid for any P. This method was introduced in [67], which in turn is inspired by classical scaling algorithms for flow problems and certain linear programs (see e.g., [70], [57], [53]).

On a high level, the augmentation methods considered here are similar to proximal methods for nonlinear programs (see, e.g., [62]) in the sense that the deviation from the current iterate is penalized in the objective function; this is also the viewpoint of [31], mentioned above. On the other hand, local branching, see [30], would be the analogue of trust region methods, see, e.g., [18].

2.1.2 Contributions

The contributions of this work fall into two categories, theoretical and computational. On the theoretical side, we provide new bounds on the number of augmentation steps required for both bit scaling and geometric scaling. In the case of geometric scaling, Theorem 2.2.10 gives a new upper bound on the number of augmentations required, improving the best-known bound of [67] by a $\log n$ factor. Crucially, this result brings the bound in line with the best-known upper bound for bit scaling when $P \subseteq [0,1]^n$.

In the case of bit scaling, in Section 2.2.3 we provide a family of examples for which the worst-case number of augmentations meets the known upper bound, up to constant factors. Moreover, the same example shows that bit scaling can perform arbitrarily worse than geometric scaling on the same problem. Furthermore, a simple improvement for both scaling methods is derived in the case that a certain "width" of the polytope is small.

On the computational side, we discuss how to implement various MIP solving techniques based on both bit scaling and geometric scaling. We also present results of computational tests using these implementations. While the augmentation methods are not competitive with a commercial solver on easy instances, we do find them useful in finding high-quality incumbent solution for very hard instances.

2.1.3 Outline

All theoretical contributions are found in Section 2.2. Section 2.2.2 introduces the bit scaling algorithm, and present the classical analysis. In Section 2.2.3 we present the worst-case example for bit scaling, showing that the classical upper bound on augmentations is tight. In Section 2.2.4 we define the geometric scaling algorithm and prove a new upper bound on necessary augmentations. This is followed by a discussion in Section 2.2.5 on the relative merits of the two algorithms when $P \subseteq [0,1]^n$.

Discussions of the computational tests are found in Section 2.3. After highlighting some particulars of the implementation in Section 2.3.1, we present and discuss numerical results in Section 2.3.2.

2.2 *Augmentation bounds for scaling methods*

2.2.1 Definitions and notation

We now pause to set some definitions to be used throughout the chapter. Given $x, y \in \mathbb{R}^n$, we write simply xy to denote the inner product of x and y. We will often work with *directions* $x - y$ induced by two vectors $x, y \in \mathbb{R}^n$. If $P \subseteq \mathbb{R}^n$ is a polyhedron, we say that $z \in \mathbb{R}^n$ is a *feasible direction* for $x \in P$ if $x + z \in P$. Moreover, z is an *augmenting direction* if $cz > 0$, and it is an *integer feasible direction* if $z \in \mathbb{Z}^n$.

In a primal algorithm, we call the step of passing from one feasible solution to another an *augmentation* step. Throughout the course of a scaling algorithm, the objective we improve against updates many times. The objective is controlled by a *scaling parameter* μ. We call the time during which μ holds constant a *(scaling) phase*.

We denote by $\mathbf{1}$ the all-ones vector and by $\mathbf{0}$ the all-zeros vector; the dimensions of these vectors will be apparent from context. We will also write $[n] := \{1, \ldots, n\}$ for $n \in \mathbb{Z}_+$. For a vector $x \in \mathbb{R}^n$, let $\operatorname{supp}(x) := \{j \in [n] : x_j \neq 0\}$ be the *support* of x. For $x \in \mathbb{R}^n$ we use $\|x\|_\infty = \max_{[n]}\{|x_i|\}$ to denote the L^∞ norm of x. All logarithms in this chapter will be to the basis 2. All other notation is standard and can be found in [66] and [55], for example.

2.2.2 Bit scaling

We first present the *bit scaling* technique for solving 0/1 integer programs (see [68]; also [27, 36]). Letting P be a polyhedron and $c \in \mathbb{Z}^n$, we want to solve $\max\{cx : x \in P \cap \{0,1\}^n\}$. For the sake of exposition, and without loss of generality, we confine ourselves to $c \geq 0$ by applying suitable coordinate flips $x_i \mapsto 1 - x_i$. The classical bit scaling algorithm is given in Algorithm 1.

Algorithm 1 Bit scaling

```
Input:  Polyhedron $P \subseteq [0,1]^n$, integral $c \in \mathbb{Z}^n$, feasible solution $x^0 \in P \cap \mathbb{Z}^n$
Output:  Optimal solution of $\max\{cx : x \in P \cap \mathbb{Z}^n\}$
  $\mu \leftarrow 2^{\lceil \log \|c\|_\infty \rceil}$, $\tilde{x} \leftarrow x^0$, $k \leftarrow 1$
  repeat
      $c^k \leftarrow \lfloor c/\mu \rfloor$
      compute $x \in P$ integral with $c^k(x - \tilde{x}) > 0$          ▷ improve w.r.t. $c^k$
      if there is no feasible solution then
          $\mu \leftarrow \mu/2$, $k \leftarrow k + 1$
      else
          $\tilde{x} \leftarrow x$                                      ▷ update solution and repeat
      end if
  until $\mu < 1$
  return $\tilde{x}$                                                   ▷ return optimal solution
```

The algorithm assumes knowledge of an initial feasible point x^0. In practice, given only a description of P, it may not be straightforward to determine $x^0 \in P$. However, for many applications finding some feasible solution is not difficult. In this algorithm, k is essentially just a counter for the scaling phases. Within each phase k, we set an objective c^k to augment over. In order to carry out an augmenting step, we need a method that can correctly determine if an improving solution exists, and actually return such a solution if it does. For the sake of analyzing the algorithm, we do not comment on any actual implementation of this augmenting step. It is important to note that the analysis from [68] (which we reproduce in

Lemma 2.2.1) shows that the algorithm requires only polynomially (with respect to n and the size of c) many augmentations before returning the optimal solution to $\max\{cx : x \in P \cap \mathbb{Z}^n\}$. Hence an efficient algorithm for the augmentation problem would also recover an efficient algorithm for the optimization problem, implying that augmentation is NP-hard.

2.2.2.1 An illustrative example

Before proceeding to analyze the algorithm, it will be instructive to give a small example of bit scaling in action. This example will reveal the main mechanics of the upper bound proof of Lemma 2.2.1, as well as motivating the form of the construction in Section 2.2.3.

Suppose we have $P \subseteq \mathbb{R}^4$ (a description of P will not be important for this example), and choose $c = (7, 4, 5, 8)$. We have $\lceil \log \|c\|_\infty \rceil = 3$, so we initialize $\mu \leftarrow 8$. One easily verifies that the vectors c^k for each scaling phase are $c^1 = (0, 0, 0, 1)$, $c^2 = (1, 1, 1, 2)$, $c^3 = (3, 2, 2, 4)$, and $c^4 = (7, 4, 5, 8)$.

For analysis of the algorithm, it is perhaps more helpful to view the construction of the vectors c^k as illustrated in Figure 6. Here, the d^k vectors are a binary decomposition of c, in that $c = 2^3 d^1 + 2^2 d^2 + 2^1 d^3 + 2^0 d^4$. Moreover, the vectors c^k can each be obtained by $2c^{k-1} + d^k$. Importantly, this is not a quirk of the particular c selected for this example. The c^k vectors can always be obtained in such a manner.

How many augmenting steps must the algorithm take in this example? For the sake of exposition, let's define $c^0 = d^0 = (0, 0, 0, 0)$. Suppose we are in the kth scaling phase, making augmentations with respect to objective c^k. Let $x^k \in P \cap \mathbb{Z}^n$ be an optimal solution with respect to c^k, and x^{k-1} optimal with respect to c^{k-1}. The kth scaling phase begins with $\tilde{x} = x^{k-1}$. For this phase, the objective function

k	d^k				c^k			
1	0	0	0	1	0	0	0	1
2	1	1	1	0	1	1	1	2
3	1	0	0	0	3	2	2	4
4	1	0	1	0	7	4	5	8

Figure 6: A binary decomposition of c. Note that for $k > 1$, the c^k vector can be obtained by $2c^{k-1} + d^k$

difference between x^{k-1} and x^k is given by

$$c^k x^k - c^k x^{k-1} = \underbrace{2c^{k-1}x^k - 2c^{k-1}x^{k-1}}_{\leq 0 \text{ by optimality of } x^{k-1}} + d^k x^k - d^k x^{k-1} \leq 4.$$

So at most $4 = n$ augmenting steps are necessary in each phase.

2.2.2.2 Classic analysis

From the above example, the path to proving that Algorithm 1 requires $O(n \log \|c\|_\infty)$ augmentation steps is clear. We include such a proof here for completeness.

Lemma 2.2.1. *Let $P \subseteq [0,1]^n$ be a polytope, and let $c \in \mathbb{Z}^n_+$. Then Algorithm 1 solves the optimization problem* $\max\{cx : x \in P \cap \mathbb{Z}^n\}$ *with at most $n \cdot (\lceil \log \|c\|_\infty \rceil + 1)$ augmenting steps.*

Proof. For each scaling phase k, let $X^k = \text{argmax}\{c^k x : x \in P \cap \mathbb{Z}^n\}$, and for technical purposes set $X^0 = P \cap \mathbb{Z}^n$. The algorithm applies at most $1 + \lceil \log \|c\|_\infty \rceil$ scaling phases. We will show that at most n augmentations are necessary within a phase, implying the desired $n \cdot (\lceil \log \|c\|_\infty \rceil + 1)$ bound.

Set some phase k. The phase begins with $\tilde{x} = x^{k-1} \in X^{k-1}$, and will end $\tilde{x} = x^k \in X^k$. Since c^k is integral, each augmenting step improves on the objective by at least one. Thus to bound the number of augmentations within a phase, it suffices to bound $c^k x^k - c^k x^{k-1}$.

Comparing $c^k = \lfloor c/\mu \rfloor$ to $c^{k-1} = \lfloor c/2\mu \rfloor$, we see that we may write $c^k = 2c^{k-1} + d^k$ for some $d^k \in \{0,1\}^n$. Using this decomposition, we write

$$c^k x^k - c^k x^{k-1} = 2c^{k-1}x^k - 2c^{k-1}x^{k-1} + d^k x^k - d^k x^{k-1}.$$

The optimality of x^{k-1} with respect to c^{k-1} implies that $2c^{k-1}x^{k-1} \geq 2c^{k-1}x^k$, i.e. $2c^{k-1}x^k - 2c^{k-1}x^{k-1} \leq 0$. Further, as $d^k, x^k, x^{k-1} \in \{0,1\}^n$ we known $d^k x^k \leq n$ and $d^k x^{k-1} \geq 0$. Hence we obtain

$$c^k x^k - c^k x^{k-1} \leq n$$

as an upper bound on the number of augmentations necessary within a scaling phase. $\square$

2.2.3 A worst-case example for bit scaling

In this section, we present one of the main findings of the chapter. In particular, we show that the upper bound in Lemma 2.2.1 is tight. For this we provide a family of polytopes $P_n \subseteq [0,1]^n$ and cost functions c^p so that the bit scaling method needs $\Omega(n \log \|c^p\|_\infty)$ augmentation steps in the worst case. Each instance of this family is parametrized by two numbers, namely $k \in \mathbb{Z}_+$, which dictates the dimension $n := 8k-2$ of the cube $[0,1]^n$, and $p \in \mathbb{Z}_+$, which controls how the objective function c^p is built, and, by construction, the number p of bit scaling phases that will be required to solve the instance.

Furthermore, this example will highlight the potential performance differences between bit scaling and geometric scaling. See Section 2.2.5 for details.

2.2.3.1 A concrete example

Before outlining how to construct these examples in general, we will start by constructing a single concrete instance. The aim is to clearly illustrate the components of the construction which allow these examples meet the $\Omega(n \log \|c^p\|_\infty)$ bound.

For this instance we will choose $k = 3$, thus the polytope P lies in $8 \cdot 3 - 2 = 22$-dimensional space. The P is constructed as the convex hull of $2 \cdot k = 6$ points, which we name $y^1, ..., y^6$. We mentally break these six points into two distinct groups, with y^1, y^2, and y^3 making up group 1, and y^4, y^5, and y^6 making group 2.

The construction of the y points and cost vector will be so that, in each scaling phase k, the vector c^k we optimize over will have

$$c^k y^1 = c^k y^2 + 1 = c^k y^3 + 2 \qquad \text{and} \qquad c^k y^4 = c^k y^5 + 1 = c^k y^6 + 2. \tag{1}$$

Thus each scaling phase will end at either y^1 or y^4 as the best solution. Additionally, scaling phases will alternate between having either all of group 1 with a higher objectives value than all of group 2's, or vice versa. Then the algorithm, which only requires finding an improving solution at each iteration, may need to visit all points within a particular group in each phase.

The particulars of the construction of the y points and possible cost vectors $c^p, p \in \mathbb{Z}_+$ are outlined in Figure 7. Each column represents one coordinate of the 22-dimensional space. Further, columns are visually broken into groups dependent on their role in the construction, which will be explained shortly. Each row is dedicated to a particular vector in $\mathbb{Z}^{22}$, with the value in each column denoting that vector's value in the corresponding coordinate.

Mirroring what we've seen in Section 2.2.2.1, the cost vectors c^p are defined recursively according to the following scheme: Set $c^1 = d^1$, then for $p > 1$, set $c^p = 2c^{p-1} + d^{\text{even}}$ if p is even, else $c^p = 2c^{p-1} + d^{\text{odd}}$ if p is odd. Examining the mechanics of Algorithm 1, one can verify that the bit scaling algorithm, when given cost vector c^p, spends the first scaling phase optimizing over c^1, the second phase optimizing over c^2, and so on until optimizing over c^p in the pth and final phase.

Now we examine the behavior of the bit scaling algorithm when given polytope $P = \text{conv}\left(y^1, ..., y^6\right)$ and cost vector c^p for some p. Figure 8 illustrates the costs of each point with respect to the first few c^p vectors. For c^1, we see that (1) is achieved

Role	Establish In-Group Order		Maintain In-Group Order		Group 1 Overtakes Group 2									Group 2 Overtakes Group 1								
Coordinate	1	2	1	2	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
y^1	1	1	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
y^2	1	0	1	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
y^3	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
y^4	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
y^5	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
y^6	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
d^1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
d^{even}	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
d^{odd}	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
c^1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c^2	2	2	1	1	2	2	2	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
c^3	4	4	3	3	5	5	5	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Figure 7: Vectors relevant to the bit scaling worst-case instance. Each row is dedicated to a particular vector in $\mathbb{Z}^{22}$, with the value in each column denoting that vector's value in the corresponding coordinate.

	y^1	y^2	y^3	y^4	y^5	y^6
c^1	5	4	3	2	1	0
c^2	10	9	8	13	12	11
c^3	29	28	27	26	25	24

Figure 8: Bit scaling worst-case instance: Costs of the y points over the first three objective vectors.

due to the coordinates in the first collection of columns in Figure 7. Moreover, the first three coordinates in the third collection of columns ensure that all points in group 1 have a higher objective value than any point in group 2. At the end of this scaling phase, the algorithm will have current solution $\tilde{x} = y^1$.

Moving on to the next scaling phase, the algorithm now must optimize over cost vector $c^2 = 2c^1 + d^{\text{even}}$. The doubling of c^1 for this phase breaks condition (1). However, the second collection of columns in Figure 7 correct for this, maintaining

(1) for the second scaling phase. Additionally, the coordinates in the fourth collection of columns add just enough to the cost of group 2's points to switch the order of the groups - now any point in group 2 has a higher objective value than any point in group 1. Thus the second scaling phase, which began at group 1 point y^1, may need to move through all of group 2's point before reaching y^4 and moving to the next phase.

The same process repeats in subsequent scaling phases, with the algorithm beginning each phase at either y^1 or y^4, then (in the worst case) spending one iteration traveling to each point in the other group. Thus worst-case behavior necessitates $3p \approx 3 \log \|c^p\|_\infty$ augmentations before termination. In the general case this will translate to $\frac{n}{8} \log \|c^p\|_\infty$, giving the desired result.

2.2.3.2 General construction - polytope

We now proceed to give the full, general construction. The polytope $P_n \subseteq [0,1]^n$ will be of the form

$$P_n = \operatorname{conv}\left(\left\{y^1, \dots, y^{2k}\right\}\right),$$

where we break the vectors $y^j \in \{0,1\}^n$ into four distinct components, $y^{j,1} \in \{0,1\}^{k-1}, y^{j,2} \in \{0,1\}^{k-1}, y^{j,3} \in \{0,1\}^{3k}$, and $y^{j,4} \in \{0,1\}^{3k}$. These four components correspond directly to the four collections of columns portrayed in Figure 7, and play the same roles in the general construction. With these four families of vectors defined, the full vector y^j is given by

$$y^j := \begin{pmatrix} y^{j,1} \\ y^{j,2} \\ y^{j,3} \\ y^{j,4} \end{pmatrix} \qquad \text{or equivalently} \qquad y^j_i := \begin{cases} y^{j,1}_i & \text{for } i \in \{1, \dots, k-1\}, \\ y^{j,2}_{i-k+1} & \text{for } i \in \{k, \dots, 2k-2\}, \\ y^{j,3}_{i-2k+2} & \text{for } i \in \{2k-1, \dots, 5k-2\}, \\ y^{j,4}_{i-5k+2} & \text{for } i \in \{5k-1, \dots, 8k-2\}. \end{cases}$$

The parts $y^{j,1}, y^{j,2}$ are defined in two batches. For the first batch with $j \in \{1, \ldots, k\}$, we define

$$y_i^{j,1} := \begin{cases} 1 & \text{if } i \geq j, \\ 0 & \text{otherwise,} \end{cases} \qquad y_i^{j,2} := \begin{cases} 1 & \text{if } i < j, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i = 1, \ldots, k-1.$$

For the second batch with $j \in \{k+1, \ldots, 2k\}$, we define

$$y_i^{j,1} := \begin{cases} 1 & \text{if } i \geq j-k, \\ 0 & \text{otherwise,} \end{cases} \qquad y_i^{j,2} := \begin{cases} 1 & \text{if } i < j-k, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{for } i = 1, \ldots, k-1.$$

We define $y^{j,3}, y^{j,4}$ with $j \in \{1, \ldots, 2k\}$ as follows

$$y_i^{j,3} := \begin{cases} 1 & \text{if } j \leq k, \\ 0 & \text{otherwise,} \end{cases} \qquad y_i^{j,4} := \begin{cases} 1 & \text{if } j > k, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{for } i = 1, \ldots, 3k.$$

See Figure 9 for an illustration.

2.2.3.3 General construction - cost vector

The cost vector is defined inductively, keeping the mechanics of the bit scaling procedure in mind. We first define $c^0 := \mathbf{0}$, and for $\ell = 1, \ldots, p$, we build $c^\ell = 2c^{\ell-1} + d^\ell$, for some vector $d^\ell \in \{0,1\}^n$ to be specified. We will find it convenient to construct $d^\ell = (d^{\ell,1}, d^{\ell,2}, d^{\ell,3}, d^{\ell,4})$ in terms of vectors $d^{\ell,1}, d^{\ell,2}, d^{\ell,3}$, and $d^{\ell,4}$ in the same manner as we did for the points y^j.

For $d^1 := c^1$, let

$$d^{1,1} := \mathbf{1}, \qquad d^{1,2} := 0, \qquad d_i^{1,3} := \begin{cases} 1 & \text{if } i \leq k, \\ 0 & \text{otherwise,} \end{cases} \quad \text{for } i = 1, \ldots, 3k, \qquad d^{1,4} := 0.$$

For $\ell \geq 2$, we set

$$d^{\ell,1} := 0, \qquad d^{\ell,2} := \mathbf{1}, \qquad d^{\ell,3} := \begin{cases} \mathbf{1} & \text{if } \ell \text{ is odd,} \\ 0 & \text{otherwise,} \end{cases} \qquad d^{\ell,4} := \begin{cases} \mathbf{1} & \text{if } \ell \text{ is even,} \\ 0 & \text{otherwise.} \end{cases}$$

In particular, after the first scaling phase, the contribution of the first $2(k-1)$ coordinates is the same for all y^j. In fact, we use the first $2(k-1)$ coordinates for the improvements steps within a scaling phase and the last $6k$ coordinates to switch between the phases; this will become clear soon. Note that for each $\ell > 1$, $\log \|c^\ell\|_\infty \in \Theta(\ell)$.

2.2.3.4 General construction - proving the lower bound

We will now derive a lower bound on the worst-case number of augmentations computed by the bit scaling algorithm when applied to a polytope P_n and cost vector c^p as defined in Section 2.2.3.2 and Section 2.2.3.3, respectively. We depict the overall structure of the construction in Figure 9, describing the points y^j and the "layers" d^ℓ of the cost function. Note how the columns in Figure 9 are divided into four segments. These four segments correspond to the four families of vectors used in defining y^j and d^ℓ. For example, the first group of columns in the y^j row depict the vector $y^{j,1}$, the second group of columns depict $y^{j,2}$, and so on.

The essence of the proof is the following: within a scaling phase, the algorithm may move to *any* solution with an improving cost with respect to vector $\lfloor c/\mu \rfloor$ (recall that μ is the scaling factor), no matter the magnitude of the improvement. In our construction, no matter the choice of k, p, the bit scaling algorithm begins by optimizing over the cost vector c^1. The construction is such that $c^1 y^1 > c^1 y^2 > \cdots > c^1 y^{2k}$. Thus if the algorithm begins at initial solution y^{2k}, it may visit *all* of the $2k$ points in P_n, ending the initial phase at y^1.

In the second scaling phase, the algorithm optimizes over c^2. We will see that we have $c^2 y^1 < c^2 y^{2k} < c^2 y^{2k-1} < \cdots < c^2 y^{k+1}$. Thus, in this phase, the algorithm may take k augmentation steps before finishing at point y^{k+1}. In the third augmentation phase, while optimizing over c^3, we similarly have $c^3 y^{k+1} < c^3 y^k < \cdots < c^3 y^1$, giving another possible k augmentations within the phase.

	$y^{j,1}$					$y^{j,2}$					$y^{j,3}$				$y^{j,4}$			
	1	2	3	$\cdots$	$k-1$	1	2	3	$\cdots$	$k-1$	1	2	$\cdots$	$3k$	1	2	$\cdots$	$3k$
y^1	1	1	1	$\cdots$	1	0	0	0	$\cdots$	0	1	1	$\cdots$	1	0	0	$\cdots$	0
y^2	0	1	1	$\cdots$	1	1	0	0	$\cdots$	0	1	1	$\cdots$	1	0	0	$\cdots$	0
y^3	0	0	1	$\cdots$	1	1	1	0	$\cdots$	0	1	1	$\cdots$	1	0	0	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
y^k	0	0	0	$\cdots$	0	1	1	1	$\cdots$	1	1	1	$\cdots$	1	0	0	$\cdots$	0
y^{k+1}	1	1	1	$\cdots$	1	0	0	0	$\cdots$	0	0	0	$\cdots$	0	1	1	$\cdots$	1
y^{k+2}	0	1	1	$\cdots$	1	1	0	0	$\cdots$	0	0	0	$\cdots$	0	1	1	$\cdots$	1
y^{k+3}	0	0	1	$\cdots$	1	1	1	0	$\cdots$	0	0	0	$\cdots$	0	1	1	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
y^{2k}	0	0	0	$\cdots$	0	1	1	1	$\cdots$	1	0	0	$\cdots$	0	1	1	$\cdots$	1
d^1	1	1	1	$\cdots$	1	0	0	0	$\cdots$	0	$d^{1,3}_1$	$d^{1,3}_2$	$\cdots$	$d^{1,3}_{3k}$	0	0	$\cdots$	0
d^2	0	0	0	$\cdots$	0	1	1	1	$\cdots$	1	0	0	$\cdots$	0	1	1	$\cdots$	1
d^3	0	0	0	$\cdots$	0	1	1	1	$\cdots$	1	1	1	$\cdots$	1	0	0	$\cdots$	0
d^4	0	0	0	$\cdots$	0	1	1	1	$\cdots$	1	0	0	$\cdots$	0	1	1	$\cdots$	1
d^5	0	0	0	$\cdots$	0	1	1	1	$\cdots$	1	1	1	$\cdots$	1	0	0	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Figure 9: Structure of y^j and d^ℓ; note that $d^{1,3}$ depends on k.

The process continues in each subsequent scaling phase, with the algorithm having the opportunity to travel through each of the points $y^{2k}, y^{2k-1}, \ldots, y^{k+1}$ in even phases, and $y^k, y^{k-1}, \ldots, y^1$ in odd phases, as depicted in Figure 10. Since $k \approx n/8$, this implies a worst case $\Omega(n)$ augmentations per scaling phase, meeting the upper bound from Lemma 2.2.1.

We now begin the formal proof. We will first show that in each phase ℓ, the first k points $y^1, \ldots, y^k$ are ordered in a decreasing fashion by the objective function c^ℓ and similar for the second k points $y^{k+1}, \ldots, y^{2k}$. In a second step we will then link the two groups.

Lemma 2.2.2 (Decreasing order within each group). *Let c^ℓ, y^j be constructed as above.*

For any $\ell \geq 1$ and $j \in \{1, \ldots, k-1\} \cup \{k+1, \ldots, 2k-1\}$, we have

$$c^\ell y^j = c^\ell y^{j+1} + 1.$$

Proof. The proof is by induction on ℓ, with base case $\ell = 1$. For $j \in \{1, \ldots, 2k\}$, define $\alpha_{1,j} := d^{1,3}y^{j,3}$. For $j \in \{1, \ldots, 2k\}$, we have

$$c^1 y^j = d^1 y^j = \underbrace{d^{1,1}y^{j,1}}_{=k-j} + \underbrace{d^{1,2}y^{j,2}}_{=0} + \underbrace{d^{1,3}y^{j,3}}_{=\alpha_{1,j}} + \underbrace{d^{1,4}y^{j,4}}_{=0} = k - j + \alpha_{1,j}.$$

By construction, we have $\alpha_{1,1} = \alpha_{1,2} = \cdots = \alpha_{1,k} = 1$ and $\alpha_{1,k+1} = \alpha_{1,k+2} = \cdots = \alpha_{1,2k} = 0$. Thus for $j \in \{1, \ldots, k-1\} \cup \{k+1, \ldots, 2k-1\}$, we can establish

$$c^1 y^j - c^1 y^{j+1} = k - j + \alpha_{1,j} - (k - (j+1) + \alpha_{1,j+1}) = 1,$$

as $\alpha_{1,j+1} = \alpha_{1,j}$.

Now assume $\ell \geq 2$. For $j \in \{1, \ldots, k\}$ we can verify that

$$\begin{aligned} c^\ell y^j &= 2c^{\ell-1}y^j + d^\ell y^j \\ &= 2c^{\ell-1}y^j + \underbrace{d^{\ell,1}y^{j,1}}_{=0,\text{ as } d^{\ell,1}=0} + \underbrace{d^{\ell,2}y^{j,2}}_{=j-1} + \underbrace{d^{\ell,3}y^{j,3}}_{=:\alpha_\ell} + \underbrace{d^{\ell,4}y^{j,4}}_{=0,\text{ as } y^{j,4}=0 \text{ for } j \leq k} \\ &= 2c^{\ell-1}y^j + (j-1) + \alpha_\ell, \end{aligned}$$

where $\alpha_\ell = 3k$ if ℓ is odd, and otherwise $\alpha_\ell = 0$. Thus, for $j \in \{1, \ldots, k-1\}$ we have

$$\begin{aligned} c^\ell y^j - c^\ell y^{j+1} &= 2c^{\ell-1}y^j + j - 1 + \alpha_\ell - (2c^{\ell-1}y^{j+1} + j + \alpha_\ell) \\ &= 2(\underbrace{c^{\ell-1}y^j - c^{\ell-1}y^{j+1}}_{=1,\text{ by induction}}) - 1 = 1. \end{aligned}$$

We can do a similar analysis for $\ell \geq 2$ and $j \in \{k+1, \ldots, 2k\}$:

$$\begin{aligned} c^\ell y^j &= 2c^{\ell-1}y^j + d^\ell y^j \\ &= 2c^{\ell-1}y^j + \underbrace{d^{\ell,1}y^{j,1}}_{=0,\text{ as } d^{\ell,1}=0} + \underbrace{d^{\ell,2}y^{j,2}}_{=j-1} + \underbrace{d^{\ell,3}y^{j,3}}_{=0,\text{ as } y^{j,3}=0 \text{ for } j \geq k+1} + \underbrace{d^{\ell,4}y^{j,4}}_{=:\beta_\ell} \\ &= 2c^{\ell-1}y^j + (j-1) + \beta_\ell, \end{aligned}$$

where $\beta_\ell = 3k$ if ℓ is even, and $\beta_\ell = 0$ otherwise . As before we obtain that for $j = k+1, \ldots, 2k-1$, $c^\ell y^j - c^\ell y^{j+1} = 1$ holds. □

Note that in the above argument the values of $d^{\ell,3}$, $d^{\ell,4}$ are irrelevant as they are eliminated in the difference of two consecutive points. However, they will become important as they enable the switching between and linking of the two groups $\{y^1, \ldots, y^k\}$ and $\{y^{k+1}, \ldots, y^{2k}\}$ as we will show now. To this end we prove the following lemma:

Lemma 2.2.3 (Decreasing intergroup ordering). *For any $\ell \geq 1$, if ℓ is odd then $c^\ell y^k = c^\ell y^{k+1} + 1$, and if ℓ is even then $c^\ell y^{2k} = c^\ell y^1 + 1$.*

Proof. The proof is by alternating induction on the odd and even case. First observe that $c^1 y^k = c^1 y^{k+1} + 1$, which will be the start of our induction for the odd case:

$$\begin{aligned} c^1 y^k - c^1 y^{k+1} &= \underbrace{d^{1,1}(y^{k,1} - y^{k+1,1})}_{=-(k-1)} + \underbrace{d^{1,2}(y^{k,2} - y^{k+1,2})}_{=0} \\ &\quad + \underbrace{d^{1,3}(y^{k,3} - y^{k+1,3})}_{=k} + \underbrace{d^{1,4}(y^{k,4} - y^{k+1,4})}_{=0} \\ &= 1. \end{aligned}$$

First, let $\ell \geq 1$ be even and suppose $c^{\ell-1} y^k = c^{\ell-1} y^{k+1} + 1$, which is satisfied in the case $\ell = 2$ by the above. Then, repeated application of Lemma 2.2.2 yields $c^{\ell-1} y^1 = c^{\ell-1} y^{2k} + 2k - 1$. Moreover, we have

$$\begin{aligned} c^\ell y^1 &= 2c^{\ell-1} y^1 + d^\ell y^1 \\ &= 2c^{\ell-1} y^1 + \underbrace{d^{\ell,1} y^{1,1}}_{=0, \text{ as } \ell > 1} + \underbrace{d^{\ell,2} y^{1,2}}_{=0, \text{ as } y^{1,2} = 0} + \underbrace{d^{\ell,3} y^{1,3}}_{=0, \text{ as } \ell \text{ even}} + \underbrace{d^{\ell,4} y^{1,4}}_{=0, \text{ as } y^{1,4} = 0} \\ &= 2c^{\ell-1} y^1, \end{aligned}$$

and

$$\begin{aligned}
c^{\ell} y^{2k} &= 2c^{\ell-1} y^{2k} + d^{\ell} y^{2k} \\
&= 2c^{\ell-1} y^{2k} + \underbrace{d^{\ell,1} y^{2k,1}}_{=0,\text{ as } \ell > 1} + \underbrace{d^{\ell,2} y^{2k,2}}_{=k-1} + \underbrace{d^{\ell,3} y^{2k,3}}_{=0,\text{ as } \ell \text{ even}} + \underbrace{d^{\ell,4} y^{2k,4}}_{=3k} \\
&= 2c^{\ell-1} y^{2k} + (k-1) + 3k = 2c^{\ell-1} y^{2k} + 4k - 1.
\end{aligned}$$

Thus, we obtain for the difference

$$\begin{aligned}
c^{\ell} y^{2k} - c^{\ell} y^{1} &= 2c^{\ell-1} y^{2k} + 4k - 1 - 2c^{\ell-1} y^{1} \\
&= 2(\underbrace{c^{\ell-1} y^{2k} - c^{\ell-1} y^{1}}_{=1-2k,\text{ from above}}) + 4k - 1 \\
&= 2(1-2k) + 4k - 1 \\
&= 1.
\end{aligned}$$

Now we consider the case where ℓ is odd, which is similar to the one above. Assume that $c^{\ell-1} y^{2k} = c^{\ell-1} y^{1} + 1$, which we now know to hold for $\ell = 3$ by means of the argument for ℓ even case from above. Then, applying Lemma 2.2.2 in increasing and decreasing direction, we obtain $c^{\ell-1} y^{k} + 2k - 1 = c^{\ell-1} y^{k+1}$. We will show that $c^{\ell} y^{k} = c^{\ell} y^{k+1} + 1$. We have

$$c^{\ell} y^{k} = 2c^{\ell-1} y^{k} + \underbrace{d^{\ell,1} y^{k,1}}_{=0} + \underbrace{d^{\ell,2} y^{k,2}}_{=k-1} + \underbrace{d^{\ell,3} y^{k,3}}_{=3k} + \underbrace{d^{\ell,4} y^{k,4}}_{=0} = 2c^{\ell-1} y^{k} + 4k - 1$$

and

$$c^{\ell} y^{k+1} = 2c^{\ell-1} y^{k+1} + \underbrace{d^{\ell,1} y^{k+1,1}}_{=0} + \underbrace{d^{\ell,2} y^{k+1,2}}_{=0} + \underbrace{d^{\ell,3} y^{k+1,3}}_{=0} + \underbrace{d^{\ell,4} y^{k+1,4}}_{=0} = 2c^{\ell-1} y^{k+1},$$

so that

$$c^{\ell} y^{k} - c^{\ell} y^{k+1} = 2(\underbrace{c^{\ell-1} y^{k} - c^{\ell-1} y^{k+1}}_{=1-2k}) + 4k - 1 = 1. \qquad \square$$

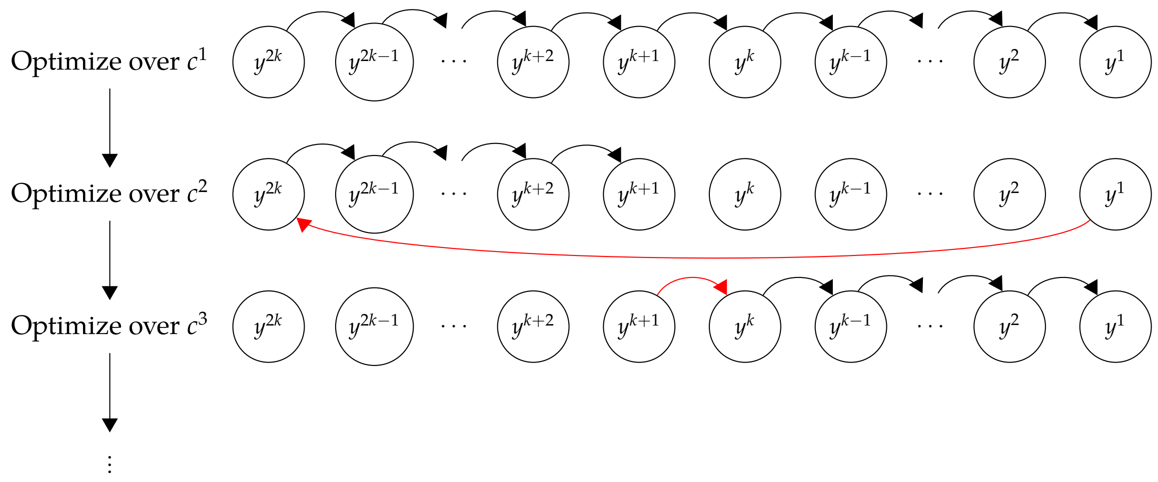


Figure 10: Points visited by the bit scaling algorithm in the worst case. Black arcs follow via Lemma 2.2.2, red arcs via Lemma 2.2.3.

With these last two lemmas in hand, we are ready to prove the worst-case lower bound. The proof describes the possible behavior of the bit scaling algorithm when given a polytope P_n and cost vector c^p, as depicted in Figure 10. The $\Omega(n \log \|c^p\|_\infty)$ lower bound proven here meets the upper bound established in Lemma 2.2.1, implying that the analysis is tight.

Theorem 2.2.4. *Choose $k \geq 1$ and set $n := 8k - 2$. Let $P_n = \operatorname{conv}\left(\{y^1, \dots, y^{2k}\}\right)$ be the polytope and c^p for some $p \geq 1$ the objective function as constructed above. Then the bit scaling algorithm optimizing c^p over P_n requires $\Omega(n \log \|c^p\|_\infty)$ augmentation steps in the worst case.*

Proof. By construction of c^p, the bit scaling algorithm optimizes over $c^1, c^2, \dots, c^p$ in successive scaling phases. The algorithm begins by optimizing over c^1. Using the results of Lemma 2.2.2 and Lemma 2.2.3 we have

$$c^1 y^{2k} < c^1 y^{2k-1} < \cdots < c^1 y^1.$$

Since an augmentation step moves to any point with improving cost, the algorithm may be forced to visit all $2k$ points when optimizing over c^1.

For $\ell \geq 2$ and ℓ even, y^1 maximizes $c^{\ell-1}$ over P_n and

$$c^\ell y^1 < c^\ell y^{2k} < c^\ell y^{2k-1} < \cdots < c^\ell y^{k+1},$$

so the bit scaling algorithm may visit all k points in $\{y^{k+1}, \ldots, y^{2k}\}$ in the ℓth scaling phase. Similarly, for $\ell \geq 2$ and ℓ odd, y^{k+1} maximizes $c^{\ell-1}$ over P_n and

$$c^\ell y^{k+1} < c^\ell y^k < c^\ell y^{k-1} < \cdots < c^\ell y^1,$$

so the algorithm may visit all k points in $\{y^1, \ldots, y^k\}$. Thus, for $\ell \in \{1, \ldots, p\}$, at least k augmentations may be necessary to optimize over c^ℓ. As $p = \lceil \log \|c^p\|_\infty \rceil$, this gives a total number of (at least)

$$k \lceil \log \|c^p\|_\infty \rceil = \frac{n+2}{8} \lceil \log \|c^p\|_\infty \rceil \in \Omega(n \log \|c^p\|_\infty)$$

augmentations necessary over the entire algorithm. $\square$

2.2.4 Geometric scaling

While the bit scaling algorithm is only valid for 0/1 polytopes, the geometric scaling algorithm (first given in [67]) can be used for general integer programs so long as the feasible region can be bounded. In particular, we aim to solve $\max\{cx : x \in P \cap \mathbb{Z}^n\}$ for an objective function $c \in \mathbb{Z}^n$ and a polytope $P := \{x \in \mathbb{R}^n : Ax = b, l \leq x \leq u\}$ with $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$, and $l, u \in \mathbb{Z}^n$. For the sake of readability of the results in this section, we define $C := \|c\|_\infty$, $U := \max_{i \in [n]} u_i$, and $L := \min_{i \in [n]} l_i$.

We once again seek to solve this optimization problem by a number of augmentation steps, starting with feasible solution $x \in P \cap \mathbb{Z}^n$. For this algorithm, the augmenting steps require to first find an augmenting direction $z \in \mathbb{Z}^n$ with $x + z \in P$ and $cz > 0$. Such a direction should be feasible, i.e., it should satisfy $x + z \in P$. A feasible direction z is *exhaustive* for x and P if $x + 2z \notin P$. Note that an exhaustive direction is always nonzero, and by integrality, an integer feasible direction is exhaustive for P if and only if it is exhaustive for the integral hull P_I.

Algorithm 2 Geometric scaling

```
Input: Integer feasible solution $x^0$
Output: Optimal solution for max $\{cx \, : \, x \in P \cap \mathbb{Z}^n\}$
  $\mu \leftarrow 2C(U - L)$, $\tilde{x} \leftarrow x^0$
  repeat
      compute $z$ solution to AUG with inputs $A, l, u, \tilde{x}, c - \mu\rho^+(x), -c - \mu\rho^-(x)$
      if there is no feasible solution then
          $\mu \leftarrow \mu/2$
      else
          pick $\alpha \in \mathbb{Z}_+$ with $\alpha \geq 1$ so that $\alpha z$ is an exhaustive direction
          $\tilde{x} \leftarrow \tilde{x} + \alpha z$                    ▷ update solution and repeat
      end if
  until $\mu < 1/n$
  return $\tilde{x}$                                   ▷ return optimal solution
```

We are now ready to present the geometric scaling algorithm, first found in [67] and presented here as Algorithm 2. On first glance, it appears very similar to bit scaling it that it requires a known feasible solution, then passes through many scaling phases marked by the changing values of μ (this time we have no explicit phase counter k). A major difference exists in the augmentation step, however. To understand what is happening here, we first must define the problem AUG, which we do next.

Problem: Directed augmentation (AUG)

Input: Matrix $A \in \mathbb{Z}^{m \times n}$, bounds $l, u \in \mathbb{Z}^n$, vectors $\tilde{x}, w^+, w^- \in \mathbb{Q}^n$

Output: Either output $z = z^+ - z^-$ with $A(z^+ - z^-) = 0, 0 \leq z^+ \leq u - x$,

$0 \leq z^- \leq x - l$, and $w^+z^+ + w^-z^- > 0$, or assert none exists

The augmentation problem AUG requires us to find a direction z broken into two components z^+ and z^-. It is convenient to think of these as the positive and negative parts of z, such that $z_j^+ z_j^- = 0$ for each coordinate $j \in [n]$. Our analyses do not require this $z_j^+ z_j^- = 0$ condition to hold, however, and we only require $z = z^+ - z^-$. The vectors z^+ and z^- must satisfy a set of linear constraints (indeed, if we hadn't split z into z^+ and z^-, the requirements would be nonlinear). In the context of Algorithm 2, the $A(z^+ - z^-) = 0$, $0 \leq z^+ \leq u - x$, and $0 \leq z^- \leq x - l$ constraints stipulate that z is a feasible direction for P at $\tilde{x}$.

The constraint involving w^+ and w^- requires some explanation. In the context of the algorithm, the AUG problem is told to satisfy (for some as-yet undefined functions $\rho^+, \rho^- : \mathbb{Z}^n \to \mathbb{Q}^n$)

$$\begin{aligned} 0 &< (c - \mu\rho^+(\tilde{x}))z^+ - (c_\mu\rho^-(\tilde{x}))z^- \\ &= c(z^+ - z^-) - \mu(\rho^+(\tilde{x})z^+ + \rho^-(\tilde{x})z^-) \end{aligned}$$

We can define, for any $z \in \mathbb{Z}^n$ with positive part z^+ and negative part z^-, the function

$$\rho(\tilde{x}, z) := \rho^+(\tilde{x})z^+ + \rho^-(\tilde{x})z^-.$$

Using this definition and the discussion above, we can say the following about the directions returned by the AUG procedure in Algorithm 2.

Lemma 2.2.5. *Any direction z returned by the AUG procedure in the geometric scaling algorithm satisfies*

$$cz - \mu\rho(\tilde{x}, z) > 0 \qquad \textit{or, equivalently} \qquad \frac{cz}{\rho(\tilde{x}, z)} > \mu.$$

So we see that the call to AUG in the geometric scaling algorithm requires a direction z that is feasible for P at $\tilde{x}$ while also guaranteeing a minimum improvement in objective quality. This improvement is scaled by some *potential function* ρ. At this point, the search for an augmenting direction may appear similar to the improvement steps taken in the interior point methods from other realms of optimization. Indeed the search for augmenting directions is very similar to the Newton directions obtained from the derivatives of the classical barrier function for linear programs (see, e.g., [5, Section 4]).

We would like these potential functions to satisfy certain criteria. In particular, we want

1. $\rho(x, z) \in O(\text{poly}(n))$,
2. $\rho(x, z) = \Omega(1/\text{poly}(n))$ whenever z is exhaustive for x, and
3. $\rho(x, \alpha \cdot z) = \alpha \cdot \rho(x, z)$ for all $\alpha \geq 0$.

The first two criteria are important for the following upper bound proofs, as such conditions will guarantee a polynomial bound on the number of augmentations necessary. The importance of the third criterion is due to the step in Algorithm 2 where we scale z to be exhaustive. This homogeneity conditions assures that αz confers the same benefit to the objective function relative to p as z does, i.e.

$$\frac{c(\alpha z)}{\rho(\tilde{x}, \alpha z)} = \frac{cz}{\rho(\tilde{x}, z)} > \mu.$$

We will now give an example potential function ρ which we will show to fit each of these criteria. In fact, this is the same potential function that is used in [67]. We will continue to use this particular function in our proofs, while allowing that similar results may be given by using different potential functions. For the remainder of the section, we will define $\rho(x, z) = \rho^+(x)z^+ + \rho^-(x)z^-$ with

$$\rho^+(x)_j = \begin{cases} \frac{1}{u_j - x_j}, & \text{if } x_j < u_j \\ \infty, & \text{otherwise} \end{cases} \quad \text{and} \quad \rho^-(x)_j = \begin{cases} \frac{1}{x_j - l_j}, & \text{if } x_j > l_j \\ \infty, & \text{otherwise} \end{cases}$$

for each coordinate $j \in [n]$.

Lemma 2.2.6. *The function ρ as defined above satisfies the requirements of a potential function. In particular,*

1. *$\rho(x, z) \leq n$ for all integer feasible points x and feasible directions z;*
2. *$\rho(x, z) > \frac{1}{2}$ whenever z is exhaustive for x.*
3. *$\rho(x, \alpha \cdot z) = \alpha \cdot \rho(x, z)$ for all $\alpha \geq 0$.*

Proof. One easily verifies property 3 from the definition of ρ. Let $z = z^+ - z^-$ be an integer feasible direction and let x be integer feasible for P. We will show that for each $j \in [n]$ we have $p(x)_j z_j^+ + n(x)_j z_j^- \leq 1$. Since $(p(x)_j z_j^+) \cdot (n(x)_j z_j^-) = 0$ by the definition of the positive and negative part, it suffices to prove that $p(x)_j z_j^+ \leq 1$ and $n(x)_j z_j^- \leq 1$. We consider the term $n(x)_j z_j^-$; the proof is analogous for $p(x)_j z_j^+$. Observe that whenever $x_j = l_j$ then $z_j^- = 0$, and hence $n(x)_j z_j^- = 0$ in this case. Thus, suppose that $x_j > l_j$. Then

$$n(x)_j z_j^- = \frac{z_j^-}{x_j - l_j} \leq 1,$$

because z is a (feasible) direction.

Now suppose that z is exhaustive for x, i.e., $x + z \in P$, but $x + 2z \notin P$. By definition, $\rho(x, z) \geq 0$. Moreover, since z is exhaustive, there exists $j \in [n]$ with either $x_j + 2z_j > u_j$, i.e., $z_j^+ > (u_j - x_j)/2$ or $x_j + 2z_j < l_j$, i.e., $z_j^- > (x_j - l_j)/2$. Hence, $p(x)_j z_j^+ > \frac{1}{2}$ in the former case or $n(x)_j z_j^- > \frac{1}{2}$ in the latter case. □

It was shown in [67] that the geometric scaling algorithm requires $O(n \log(nC(U - L)))$ augmentations before terminating. With slight modification to their proof, we can reduce this bound to $O(n \log(C(U - L)))$. We present this result now. To keep the presentation of the main proof clean, we next prove some sub-results in support if the main theorem.

Lemma 2.2.7. *Suppose the geometric scaling algorithm augments from solution $\tilde{x}$ to solution $x = \tilde{x} + \alpha z$. Then $cx > c\tilde{x}$.*

Proof. The result follows simply from Lemma 2.2.5

$$c(x - \tilde{x}) - \mu\rho(\tilde{x}, x - \tilde{x})$$

and the fact that $\mu, \rho > 0$. □

Lemma 2.2.8. *Let $\tilde{x}$ be the last solution in the scaling phase for μ in the geometric scaling algorithm. Then*

$$c(x - \tilde{x}) \leq \mu\, n$$

for any integer $x \in P$.

Proof. If $\tilde{x}$ is the final solution in the phase for μ, this means that no $x \in P \cap \mathbb{Z}^n$ exists with $c(x - \tilde{x}) - \mu \cdot \rho(\tilde{x}, x - \tilde{x}) > 0$. Since ρ is bounded by n, the result follows. □

Lemma 2.2.9. *The geometric scaling algorithm completes at most $4n$ augmentation steps between successive updates of μ.*

Proof. Let $y^0, y^1, \ldots$ be the points in P visited by the algorithm during the scaling phase for a given μ. In particular, y^0 is the current solution after the last update of μ. By Lemma 2.2.8, we have

$$c(x^* - y^0) \leq 2\mu n,$$

where x^* is an integral optimal solution for the original problem. Now, consider any two consecutive iterates y^i and y^{i+1}. By definition of Algorithm 2, the relation $c(y^{i+1} - y^i) - \mu \cdot \rho(y^i, y^{i+1} - y^i) > 0$ holds. Moreover, as the direction $y^{i+1} - y^i$ is exhaustive, using Lemma 2.2.6 we have

$$c(y^{i+1} - y^i) > \mu \cdot \rho(y^i, y^{i+1} - y^i) \geq \frac{\mu}{2} \geq \frac{1}{4n} c(x^* - y^0).$$

Hence after $4n$ augmentations from y^i to y^{i+1}, we come to a point with objective at least as high as x^*. Thus by Lemma 2.2.7 no more augmentations are possible. □

Already with the above results, one can show the $O(n \log(nC(U - L)))$ bound of [67] by simply noting that the algorithm goes through $\log(nC(U - L))$ scaling phases. The improvement to $O(n \log(C(U - L)))$ comes by noticing that the absolute gap between the current solution and the optimal solution after $\log(C(U - L))$ phases is at most n, implying that only n augmentations are necessary from that point forward.

Theorem 2.2.10. *Suppose the geometric scaling algorithm is run with ρ the potential function from Lemma 2.2.6. Then the algorithm completes after $O(n\log(C(U-L)))$ augmentation steps.*

Proof. The algorithm initializes with $\mu = 2C(U-L)$. Hence after $\lceil\log(C(U-L))\rceil + 1$ updates of μ, we have $\mu \leq 1$, and by Lemma 2.2.9 have completed at most $4n(\lceil\log(C(U-L))\rceil + 1)$ augmentations in total. Let $\tilde{x}$ be the last solution computed by the algorithm after these first $\lceil\log(C(U-L))\rceil + 1$ scaling phases.

At this point we may stop counting augmentations per phase and simply count the number of remaining improvements that are possible. As $\mu \leq 1$, Lemma 2.2.8 implies $c(x^* - \tilde{x}) \leq n$, where x^* is an integral optimal solution with respect to c. Since all data is integral and by Lemma 2.2.7, every augmentation improves the objective function by at least 1. It follows that no more than n solutions may be generated before obtaining a solution with cost cx^*. Hence the algorithm terminates after at most $4n(\lceil\log(C(U-L))\rceil + 1) + n$ augmentations. □

To close this section, we note that this result has further implications when comparing bit scaling to geometric scaling for $P \subseteq [0,1]^n$. We will see this in Section 2.2.5.

2.2.5 Comparing bit scaling and geometric scaling

We've already seen that geometric scaling is more versatile than bit scaling in that bit scaling is valid only over $P \subseteq [0,1]^n$. If we focus strictly on the domain where they are both valid (again, $P \subseteq [0,1]^n$), is there reason to favor one over the other?

First, we note that if $P \subseteq [0,1]^n$, then we may set $l = \mathbf{0}, u = \mathbf{1}$ for the geometric scaling algorithm. Thus Theorem 2.2.10 tells us that $O(n\log\|c\|_\infty)$ augmentations are necessary, equivalent to the bound for bit scaling in Lemma 2.2.1 (note that using the results of [67], the bound for geometric scaling would only be $O(n \log(n \|c\|_\infty))$). Hence, judging by the maximum number of augmentations needed, no preference is found for one over the other.

A point against geometric scaling could be that the associated augmentation problem is in dimension $2n$, since the augmenting direction z must is split by means of $z = z^+ - z^-$. However, it is easy to show that the potential function given in Section 2.2.4 reduces to $\rho(x, z) = |\operatorname{supp} z|$ when all feasible solutions are 0/1 vectors. With x known, this can be converted into a linear function in n variables, eliminating this possible advantage for bit scaling.

However, the worst-case example for bit scaling from Section 2.2.3 highlights a distinct advantage for geometric scaling over bit scaling. Recall that in this example, the bit scaling algorithm may be forced to augment $O(n \log \|c\|_\infty)$ times over the course of the algorithm. However, the example polyhedron P contains only $\Theta(n)$ integer points. Bit scaling achieves a high worst-case bound here because it may be forced to revisit the same solution $x \in P$ multiple times. However, from Lemma 2.2.7 we know that the geometric scaling algorithm will *never* revisit a point. Thus for the same example, geometric scaling will augment only $O(n)$ times. Hence we have the following result.

Corollary 2.2.11. *For any $p \geq 1$, there exists a polytope $P \subseteq [0,1]^n$ with $n = 8k + 2$, $k \in \mathbb{Z}_{>0}$ and an objective function $c = c^p$, so that bit scaling computes $\Omega(n \log \|c^p\|_\infty) = \Omega(np)$ augmenting directions in the worst case, while geometric scaling needs $O(n)$ augmenting directions. In particular, the relative difference can be made arbitrarily large by choosing p appropriately.*

As a final note to the section, we mention that with some preprocessing, this negative result for bit scaling can be mitigated. In particular, the rounding scheme of [32] can be used to turn an arbitrary $c \in \mathbb{Q}^n$ into a vector $\bar{c} \in \mathbb{Z}^n$ with encoding length $O(n^3)$ in time polynomial in n and $\log \|c\|_\infty$ such that optimizing both vectors results in the same optimal solution. Thus, bit scaling requires at most $O(n^4)$

augmentations in the worst-case *with* preprocessing of the objective function. We obtain the same worst-case bound on the number of augmentations for geometric scaling.

2.2.6 Improved bounds for structured 0/1 polytopes

When proving worst-case bounds for both bit scaling and geometric scaling, a crucial element is the $O(n)$ bound on the number of improvements made per scaling phase. In the case of bit scaling, this bound is due to the number of positive entries in the vector $x - \tilde{x}$ being at most n for any integral point $x, \tilde{x} \in P$. For geometric scaling, the bound arises from potential function values. In particular, the potential $\rho(x,z) := |\operatorname{supp}(z)|$ is bounded from above by n. If this bound can be reduced for special polytopes, it would have direct consequences for worst-case bounds of either algorithm.

One condition that guarantees such a reduction is the following: Let $P \subseteq [0,1]^n$ be a polytope, and suppose there exists some function $f : \mathbb{Z}_+ \to \mathbb{Z}_+$ such that every integral point $x \in P$ has no more than $f(n)$ nonzero entries. In particular we are hoping for an $o(n)$ function, such as $\sqrt{n}$ or $\log n$. We then obtain the following improved worst-case bounds for both bit scaling and geometric scaling.

Theorem 2.2.12. *Let $c \in \mathbb{R}^n$ be a cost vector and $P \subseteq [0,1]^n$ a polytope. Suppose there exists a function $f : \mathbb{Z}_+ \to \mathbb{Z}_+$ such that every integral point $x \in P$ has at most $f(n)$ nonzero entries. Then, given an initial solution $x^0 \in P$, both Algorithm 1 and Algorithm 2 solve the optimization problem* $\max\{cx : x \in P \cap \mathbb{Z}^n\}$ *after $O(f(n) \log \|c\|_\infty)$ augmentations.*

Proof. For Algorithm 1 the proof follows as for Lemma 2.2.1, but replacing the bound $d^k(x^k - x^{k-1}) \leq n$ by $d^k(x^k - x^{k-1}) \leq 2f(n)$. In the case of Algorithm 2, the proofs goes as the one for Theorem 2.2.10, but using that $L = 0, U = 1$, and $\rho \leq 2f(n)$. □

Many well-studied polytopes satisfy that the integral points have a number of vertices that is $o(n)$, especially those arising from graph-theoretic problems. For example, take the traveling salesman polytope $P \subseteq [0,1]^{|E|}$ on the complete graph with k nodes and $|E| = \binom{k}{2}$ edges. Even though the polytope is contained in a space of dimension $\binom{k}{2}$, its integral points (corresponding to tours on the graph) contain exactly k nonzero entries, spanning a low dimensional subspace. Hence optimizing over P using either Algorithm 1 or Algorithm 2 can be done in $O(k \log \|c\|_\infty)$ augmentations, a factor-k improvement over the general $O(k^2 \log \|c\|_\infty)$ upper bound.

2.3 *Implementation*

After having examined theoretical questions involving the various scaling methods, it is natural to wonder whether these methods might be of practical use for solving integer programs. To test this, we must address how to solve the associated augmentation problems, which we ignored during theoretical analysis. The strategy employed here is to use a MIP solver to carry out the augmentation steps. One may wonder how using a MIP solver as a subroutine could ever improve upon using the same solver to solve the problem outright. The idea is that by guiding the early stages of the algorithm with scaled objectives that are perhaps more "simple" than the full objective, one can make primal gains more quickly and prune more of the branch-and-bound tree earlier in the algorithm.

The augmenting MIP is generally solved by adding an objective cut $c(x - \tilde{x}) \geq \delta$ for an appropriately chosen $0 < \delta < 1$. We run this MIP until an improving solution is found. In the case of geometric scaling, we then exhaust the direction obtained via a line search.

2.3.1 Algorithms

We now briefly review the types of algorithms tested. For a fuller accounting of each method, we direct the reader to [47], the preprint version of an article based on this work. The following is a list of the algorithms tested.

- **Augment:** A basic augmenting algorithm that does not use scaling to guide the search for new solutions. The associated MIP subproblems consist of the problem's original constraints plus an objective cut.

- **Bit scaling:** The classical bit scaling algorithm of Algorithm 1 is implemented, along with a variant that maximizes c^k at each augmenting step (hence there is only one augmentation per scaling phase). A few variants on these basic ideas are tested.

- **Geometric scaling:** Algorithm 2 is tested with pseudo-potential function $\rho(x,z) = \|z\|_1$. This function does not satisfy the linear homogeneity condition $\rho(x,\alpha z) = \alpha(x,z)$ (except for $0/1$ programs, where scaling is unnecessary), but is used anyway due to it's simplicity. We also modify the algorithm slightly by choosing a different factors by which μ is altered in each phase (by default it is divided by 2, but we also try values ranging from 8 to 1024).

- **Primal heuristic based on geometric scaling:** Initial results from the geometric scaling algorithm motivated us to test a heuristic using the method. In particular, we use the MIP solver's default settings unless no primal solution has been found after a certain number of nodes are solved. The geometric scaling algorithm is then run for a predetermined amount of time, and any solutions recovered are returned back to the master problem. Regular branch-and-cut resumes until the conditions for the heuristic are met again. We also test two further variants of this scheme - one uses inference branching instead of default branching, and the other uses the factor 64 to update μ in each scaling phase.

2.3.2 Results

We implemented the discussed algorithms in C using the framework SCIP, see [1, 69]. In particular, we use SCIP 3.2.0 with CPLEX 12.6.1 as the LP-solver. SCIP runs with default settings, except that we turn off the "components" presolver, since it would decompose the problem into several runs, making a comparison more difficult. Tests were run on a Linux cluster with 3.2 GHz Intel i3 processors with 8 GB of main memory and 4 MB of cache, running a single process at a time.

We use the following test sets:

MIPLIB2010 The 87 benchmark instances from MIPLIB 2010[1], see [46].

LB We use the test set of 29 instances from the "local branching" paper[2], see [30]. This test set has also been used in [38].

QUBO We use a test set of linearizations of 50 instances for quadratically unconstrained Boolean optimization (QUBO)[3], see [22, 23].

We now briefly summarize the results of the tests, noting that further discussions are found in [47], and full results are available in the online supplement referenced within.

2.3.2.1 MIPLIB 2010 test set

Table 1 shows a comparison of the different augmentation methods (along with SCIP/CPLEX running on default parameters) on the test set MIPLIB2010. In the table, "#nodes" and "time" give the shifted geometric means[4] over all instances of

[1]available at http://miplib.zib.de/

[2]available at http://www.or.deis.unibo.it/research_pages/ORinstances/MIPs.html

[3]available at http://researcher.watson.ibm.com/researcher/files/us-sanjeebd/chimera-data.zip

[4]The shifted geometric mean of values $t_1, \ldots, t_n$ is defined as $\left(\prod(t_i + s)\right)^{1/n} - s$ with shift s. We use a shift $s = 10$ for time and $s = 100$ for nodes in order to decrease the strong influence of the very easy instances in the mean values.

Table 1: Aggregated results of the different algorithms on test set MIPLIB 2010 (1 hour time limit, 87 instances)

name	#nodes	time	#run	#best	#improv.	#subprob.	#phases	#exhaust	prim-$\int$
augment	14793.5	924.71	83	63	12.9	12.9	12.9	0.0	54.5
bitscale	20116.2	934.93	59	67	3.7	8.4	4.8	0.0	57.4
bitscale-classic	25086.0	1120.62	59	62	4.1	11.0	6.9	0.0	60.4
bitscale-noimprove	20343.4	918.31	60	69	4.2	8.8	4.6	0.0	56.7
bitscale-complete	26902.6	1070.71	59	59	2.0	6.6	4.6	0.0	103.5
geometric	5628.8	1632.31	83	65	6.2	23.6	17.4	0.0	49.8
geom-8	8637.7	1313.55	83	69	5.7	12.8	7.1	0.0	40.7
geom-64	8680.9	1122.85	83	72	6.4	10.7	4.3	0.0	39.7
geom-256	8358.2	1128.40	83	69	7.1	10.9	3.7	0.0	39.7
geom-512	9895.4	1112.03	83	69	7.0	10.3	3.3	0.0	41.8
geom-1024	8392.5	1016.80	83	71	7.1	10.3	3.2	0.0	39.8
geom-heur	16858.8	741.52	68	71	1.3	33.6	32.4	0.0	30.4
geom-heur-infer	21343.6	732.30	68	72	1.0	41.7	40.7	0.0	30.1
geom-heur-64	16158.0	683.18	68	73	1.7	9.9	8.1	0.0	29.3
default	15495.9	557.12	0	74	0.0	0.0	0.0	0.0	26.3

the total number of nodes (including subproblems) and the time (in seconds), respectively. Column "#run" presents the number of instances for which an augmentation routine ran. Column "#best" refers to the number of times the best known primal solution value has been found. With respect to the augmentation methods, the columns "#improv.", "#subprob.", "#phases", and "#exhaust" refer to the average number of times an improved primal solution has been found, the number of subproblems (MIPs) solved, the number of phases, and the number of exhausting directions found, respectively. The number of phases refers to the number of subproblems solved with the same value of μ for bit and geometric scaling (in this case, #phases + #improv = #subprob); note that we count a possible search for the first primal solution as one phase. For augment, the number of phases equals the number of improving solutions and the number of subproblems.

Finally, the last column gives the *primal integral*, see [8]. The primal integral is the value we obtain by integrating the gap between the current primal and best primal bound over time[5]. Thus, a smaller primal integral indicates a higher solution quality over time.

[5]We define the gap between primal bound p and best primal bound b as $|p - b| / \max(|p|, |b|)$.

The overall picture here for the augmentation algorithms is bleak. The default MIP solver clearly outperforms the primal methods on these instances. Among the primal methods, the standard augmenting procedure performs surprisingly well. The bit scaling methods appear to achieve slightly better times than the geometric scaling variants, though there is not much difference overall, and geometric scaling found the best solution more often. Amongst all non-default methods, the geometric scaling-based heuristic performs best in terms of time, despite a fairly high number of nodes being solved.

Although the results here are not encouraging, there are good reasons to suspect this should be the case. First, the MIPLIB collections are well-known and broadly-used test sets. As such, they are often used for benchmark analyses, and there is reason to believe that many solvers are (not necessarily intentionally) overtuned to perform well on them. Second, these are relatively easy problems already. One could suspect that the true strength of primal methods is quickly finding good primal solutions, something that may be more apparent on problems that cannot be solved quickly.

2.3.2.2 LB test set

For the next test set, we compare the basic augmenting procedure, the best of the bit scaling and geometric scaling variants, and the three geometric scaling-based heuristic methods. We display the best solution found by each method after an hour time limit in Table 2.

Results show that several augmentation methods are competitive with the default settings when it comes to identifying good primal solutions. Indeed, the vanilla geometric scaling-based heuristic was responsible for the most "best" values, finding solutions at least as good as all other methods in 18 for the 29 instances. The generic augmenting procedure performs the worst, finding the best value only

Table 2: Best primal values for different variants on the LB test set (29 instances, 1 hour time limit). All problems are minimization instances. For each instance, the best values obtained are marked in black, otherwise the values are marked gray.

problem	default	augment	bitscale	geom-64	geom-heur	geom-heur-infer	geom-heur-64
A1C1S1	11,643.33	11,989.36	11,977.50	11,638.86	11,557.22	11,566.59	11,590.45
A2C1S1	10,983.28	11,422.77	11,115.34	11,040.72	10,897.77	10,994.27	10,909.95
arki001	7,580,813.05	7,581,527.87	7,580,813.05	7,582,202.93	—	7,580,814.51	7,580,813.05
B1C1S1	24,798.51	25,456.98	27,309.51	25,458.30	25,630.75	25,123.51	25,042.56
B2C1S1	25,763.12	27,253.74	26,592.19	26,167.32	26,412.44	25,926.61	26,002.11
biella1	3,065,005.78	3,065,005.78	3,065,005.78	3,065,005.78	3,065,005.78	3,065,005.78	3,065,005.78
core2536-691	689.00	689.00	689.00	689.00	689.00	689.00	689.00
core2586-950	970.00	972.00	1213.00	971.00	955.00	960.00	966.00
core4284-1064	1091.00	1100.00	3279.00	1080.00	1072.00	1073.00	1079.00
core4872-1529	1580.00	1584.00	1769.00	1579.00	1546.00	1560.00	1575.00
danoint	65.67	65.67	65.67	65.67	65.67	65.67	65.67
glass4	1,600,013,500.00	1,500,014,200.00	2,200,016,050.00	1,620,014,440.00	1,500,012,650.00	1,550,012,462.72	1,566,683,416.66
markshare1	7.00	9.00	32.00	12.00	10.00	10.00	10.00
markshare2	12.00	13.00	128.00	17.00	14.00	14.00	10.00
mkc	−559.11	−542.28	−557.56	−561.93	−562.93	−560.85	−561.33
net12	214.00	214.00	214.00	214.00	214.00	214.00	214.00
NSR8K	127,262,743.24	68,351,187.10	2,176,184,843.46	21,415,513.00	127,262,743.24	127,262,743.24	127,262,743.24
nsrand_ipx	51,200.00	54,880.00	55,200.00	52,000.00	51,200.00	51,200.00	51,200.00
rail507	174.00	174.00	174.00	174.00	174.00	174.00	174.00
roll3000	12,890.00	12,899.00	13,380.00	12,904.00	12,890.00	12,890.00	12,890.00
seymour	425.00	425.00	425.00	424.00	424.00	425.00	424.00
sp97ar	663,515,230.72	726,599,877.76	674,470,726.72	662,299,239.68	674,213,859.52	664,157,022.72	673,642,038.40
sp97ic	435,258,209.12	450,307,285.28	430,937,067.04	439,446,697.12	434,570,609.44	432,663,431.84	439,022,248.00
sp98ar	530,322,047.84	551,452,928.96	532,671,408.48	530,242,941.12	530,437,736.32	530,489,389.92	530,251,516.00
sp98ic	451,409,231.04	465,544,414.56	455,081,136.48	450,843,038.08	450,519,098.72	449,226,843.52	453,626,659.52
swath	494.09	502.24	506.44	495.02	467.41	481.95	477.57
tr12-30	130,596.00	130,596.00	139,741.00	130,596.00	130,596.00	130,596.00	130,596.00
UMTS	30,094,335.00	30,091,967.00	30,091,457.00	30,092,333.00	30,093,479.00	30,092,081.00	30,091,738.00
van	5.09	5.59	5.35	6.12	5.09	5.09	5.09
#best:	13	6	8	10	18	10	11

for instances where (almost) all other instances found the same solution.

Overall, the scaling methods presented perform admirably as compared to the default MIP solver. This is in contrast to what was found on the MIPLIB2010 test set, perhaps giving evidence to our hypothesis that primal augmentation techniques are more effective on more difficult problems.

2.3.2.3 QUBO test set

For the QUBO test set, we test the default MIP solver versus geometric scaling and the heuristics based on it. Table 3 shows the best primal values and the primal integral achieved by each method on each instance. For these instances, the other standalone augmentation methods do not perform well – we skip their results here.

These particular instances are much harder (in terms of necessary solve time and known optimality gaps) than instances in the other test sets. Interestingly, on these test instances the default settings are no longer competitive with the scaling-based methods when it comes to finding good solutions. Indeed, in only one of 50 instances did the method find a solution stronger than any of the other methods. Primal integral values are generally higher as well.

The winner here appears to be geometric scaling, which for the first time outperforms the geometric scaling-based heuristics on a test set. Overall, the QUBO instances show the best potential for scaling-based augmentation methods.

2.4 Conclusions and future work

This work gives tightened theoretical analyses for scaling-based augmentation methods. In particular, the upper bound for augmentations in the geometric scaling method is improved, an example shows that bit scaling may require its worst-case number of augmentations, and it is shown that bit scaling can perform arbitrarily worse than geometric scaling on the same problem. It is still an open problem to

Table 3: Best primal values and primal integral of the default settings and variants of the heuristic based on geometric scaling (50 instances, 1 hour time limit). For each instance, the best primal values are marked in black, otherwise the values are marked gray; all problems are minimization instances.

Problem	default		geom-64		geom-heur		geom-heur-infer		geom-heur-64	
	Primal	Prim-$\int$	Primal	Prim-$\int$	Primal	Prim-$\int$	Primal	Prim-$\int$	Primal	Prim-$\int$
chim8-4.1	−796	153.8	−822	58.9	−830	109.9	−788	188.3	−798	144.9
chim8-4.2	−776	192.6	−766	193.5	−794	60.5	−800	33.4	−806	17.8
chim8-4.3	−784	281.5	−840	145.7	−790	218.7	−790	218.6	−800	181.7
chim8-4.4	−806	291.1	−876	24.7	−828	203.5	−840	160.0	−852	107.2
chim8-4.5	−850	208.1	−882	58.0	−840	175.1	−828	223.3	−852	128.5
chim8-4.6	−790	218.4	−798	189.7	−840	140.9	−830	158.5	−836	132.6
chim8-4.7	−756	301.5	−810	134.2	−802	102.7	−824	188.6	−806	85.3
chim8-4.8	−786	248.0	−822	26.2	−796	125.8	−824	4.8	−814	68.7
chim8-4.9	−850	154.6	−810	265.7	−872	159.9	−802	291.8	−828	186.1
chim8-4.10	−810	349.5	−896	17.7	−812	341.1	−830	269.3	−828	280.2
chim8-4.11	−764	154.6	−768	113.3	−748	200.7	−790	68.6	−730	280.8
chim8-4.12	−746	306.5	−774	161.7	−786	105.7	−780	130.9	−808	8.2
chim8-4.13	−818	158.7	−848	17.8	−798	218.4	−788	259.6	−800	210.5
chim8-4.14	−806	63.7	−770	218.3	−818	36.4	−800	140.5	−792	124.6
chim8-4.15	−836	243.4	−896	14.9	−868	115.9	−880	67.8	−868	118.3
chim8-4.16	−834	164.7	−852	70.1	−814	205.4	−862	42.0	−818	194.2
chim8-4.17	−802	102.7	−732	376.1	−810	81.2	−816	157.7	−796	114.7
chim8-4.18	−856	63.4	−808	268.0	−868	15.2	−870	5.6	−870	11.8
chim8-4.19	−870	200.2	−906	35.3	−886	84.4	−876	126.0	−866	164.5
chim8-4.20	−818	249.5	−874	31.1	−878	51.5	−812	274.6	−850	120.3
chim8-4.21	−818	98.7	−816	120.5	−836	22.3	−830	47.6	−840	5.6
chim8-4.22	−816	122.7	−834	53.3	−820	110.9	−844	58.7	−834	48.3
chim8-4.23	−780	196.4	−824	39.6	−788	168.6	−768	249.4	−798	120.2
chim8-4.24	−840	222.8	−834	199.5	−862	87.4	−862	91.6	−880	123.4
chim8-4.25	−880	91.2	−872	115.8	−872	80.5	−858	136.8	−890	68.0
chim8-4.26	−796	190.6	−838	66.1	−792	205.4	−826	174.8	−788	220.9
chim8-4.27	−834	54.5	−844	36.3	−846	16.0	−834	57.0	−834	57.2
chim8-4.28	−782	82.4	−746	249.8	−792	34.2	−790	44.2	−798	11.4
chim8-4.29	−790	193.3	−820	74.4	−792	186.2	−834	94.3	−810	113.8
chim8-4.30	−842	200.1	−808	341.0	−870	87.4	−864	111.5	−890	42.6
chim8-4.31	−870	137.1	−890	58.8	−900	63.3	−882	198.9	−860	243.8
chim8-4.32	−790	192.8	−830	11.1	−824	144.3	−818	131.9	−822	160.7
chim8-4.33	−884	98.4	−830	275.1	−878	83.5	−896	51.4	−870	110.0
chim8-4.34	−890	24.9	−882	53.5	−872	80.2	−876	68.7	−860	126.7
chim8-4.35	−782	98.6	−798	84.0	−788	51.3	−774	112.7	−794	25.1
chim8-4.36	−788	133.1	−816	13.4	−792	111.2	−776	180.2	−780	164.1
chim8-4.37	−790	64.7	−798	12.1	−760	176.5	−798	72.7	−798	33.7
chim8-4.38	−806	349.6	−796	266.9	−856	154.8	−792	277.7	−840	184.2
chim8-4.39	−856	79.6	−866	20.7	−850	70.5	−846	86.8	−846	87.6
chim8-4.40	−790	82.9	−760	303.5	−788	79.8	−802	71.6	−800	38.9
chim8-4.41	−880	209.0	−890	36.7	−846	185.6	−836	223.6	−864	114.4
chim8-4.42	−658	190.7	−694	210.2	−678	89.6	−678	89.9	−684	58.8
chim8-4.43	−734	108.1	−740	86.0	−756	4.6	−742	70.2	−752	23.9
chim8-4.44	−742	145.7	−704	328.5	−772	174.5	−756	81.3	−764	48.2
chim8-4.45	−818	251.7	−846	26.3	−842	29.5	−842	24.6	−834	81.0
chim8-4.46	−854	154.6	−854	128.0	−840	152.5	−874	68.8	−832	179.0
chim8-4.47	−848	136.6	−856	66.1	−868	96.4	−834	145.0	−850	88.2
chim8-4.48	−826	94.0	−834	12.2	−812	98.6	−812	98.5	−832	18.1
chim8-4.49	−800	159.8	−820	15.1	−760	268.9	−746	328.3	−786	154.9
chim8-4.50	−822	114.2	−802	189.1	−814	124.9	−842	72.1	−838	39.9
#best	1		19		11		13		9	
AM prim-$\int$ (#50)		167.7		118.3		119.8		130.6		109.5
GM prim-$\int$ (#50)		148.0		73.5		94.6		100.0		79.4

give a family of instances where geometric scaling meets it's theoretical worst-case upper bound - in fact, the authors are unaware of any example that requires a number of augmentations super-linear in n. Indeed, it is possible that the new upper bound is still too loose. A result in either direction would be of interest.

The computational results suggest that scaling methods can help close the optimality gap on hard MIP instances. Each of the methods employed here involve several parameters, hence extensive parameter tuning could help to improve on these results. However, a primal algorithm that is competitive overall with traditional MIP solving techniques remains elusive.

CHAPTER III

CG AND MOD-K CUTS IN THE 0/1 CUBE

3.1 Introduction

Cutting planes hold great importance in integer programming as a tool for tightening linear programming relaxations. Perhaps the most important class of cutting planes historically are Chvátal-Gomory cuts (see [17, 34, 35]), which are born of a rather simple observation: If the inequality $c^T x \leq \delta$ is valid for a polyhedron P and $c \in \mathbb{Z}^n$, then the inquality $c^T x \leq \lfloor \delta \rfloor$ holds for all integral points in P.

Formally, let P be a rational polyhedron with integer hull $P_I = \text{conv}\,(P \cap \mathbb{Z}^n)$. A *Chvátal-Gomory cut* (or *CG cut*) is any inequality of the form $c^T x \leq \lfloor \delta \rfloor$ where $c \in \mathbb{Z}^n$ and $c^T x \leq \delta$ is valid for all $x \in P$. The *Chvátal-Gomory closure* (*CG closure*) of P, denoted P', is the intersection of P with its CG cuts. We will alternately denote $P^{(1)} := P'$ and recursively define $P^{(k)} = (P^{(k-1)})'$. It is well known that P' is again a rational polyhedron, and also there always exists finite $t \in \mathbb{Z}$ so that $P_I = P^{(t)}$. The smallest t for which this holds is known as the *Chvátal-Gomory rank* (*CG rank*) of P, which we denote by $\text{rk}(P)$.

3.1.1 Related work

While a polyhedron $P \subseteq \mathbb{R}^n$ always has a rank that is finite, it need not be small. Indeed, simple examples show that the CG-rank may be arbitrarily large as compared to the dimension n. However, if we restrict $P \subseteq [0, 1]^n$, this is no longer true. Bounds which are polynomial in n have long been known, with the current best bound of $\text{rk}(P) \in O(n^2 \log n)$ proved in [29]. No matching lower bound has been shown, though in [64] polytopes with rank $\Omega(n^2)$ are described.

While the previous bounds were stated in terms of the dimension n, other bounds which are completely independent of n are also known. In [19] CG rank bounds are derived based on the structure of the set $S = P \cap \mathbb{Z}^n$ of integral extreme points of $P \subseteq [0,1]^n$. In particular, from S construct a graph with a vertex corresponding to each point in the set $\{0,1\}^n \setminus S$, and edges between all pairs of points which differ in precisely one coordinate. If the treewidth of this graph is at most 2, then $\text{rk}(P) \leq 4$. These results are extended in [6] for any treewidth value (i.e. not only equal to 2), and the same work derives bounds based on other properties of S.

In [13], CG and other popular cuts are generalized as so-called *aggregation cuts*. This work derives novel lower bounds for CG rank in the case that the polyhedron P is of packing or covering type.

Complexity questions regarding the CG closure have also generated interest in the literature. Given a polyhedron P and rational $x \in P$, the CG separation problem asks if there exists a CG cut for P that is violated by x. This problem was shown to be NP-complete in the general case $P \subseteq \mathbb{R}^n$ by [28] and also recently for the case $P \subseteq [0,1]^n$ as reported in [48]. In [20], it is shown that deciding whether the CG closure of a rational polyhedron is empty is also an NP-complete problem, even if the polyhedron of interest contains no integer points.

Similar results can also be shown when CG cuts are replaced with different classes of cuts. One popular class of cuts are the so-called mod-k cuts, which we define now. Suppose that $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ with $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$. An alternate definition of a CG cut is any inequality of the form $\lambda^T Ax \leq \lfloor \lambda^T b \rfloor$ where $\lambda^T A \in \mathbb{Z}^n$ and $\lambda \in [0,1)^m$. A *mod-k cut* is a CG cut of the above form where we restrict $\lambda \in \{0, 1/k, ..., (k-1)/k\}^m$. Such cuts were introduced in [16], in which it was shown that separating over "maximally violated" mod-k cuts can be done in polynomial time. The cuts from the special case of $k = 2$ have received more attention in the literature, beginning with [15], under the name of $\{0, \frac{1}{2}\}$ cuts. It is already known that the separation problem for mod-2 cuts is NP-complete, both in the general case ([15]) and when we restrict $P \subseteq [0,1]^n$ ([52]).

3.1.2 Outline

The $\log(n)$ gap between the upper and lower bounds for $\mathrm{rk}(P)$, $P \subseteq [0,1]^n$ leaves open work for determining tighter bounds. In Section 3.2 we prove a bound that is strictly tighter than the best-known bounds from [29]. While the new bound is still only $O(n^2 \log n)$ in general, the nature of the proof allows us to give improved bounds for certain classes of polyhedra. In particular, we prove new bounds for symmetric polytopes, for polytopes with a limited number of integral points, and for certain polyhedra that arise from combinatorial optimization problems.

In Section 3.3, we address the complexity of the general separation problem over mod-k cuts. Mirroring the known results for both CG and $\{0, \frac{1}{2}\}$ cuts, we prove this problem to be NP-hard, even in the case the $P \subseteq [0,1]$.

In what follows, we use the notation $\mathbf{0}_{m\times n}$ to denote a matrix of size $m \times n$ consisting of all zeros. Similarly, $\mathbf{1}_{m\times n}$ denotes a matrix of all ones. We let I_n denote an identity matrix of size $n \times n$. We may suppress subscripts in cases where the dimension is clear by context.

3.2 New upper bounds for CG rank in 0/1 polytopes

In this section, we give a new upper bound on the rank of polyhedra P with $P \subseteq [0,1]^n$. The ingredients of the proof are very similar to that of [29], but with more care taken in the inductive steps to recover a tighter overall bound. After this bound is proven, we show how to use it to give improved bounds for certain classes of polyhedra.

3.2.1 Preliminaries

Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, $A \in \mathbb{Q}^{m\times n}, b \in \mathbb{Q}^m$ be a rational polyhedron. An inequality $c^T x \leq \delta$ is *valid* for P if it holds true for all $x \in P$. A *face* of P is any set of the form $P \cap \{x : c^T x = \delta\}$ for some valid inequality $c^T x \leq \delta$. A face F is called a *facet* if its dimension is one less than the dimension of P, i.e. $\dim(F) = \dim(P) - 1$. If $c \in \mathbb{R}^n, \delta \in \mathbb{R}$ are so that the face $P \cap \{x : c^T x = \delta\}$ is a facet, then we call c a *facet-defining vector* of P. Geometrically, c is normal to the facet, so we sometimes use the term *normal* in place of *vector* for c. As we are working with rational polyhedra, we may always choose facet defining vectors from $\mathbb{Z}^n$.

The *integrality gap* of $c \in \mathbb{R}^n$ with respect to polyhedron P is given by

$$\max_{x \in P} c^T x - \max_{x \in P_I} c^T x.$$

In other words, it is the difference between the optimal values obtained when maximizing c over P as opposed to its integral hull P_I. We say c is *saturated* with respect to P if this integrality gap is 0. It is elementary to see that if all of P_I's facet-defining vectors are saturated with respect to P, then $P = P_I$. A common approach to proving $\text{rk}(P) \leq k$ is to show that any integral vector (and hence any facet-defining vector for P_I) is saturated with respect to $P^{(k)}$.

As a rhetorical convenience, we may refer to a round of the *CG procedure* when describing the creation of P' from P. In a natural extension of this convention, we say that $P^{(k)}$ is the result of k applications of the CG procedure. Thus, saying that "c is saturated (with respect to P) after k rounds of the CG procedure," is taken to mean that "c is saturated with respect to $P^{(k)}$."

Our proofs make use of several previously known results. The following is a classic result in the theory of CG cuts and can be found for example in [66, p. 340].

Lemma 3.2.1. *Let F be a face of a rational polyhedron P. Then $F' = P' \cap F$.*

The next result, due to [12], bounds the rank of an integer-empty polytope $P \subseteq [0,1]^n$, and covers such cases in the main result of this section. We include a proof due to this importance.

Lemma 3.2.2. *Let $P \subseteq [0,1]^n$ be a d-dimensional polytope, $d \geq 1$, with $P_I = \emptyset$. Then* $\mathrm{rk}(P) \leq d$.

Proof. The proof goes by induction on d and n. For the base case, let $n \in \mathbb{Z}_+$ and suppose $d = 1$. Then P is the convex hull of two distinct points $a, b \in [0,1]^n$. Since P contains no integer points, for some coordinate $i \in [n]$ we must have $0 < a_i < 1$. Suppose $a_i \geq b_i$ (the $a_i \leq b_i$ case may be handled similarly). Then the inequality $x_i \leq a_i$ is valid for P. This means that $x_i \leq 0$ is valid for P', so $P' \subseteq \{x : x_i = 0\}$. Then $P' \subseteq \{b\}$, as any point $c \in P \setminus \{a, b\}$ can be written $c = \lambda a + (1-\lambda)b$ with $\lambda \in (0,1)$, and since $a_i > 0$ and $b_i \geq 0$, then $c_i \neq 0$. But a symmetric argument also proves that $P' \subseteq \{a\}$. Then $P' \subseteq \{a\} \cap \{b\} = \emptyset$, and hence $\mathrm{rk}(P) = 1$.

For the induction step, let $d, n > 1$. If $P \subseteq \{x : x_1 = 0\}$ or $P \subseteq \{x : x_1 = 1\}$ we are done by induction on n. Otherwise, the dimension of both $P_0 := P \cap \{x : x_1 = 0\}$ and $P_1 := P \cap \{x : x_1 = 1\}$ is strictly smaller than d. Then we have $\emptyset = P_0^{(d-1)} = P^{(d-1)} \cap \{x : x_1 = 0\}$ by the induction hypothesis (for the first equality) and Lemma 3.2.1 (the second), and similarly we have $P_1^{(d-1)} = \emptyset$. Thus $0 < \min\{x_1 : x \in P^{(d-1)}\} \leq \max\{x_1 : x \in P^{(d-1)}\} < 1$, implying that $P^{(d)} = \emptyset$. □

We note that the result does not hold for the pathological case $d = 0$; for this case, it is not hard to show $\mathrm{rk}(P) \leq 1$.

The following result, a re-worded version of [29, Lemma 3.1], is a key technical ingredient to both the bound of [29] and our new bound. It allows for a reduction in integrality gap of c when the associated face of P does not span the entire width of the cube $[0,1]^n$ in some coordinate.

Lemma 3.2.3. *Let $c^T x \leq \alpha$ be valid for P_I and $c^T x \leq \alpha + k + 1$ valid for P, with $c \in \mathbb{Z}^n$ and $P \subseteq [0,1]^n$ a polyhedron. If for some index $i \in \{1, ..., n\}$ it holds that $F := P \cap \{x : c^T x = \alpha + k + 1\}$ has empty intersection with $\{x : x_i = 0\}$ and $\{x : x_i = 1\}$, then*

$c^T x \leq \alpha + k$ is valid for $P^{(2)}$.

Proof. As $F \cap \{x : x_i = 1\} = \emptyset$, it follows that there exists $\epsilon \in (0, 1)$ such that $x_i \leq 1 - \epsilon$ is valid for F, and so $x_i \leq 0$ is valid for F'. Similarly, as $F \cap \{x : x_i = 0\} = \emptyset$, the inequality $x_i \geq 1$ is also valid for F', implying $F' = \emptyset$. Applying Lemma 3.2.1 we obtain

$$\emptyset = F' = P' \cap F = P' \cap \{x : c^T x \leq \alpha + k + 1\}.$$

Then there exists $\delta \in (0, 1)$ so that $c^T x \leq \alpha + k + 1 - \delta$ is valid for P', so $c^T x \leq \alpha + k$ is valid for $P^{(2)}$. $\square$

Lastly, we state a result from [29, Theorem 4.6], which suffices to prove the base case of the induction in our main result.

Lemma 3.2.4. *Let $P \subseteq [0, 1]^n$ be a nonempty polytope, and suppose $c \in \mathbb{Z}^n$ is so that $c^T x \leq \delta$ is valid for P_I. Then $c^T x \leq \delta$ has depth at most $n + \|c\|_1$ with respect to P.*

3.2.2 New upper bounds for CG rank

The upper bound in [29] is shown by first proving that any normal c is saturated after $n^2 + 2n \log \|c\|_\infty$ rounds of the CG procedure. Careful consideration allows us to reduce this quantity to $2n + 2d \log \|c\|_\infty$, where

$$d := \max_{a \in \{0,1\}^n} \left[\max_{x \in P_I} a^T x - \min_{x \in P_I} a^T x \right] \leq n.$$

We note that this "width" parameter d is defined with respect to the integer hull P_I and not P itself. Thus any results making use of this parameter hold for *any* relaxation of P_I in the $[0, 1]^n$ cube.

The parameter d has general upper bound n, and since $P \subseteq [0, 1]^n$ we can use the standard bound $\|c\|_\infty \in O(n \log n)$ (a result that follows from Hadamard's maximum determinant problem, see e.g. [58]). Thus in the worst case this new bound still implies an upper bound of $O(n^2 \log n)$ for $\text{rk}(P)$. However, the reduction of the leading term from n^2 to $2n$ dispenses of the previously automatic n^2 bound, and allows for diminished bounds when a polytope's facet-defining normals are known to have small coefficients. Furthermore, for certain polyhedra this width may be $o(n)$, allowing for bounds lower than previously available. We make use of both these facts to prove bounds for specific polyhedra in Section 3.2.3.

We now proceed to give the main result of the section. Throughout, we assume that $c \geq 0$, which is done without loss of generality by application of suitable coordinate flips. First, we state the main technical component of the result.

Proposition 3.2.5. *Let $P \subseteq [0,1]^n$ be a polytope with*

$$\max_{x \in P_I} a^T x - \min_{x \in P_I} a^T x \leq d$$

for all $a \in \{0,1\}^n$ and some $d \in \mathbb{Z}$, and let $c \in \mathbb{Z}^n_+$ with $c \neq 0$. Then for any $k \in \{0, 1, ..., d\}$ and $t \geq 2n + 2d(\log \|c\|_\infty + 1) - 2k$, the integrality gap of c with respect to $P^{(t)}$ is at most k.

The proof goes by a somewhat non-standard induction involving all of n, k, and $\|c\|_\infty$. We will first give a few supporting results, so that the main proof can better highlight the mechanics of this induction. The first lemma requires that the result of Proposition 3.2.5 holds for small $\|c\|_\infty$ when $k = 0$, then proves that the result is further valid for a larger value of $\|c\|_\infty$, so long as $k = d$.

Lemma 3.2.6. *Set $c \in \mathbb{Z}^n_+$, and let $\|c\|_\infty = C$. Suppose that for each $C' \in \{0, ..., C-1\}$ it holds that any normal $c' \in \mathbb{Z}^n_+$ with $\|c'\|_\infty = C'$ is saturated with respect to $P^{(t')}$ for all $t' \geq 2n + 2d(\log \|c'\|_\infty + 1)$. Then the integrality gap of c with respect to $P^{(t)}$ for $t \geq 2n + 2d(\log \|c\|_\infty + 1) - 2d$ is at most d.*

Proof. We can always write $c = 2c_1 + c_0$ with $c_1 = \lfloor c/2 \rfloor$ and $c_0 \in \{0,1\}^n$. Since $\log \|c_1\|_\infty \leq \log \|c\|_\infty - 1$, by hypothesis we have that c_1 is saturated after

$$2n + 2d(\log \|c_1\|_\infty + 1) \leq 2n + 2d(\log \|c\|_\infty + 1) - 2d$$

rounds of the CG procedure. Select any $t \in \mathbb{Z}$ at least this value.

Now we bound the integrality gap of c with respect to $P^{(t)}$. Let $\bar{x} \in \text{argmax}_{x \in P_I} c_1^T x$. We start by writing

$$\begin{aligned} \max_{x \in P^{(t)}} c^T x - \max_{x \in P_I} c^T x &= \max_{x \in P^{(t)}} (2c_1^T x + c_0^T x) - \max_{x \in P_I} (2c_1^T x + c_0^T x) \\ &\leq \max_{x \in P^{(t)}} 2c_1^T x + \max_{x \in P^{(t)}} c_0^T x - (2c_1^T \bar{x} + c_0^T \bar{x}) \end{aligned} \tag{2}$$

Since c_1 is saturated with respect to $P^{(t)}$, we have that $\max_{x \in P^{(t)}} c_1^T x = c_1^T \bar{x}$. Combining this with (2), we get

$$\begin{aligned} \max_{x \in P^{(t)}} c^T x - \max_{x \in P_I} c^T x &\leq \max_{x \in P^{(t)}} c_0^T x - c_0^T \bar{x} \\ &\leq \max_{x \in P^{(t)}} c_0^T x - \min_{x \in P_I} c_0^T x. \end{aligned} \tag{3}$$

As $\|c_0\|_\infty \leq 1$, we can apply Lemma 3.2.4 to get that c_0 is saturated after $2n \leq t$ rounds of the CG procedure. In particular, we have $\max_{x \in P^{(t)}} c_0^T x = \max_{x \in P_I} c_0^T x$. Thus we can rewrite (3) as

$$\max_{x \in P^{(t)}} c^T x - \max_{x \in P_I} c^T x \leq \max_{x \in P_I} c_0^T x - \min_{x \in P_I} c_0^T x \leq d,$$

as desired. □

The next lemma supposes the results of Lemma 3.2.6 are valid, and uses this to prove that the result of Proposition 3.2.5 holds for any other value of k.

Lemma 3.2.7. *Set $c \in \mathbb{Z}_+^n$, and suppose it holds for any $t' \geq 2n + 2d(\log \|c\|_\infty + 1) - 2d$ that the integrality gap of c with respect to $P^{(t')}$ is at most d. Then if $k \leq d$, for $t \geq 2n + 2d(\log \|c\|_\infty + 1) - 2k$ the integrality gap of c with respect to $P^{(t)}$ is at most k.*

Proof. The proof goes by induction on n and k, where the induction on k goes downward from d to 0. The base case for k is satisfied by assumption, and the base case $n = 1$ holds trivially since a 1-dimensional polytope has rank most 1.

Now suppose $n > 1$ and $k < d$. By the induction hypothesis (on k), the gap of c after $t \geq 2n + 2d(\log \|c\|_\infty + 1) - 2(k+1)$ rounds is at most $k+1$. In fact, we can re-write this quantity to say

$$t \geq 2n + 2d(\log \|c\|_\infty + 1) - 2(k+1) = 2(n-1) + 2d\log_3(2\|c\|_\infty - 1) - 2k.$$

Consider the polytope $P_0 := P \cap \{x : x_1 = 0\}$. As this polytope is $(n-1)$-dimensional, the induction hypothesis (on n) gives that the integrality gap of c with respect to $P_0^{(t)}$ is at most k. Then, letting $\alpha = \max_{x \in P_I} c^T x$, we have $P_0^{(t)} \cap \{x : c^T x = \alpha + k + 1\} = \emptyset$. The same can be said replacing P_0 by $P_1 := P \cap \{x : x_1 = 1\}$. Then the requisite conditions for Lemma 3.2.3 apply, implying that $c^T x \leq \alpha + k$ is valid for $P^{(t+2)}$. Hence the integrality gap of c after any number of rounds exceeding $t + 2 = 2n + 2d\log_3(2\|c\|_\infty - 1) - 2k$ is at most k, as required. □

With these results in hand, we are ready to prove the main technical lemma.

Proof of Proposition 3.2.5. The case of $P_I = \emptyset$ is covered by Lemma 3.2.2, so suppose P_I is nonempty. To begin the proof, note that if $c \in \mathbb{Z}^n_+$ has $\|c\|_\infty = 1$ the claim holds since by Lemma 3.2.4, c is saturated with respect to $P^{(2n)}$.

Moving to the case $\|c\|_\infty = 2$, we see that the setup of Lemma 3.2.6 is satisfied. Hence the desired result holds for this c and $k = d$. But this implies that the assumptions of Lemma 3.2.7 are also satisfied. Hence the desired results holds for c and any $k \leq d$.

Thus for any c with $\|c\|_\infty \leq 2$ and any k the result holds. Hence we can apply the same logic to the case $\|c\|_\infty = 3$, or indeed for any value of $\|c\|_\infty$. Thus the result is given by induction. □

Taking Proposition 3.2.5 and choosing an integrality gap of $k = 0$ immediately gives our main result.

Theorem 3.2.8. *Let $P \subseteq [0,1]^n$ be a polytope and let $Ax \leq b$, $A \in \mathbb{Z}^{m \times n}, b \in \mathbb{Z}^m$ be an inequality description of P_I. Then*

$$\operatorname{rk}(P) \in O(n + d \log \|A\|_\infty).$$

3.2.3 Applying the new bound

In this section, we use Theorem 3.2.8 to prove new bounds for certain classes of polyhedra. In particular, new bounds are given for symmetric polytopes, polytopes with a limited number of integral points, and for certain polyhedra that arise from combinatorial optimization problems. The application of Theorem 3.2.8 is key in many cases: the same proof techniques using previously known results would provide bounds asymptotically worse than those found here.

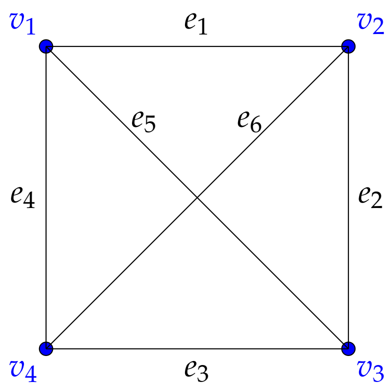
3.2.3.1 Symmetric polyhedra

We first address the case of polyhedra which are invariant under permutation of coordinates. More formally, for $n \in \mathbb{Z}$ we let S_n be the symmetric group on the set $\{1, \ldots, n\}$ (i.e. the set of all permutations of $\{1, \ldots, n\}$). A set $X \subseteq \mathbb{R}^n$ is *symmetric* if

$$\{(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)}) : x \in X\} = X$$

for all $\sigma \in S_n$.

We stress that the definition of symmetry considered here is somewhat strict, as any permutation of the variables in the problem must be allowed. Consider for example the traveling salesman (TSP) polytope, whose extreme points encode all possible tours of a complete graph. Though the underlying graph is invariant under permutation of *nodes*, the TSP polytope is formulated in the space of *edges*. Hence the TSP polytope is not symmetric in the sense defined above - see Figure 11 for an example. A polyhedron that does fit this symmetry definition is the stable



e_1	e_2	e_3	e_4	e_5	e_6
1	1	1	1	0	0
1	0	1	0	1	1
0	1	0	1	1	1

Figure 11: The TSP polytope on 4 vertices is not symmetric with respect to coordinate permutations: The complete graph on 4 nodes is displayed on the left, while the three vertices of the TSP polytope (corresponding to the three possible tours in the graph) are on the right. The polytope is not symmetric since, for example, the permutation that swaps the first two coordinates maps the vertices to (1, 1, 1, 1, 0, 0), (0, 1, 1, 0, 1, 1), (1, 0, 0, 1, 1, 1) - these last two points do not correspond to tours.

set polytope of a complete graph.[1]

We first prove a bound on $\text{rk}(P)$ for all P whose integer hulls are both symmetric and monotone. A set $X \subseteq \mathbb{R}^n$ is *monotone* if $x \in X$ and $0 \leq y \leq x$ implies $y \in X$. In contrast to other proofs in this section, this result does not depend on Theorem 3.2.8, instead requiring only knowledge from previously known results.

Corollary 3.2.9. *Let $P \subseteq [0,1]^n$ be a polytope so that P_I is symmetric and monotone. Then* $\text{rk}(P) \leq 2n$*.*

Proof. Select $\bar{x} \in \text{argmax}\{\mathbf{1}^T x : x \in P_I \cap \{0,1\}^n\}$. We show that P_I is defined by the inequality $\mathbf{1}^T x \leq \mathbf{1}^T \bar{x}$, along with the variable bound inequalities. Specifically, we show $P_I = Q := \{x \in [0,1]^n : \mathbf{1}^T x \leq \mathbf{1}^T \bar{x}\}$, from which we see that each facet-defining vector c of P_I has $\|c\|_1 \leq n$. The result then follows by Lemma 3.2.4.

It is clear that $P_I \subseteq Q$, as any vertex $x \in \{0,1\}^n$ of P_I must have $\mathbf{1}^T x \leq \mathbf{1}^T \bar{x}$ by the selection of $\bar{x}$. Further, we also have that the vertices of Q are all in P_I: all vertices of Q are integral (by the theory of total unimodularity, see e.g. [55]). Any $x \in \{0,1\}^n \cap$

[1]Granted, the stable set problem is not very interesting when the graph is complete, but any stable set polytope will display some *local* symmetry where cliques exist. Thus via Lemma 3.2.1 we can still make statements about $\text{rk}(P)$ using these symmetry results.

Q is also a member of P_I since, by construction of Q, $\mathbf{1}^T x \leq \mathbf{1}^T \bar{x}$. If $\mathbf{1}^T x = \mathbf{1}^T \bar{x}$, then $x \in P_I$ by symmetry, and if $\mathbf{1}^T x < \mathbf{1}^T \bar{x}$, then $x \in P_I$ by monotonicity. $\square$

We now prove CG rank bounds for all symmetric $P \subseteq [0,1]$, monotone or not. The proof is similar in spirit to Corollary 3.2.9: we first give a general representation of such polyhedra, then bound the size of facet-defining vectors using this representation. We proceed by proving the general representation:

Proposition 3.2.10. *Let $P \subseteq [0,1]^n$ be a symmetric, integral polytope, and let $T = \{\mathbf{1}^T x :$ x a vertex of $P\}$. Then P is defined by the system:*

$$\mathbf{0} \leq x \leq \mathbf{1} \tag{4a}$$

$$\min(T) \leq \mathbf{1}^T x \leq \max(T) \tag{4b}$$

$$(u-m)\sum_{i=1}^{m} x_{\sigma(i)} - (m-\ell)\sum_{i=m+1}^{n} x_{\sigma(i)} \leq (u-m)\ell, \tag{4c}$$

for all $\sigma \in S_n$, ℓ, u such that $[\ell, u] \cap T = \{\ell, u\}$, $m \in \mathbb{Z} \cap [\ell+1, u-1]$

Proof. Let Q be the polytope defined by the system (4), so that our task is to show $P = Q$. We show $P \subseteq Q$ by proving that an integral point $x \in \{0,1\}^n$ satisfies system (4) if and only if $\mathbf{1}^T x \in T$; this implies that every vertex of P is a member of Q, and hence $P \subseteq Q$.

We now prove the above claim. If T is such that there are no (4c) inequalities the result is trivial. Thus we suppose this isn't the case, i.e. there exist $\ell, u \in T$ such that $[\ell, u] \cap T = \{\ell, u\}$ and $\mathbb{Z} \cap [\ell+1, u-1] \neq \emptyset$. Select ℓ, u to satisfy these conditions, and choose $x \in \{0,1\}^n$. If $\mathbf{1}^T x \leq \ell$, then for any $m \in \mathbb{Z} \cap [\ell+1, u-1]$ and $\sigma \in S_n$ we have

$$(u-m)\sum_{i=1}^{m} x_{\sigma(i)} - (m-\ell)\sum_{i=m+1}^{n} x_{\sigma(i)} \leq (u-m)\ell - (m-\ell)0 \leq (u-m)\ell,$$

so x satisfies the constraint. Similarly, if $\mathbf{1}^T x \geq u$ then

$$(u-m)\sum_{i=1}^{m} x_{\sigma(i)} - (m-\ell)\sum_{i=m+1}^{n} x_{\sigma(i)} \leq (u-m)m - (m-\ell)(u-m) \leq (u-m)\ell,$$

and the constraint is satisfied by x. However, if $\mathbf{1}^T x \in [\ell+1, u-1]$, set $m = \mathbf{1}^T x$. There exists some $\sigma \in S_n$ so that

$$(u-m)\sum_{i=1}^{m} x_{\sigma(i)} - (m-\ell)\sum_{i=m+1}^{n} x_{\sigma(i)} = (u-m)m - (m-\ell)0 > (u-m)\ell.$$

Hence the integral points $x \in \{0,1\}^n$ that satisfy all constraints of form (4c) for the chosen ℓ, u are those with $\mathbf{1}^T x \in \{1, \dots, \ell, u, \dots, n\}$. Intersecting over all ℓ, u pairs (and adding the requirements of (4b)) implies that the set of integral x satisfying (4) are precisely those with $\mathbf{1}^T x \in T$, and the claim is proven.

To get $Q \subseteq P$, we show that each facet-defining inequality $c^T x \leq \delta$ of P is implied by the system (4). In fact, due to symmetry, it suffices to consider only such inequalities with $c_1 \geq c_2 \geq \cdots \geq c_n$. Select such an inequality, and let $F = P \cap \{x : c^T x = \delta\}$ be the associated facet. Let $k_1 < \cdots < k_t$ be the sequence of numbers k for which F contains an integral point x with $\mathbf{1}^T x = k$.

Clearly, one of the following cases must hold.

1. $c_{k_1} \geq 0$ or $k_1 = 0$.
2. $c_{k_t} \leq 0$ or $k_t = n$.

We consider only the first case, as we have $P \subseteq [0,1]^n$ so the second may be reduced to it via coordinate flips $x_i \mapsto 1 - x_i$ for each $i \in \{1, \dots, n\}$. Further, note that by construction of c, the maximum value of $c^T x$ on integral points with $\mathbf{1}^T x = k$ is $\sum_{i=1}^{k} c_i$, hence we have

$$\delta = \sum_{i=1}^{k_1} c_i = \cdots = \sum_{i=1}^{k_t} c_i. \tag{5}$$

This detail will be used multiple times below.

We show that $c^T x \leq \delta$ follows from (4) in three separate subcases:

1. $c_{k_1+1} \leq 0$ or $k_1 = n$.
2. $k_1 = \max(T)$.

3. $c_{k_1+1} > 0$ and k_1 has a successor in T.

Clearly, any c must fall into one of these categories. For the case $c_{k_1+1} \leq 0$ or $k_1 = n$, (5) and the non-positivity of all coordinates beyond k_1 implies

$$c^T x \leq \sum_{i=1}^{k_1} c_i = \delta$$

when combined with the variable bounds (4a). Hence the desired inequality is implied by (4).

For the case $k_1 = \max(T)$, we apply (4b) to get that $\mathbf{1}^T x \leq k_1$. Thus, combining again with (5) we have

$$\begin{aligned} c^T x &= \sum_{i=1}^{n} (c_i - c_{k_1}) x_i + c_{k_1} \sum_{i=1}^{n} x_i \\ &\leq \sum_{i=1}^{k_1} (c_i - c_{k_1}) + c_{k_1} k_1 \\ &= \sum_{i=1}^{k_1} c_i \\ &= \delta. \end{aligned}$$

So the facet-defining inequality is implied.

Lastly, suppose k_1 has a successor $k_2 \in T$ and $c_{k_1+1} > 0$. In order to keep $\sum_{i=1}^{k_2} c_i \leq \delta$, we must have $\sum_{i=k_1+1}^{k_2} c_i \leq 0$ and hence $c_{k_2} < 0$. Denote by $m \in \{1, \ldots, n\}$ the largest coordinate such that $c_m \geq 0$. We claim that each vertex of F satisfies

$$(k_2 - m) \sum_{i=1}^{m} x_i - (m - k_1) \sum_{i=m+1}^{n} x_i \leq (k_2 - m) k_1. \tag{6}$$

(i.e. the (4c) inequality with σ the identity permutation, $\ell = k_1, u = k_2$, and $m = m$) *with equality*. If this is true, then the valid inequality (6) touches enough vertices of P to be a facet itself. Indeed, as these integral points are the facets of F, the facet induced by (6) is F.

To prove the claim, we first note that since $c_i < 0$ for $i \geq k_2$, we have $\delta = \sum_{i=1}^{k_2} c_i > \sum_{i=1}^{s} c_i$ for any $s \in \{k_2 + 1, \ldots, n\}$. Thus every vertex x of F has $\mathbf{1}^T x \in$

$\{k_1, k_2\}$. Now, select a vertex x of F. If $\mathbf{1}^T x = k_1$, then to keep $c^T x = \delta = \sum_{i=1}^{k_1} c_i$, any index i with $x_i = 1$ must satisfy either $i \leq k_1$ or $c_i \geq c_{k_1}$. Since $x_i < 0$ for all $i > m$, all k_1 non-zero coordinates of x must come from the set $\{1, \ldots, m\}$. So x satisfies (6) with equality.

Lastly, if $\mathbf{1}^T x = k_2$, the value $c^T x$ is maximized only if $x_i = 1$ for $i \in \{1, \ldots, m\}$ (since, by construction, $c_m > c_i$ for $i \in \{m+1, \ldots, n\}$). Thus to keep $c^T x = \delta = \sum_{i=1}^{k_2} c_i$, we must have that $x_i = 1$ for $i \in \{1, \ldots, m\}$, as well as for $k_2 - m$ indices in $\{m+1, ..., n\}$. Then we have

$$(k_2 - m) \sum_{i=1}^{m} x_i - (m - k_1) \sum_{i=m+1}^{n} x_i = (k_2 - m)m - (m - k_1)(k_2 - m) = k_1(k_2 - m),$$

hence x satisfies (6) with equality, as desired.

$\square$

With this result proven, one easily shows the following:

Corollary 3.2.11. *Let $P \subseteq [0,1]^n$ be a polytope so that P_I is symmetric. Then* $\mathrm{rk}(P) \in O(n \log n)$.

Proof. From Proposition 3.2.10, we see that each facet-defining vector c for P_I has $\|c\|_\infty \leq n$. Thus the result follows by application of Theorem 3.2.8. $\square$

Note that the new result of Theorem 3.2.8 is important here. If we instead use the $O(n^2 + n \log \|c\|_\infty)$ bound of [29], we recover only a $O(n^2)$ bound for symmetric polyhedra.

This result also provides insights on the difficulties faced by researchers trying to develop lower bounds for $\mathrm{rk}(P)$, $P \subseteq [0,1]^n$. Until the breakthrough of [64] proving a $\Omega(n^2)$ bound, the best known examples were all drawn from symmetric polyhedra (see [29], [60]). By Corollary 3.2.11, such examples could never hope to show a bound better than $\Omega(n \log n)$.

3.2.3.2 Polyhedra with a limited number of integral points

When an integral polytope contains only a small number of vertices, the facet-defining vectors c are not too complex (i.e. $\|c\|_\infty$ cannot be very large). More specifically, we can show the following:

Proposition 3.2.12. *Let $P \subseteq [0,1]^n$ be an integral polytope with at most k vertices. Any facet-defining vector c of P has $\|c\|_\infty \leq 2^{2^k-1}$.*

Along with Theorem 3.2.8, the above bound also implies bounds for the CG rank. In particular, if $k \in \Theta(\log(n \log n))$ then $\log \|c\|_\infty \approx n \log n$. Thus for $k \in o(\log(n \log n))$, we obtain a bound strictly better than the usual $O(n^2 \log n)$. For example, if $k = \log(\log n)$ then CG rank is bounded by $O(n \log n)$.

The proof of Proposition 3.2.12 uses a fact concerning the growth of coefficients when using the well-known *Fourier-Motzkin elimination* (FME) method (see e.g. [66, Section 12.2] for an overview) for projecting a polyhedron to a lower-dimensional space.

Lemma 3.2.13. *Suppose FME is applied to the system $Ax \leq b$, $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$, recovering the system $A'x \leq b'$, $A \in \mathbb{Z}^{t \times n-1}$, $b \in \mathbb{Z}^t$. Then we may choose A' such that $\|A'\|_\infty \leq 2\,\|A\|_\infty^2$.*

Proof. Suppose, without loss of generality, that we apply FME to A to project out the variable x_n. An inequality $a^T x \leq \beta$ in $A'x \leq b'$ comes from $Ax \leq b$ in one of two forms. In the first case, $A_i = (a_1, \ldots, a_{n-1}, 0)$ for some row A_i of A, and hence $\|a\|_\infty \leq \|A\|_\infty$.

In the second case, $a^T x \leq \beta$ is built from two inequalities $a_{i,1}x_1 + \cdots + a_{i,n}x_n \leq b_i$ and $a_{j,1}x_1 + \cdots + a_{j,n}x_n \geq b_j$ from $Ax \leq b$, with $a_{i,n}, a_{j,n} > 0$. The variable x_n is eliminated by isolating x_n in each inequality, obtaining

$$x_n \leq \frac{b_i}{a_{i,n}} - \frac{a_{i,1}}{a_{i,n}} x_1 - \cdots - \frac{a_{i,n-1}}{a_{i,n}} x_{n-1}$$

and

$$x_n \geq \frac{b_j}{a_{j,n}} - \frac{a_{j,1}}{a_{j,n}} x_1 - \cdots - \frac{a_{j,n-1}}{a_{j,n}} x_{n-1},$$

then combining to obtain

$$\frac{b_j}{a_{j,n}} - \frac{a_{j,1}}{a_{j,n}} x_1 - \cdots - \frac{a_{j,n-1}}{a_{j,n}} x_{n-1} \leq \frac{b_i}{a_{i,n}} - \frac{a_{i,1}}{a_{i,n}} x_1 - \cdots - \frac{a_{i,n-1}}{a_{i,n}} x_{n-1},$$

or equivalently

$$\left(\frac{a_{i,1}}{a_{1,n}} - \frac{a_{j,1}}{a_{j,n}} \right) x_1 + \cdots + \left(\frac{a_{i,n-1}}{a_{1,n}} - \frac{a_{j,n-1}}{a_{j,n}} \right) x_{n-1} \leq \frac{b_i}{a_{i,n}} - \frac{b_j}{a_{j,n}}.$$

To ensure integrality, we can multiply by $a_{i,n}a_{j,n}$ to recover

$$(a_{i,1}a_{j,n} - a_{j,1}a_{i,n})x_1 + \cdots + (a_{i,n-1}a_{j,n} - a_{j,n-1}a_{i,n})x_{n-1} \leq b_i a_{j,n} - b_j a_{i,n}.$$

Clearly, each $(a_{i,k}a_{j,n} - a_{j,k}a_{i,n})$ term has magnitude at most $2 \left\| A \right\|_\infty^2$. □

With this result, we are ready to prove Proposition 3.2.12:

Proof of Proposition 3.2.12. Let $v^1, ..., v^k \in P \cap [0,1]^n$ be the vertices of P. Then $x \in P$ if and only if there exists $\lambda \in \mathbb{R}^k$ such that $(x, \lambda) \in Q$, where the polyhedron Q is defined by

$$\begin{aligned} 0 \leq \lambda_j &\leq 1 && \text{for all } j \in \{1, \ldots, k\} \\ \sum_{j=1}^{k} \lambda_j &= 1 \\ x_i &= \sum_{j=1}^{k} v_i^j \lambda_j && \text{for all } i \in \{1, \ldots, n\}. \end{aligned}$$

So P is the projection of Q into the x coordinates. Thus P may be obtained from Q by k successive applications of FME (with each application projecting out one component of λ). Noting that the largest coefficient in the above system is 1, the result follows by Lemma 3.2.13. □

3.2.3.3 Polyhedra from combinatorial optimization

The inclusion of the parameter d in the bound of Theorem 3.2.8 allows for a reduced bound when considering polytopes whose integral points x have small $\|x\|_1$. This is particularly common for polyhedra related to combinatorial optimization problems.

As an example, consider the TSP polytope associated with the complete graph on v nodes. The dimension of the TSP polytope is governed by the number of edges in the graph, which is $\binom{v}{2} \approx v^2$. However, each integer solution vector contains precisely v ones, implying that $d \leq v$. Thus we have the following:

Corollary 3.2.14. *Let $P \subseteq \mathbb{R}^{\binom{v}{2}}$ be a relaxation of TSP polytope associated with the perfect graph on v nodes. We have* $\mathrm{rk}(P) \in O(v^3 \log v)$.

Note that the results of [29] would only provide a bound of $O(v^4 \log v)$.

3.3 *Separating over the mod-k closure*

We now return to the question of determining the complexity of separating over mod-k cuts. For the special case $k = 2$, it has been shown by [15] that the separation problem is NP-complete, and [52] extends this to the case $P \subseteq [0,1]^n$. Here, we come to the same conclusion for general k. Indeed, the proofs of this section may also be seen as generalizations of the proofs in [15] and [52], with a few extra modifications necessary.

3.3.1 Problem statement

Given a polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, $A \in \mathbb{Z}^{m \times n}, b \in \mathbb{Z}^m$ and positive integer k, a *mod-k cut* is an inequality of the form $\lambda^T Ax \leq \lfloor \lambda^T b \rfloor$ where $\lambda^T A \in \mathbb{Z}^n$ and $\lambda \in \{0, 1/k, ..., (k-1)/k\}^m$. Denote by $P_k(A,b)$ the *mod-k closure* of $Ax \leq b$, i.e. the intersection of P with all mod-k cuts derived from A and b. For an integer $k \geq 2$, a natural question is the following:

Problem: mod-k Closure Separation (kCSEP)

Input: Integral $A \in \mathbb{Z}^{m \times n}, b \in \mathbb{Z}^m$, rational vector $x' \in \mathbb{Q}^n$ with $Ax' \leq b$.

Question: Is $x' \notin P_k(A, b)$? i.e. does there exist mod-k cut $\lambda^T Ax \leq \lfloor \lambda^T b \rfloor$ violated by x'?

3.3.2 Generalizing from the mod-2 case

In proving their mod-2 results, [15] and [52] make reductions from the following NP-complete problems:

Problem: Decoding Linear Codes (DLC)

Input: Matrix $Q \in \{0,1\}^{r \times t}$, vector $d \in \{0,1\}^r$, and positive integer K.

Question: Is there a vector $z \in \{0,1\}^t$ with no more than K entries equal to 1 such that $Qz \equiv d \pmod 2$?

Problem: Weighted Binary Clutter (WBC)

Input: Matrix $Q \in \{0,1\}^{r \times t}$, vector $d \in \{0,1\}^r$, and nonnegative weight vector $w \in \mathbb{Q}^t_{\geq 0}$

Question: Is there a vector $z \in \{0,1\}^t$ such that $Qz \equiv d \pmod 2$ and $w^T z < 1$?

In [15], the authors make a connection between WBC and mod-2 Closure Separation using the following simple observation: A vector x' violates a mod-2 cut if and only if there exists some $\mu \in \{0,1\}^m$ with $\mu^T A \equiv 0 \pmod 2$ and $\mu^T b \equiv 1 \pmod 2$ such that $\mu^T(b - Ax') < 1$ holds. It is easy to see a link between these conditions and the description of WBC, which is exploited in the reduction.

3.3.2.1 From mod-2 to mod-k

A similar set of conditions exist characterizing the existence of a violated mod-k cut for general k. These conditions are well known, and have been expressed e.g. in [16]. The conditions are given below, along with a proof of their validity.

Lemma 3.3.1. *Let $A \in \mathbb{Z}^{m\times n}, b \in \mathbb{Z}^m$ define polyhedron $P = \{x : Ax \leq b\}$, and select $k \in \mathbb{Z}_+$. For any $x' \in P$, there exists a mod-k cut violated by x' if and only if there exists $\mu \in \{0, 1, ..., k-1\}^m$ and $\theta \in \{1, ..., k-1\}$ such that*

$$\mu^T A \equiv 0 \pmod{k}, \qquad \mu^T b \equiv \theta \pmod{k}, \qquad \textit{and} \qquad \theta > \mu^T(b - Ax').$$

Proof. Select $x' \in P$. Suppose $\lambda \in \{0, 1/k, ..., (k-1)/k\}$ defines a mod-k cut violated by x', i.e. we have $\lambda^T A \in \mathbb{Z}^n$ and $\lambda^T Ax' > \lfloor \lambda^T b \rfloor$. Then let $\mu = k\lambda$ and $\theta = k(\lambda^T b - \lfloor \lambda^T b \rfloor)$. We clearly have $\mu^T A \equiv 0 \pmod{k}$, as well as

$$\mu^T b = \theta + k \left\lfloor \lambda^T b \right\rfloor \equiv \theta \pmod{k}$$

and

$$\begin{aligned} \mu^T(b - Ax') &= k\lambda^T b - k\lambda^T Ax' \\ &= \theta + k \left\lfloor \lambda^T b \right\rfloor - k\lambda^T Ax' \\ &< \theta + k \left\lfloor \lambda^T b \right\rfloor - k \left\lfloor \lambda^T b \right\rfloor \\ &= \theta. \end{aligned}$$

For the other direction, suppose there exists $\mu \in \{0, 1, ..., k-1\}^m$ and $\theta \in \{1, ..., k-1\}$ such that $\mu^T A \equiv 0 \pmod{k}, \mu^T b \equiv \theta \pmod{k}$, and $\theta > \mu^T(b - Ax')$. Then let $\lambda = \mu/k$, and we clearly have $\lambda^T A \in \mathbb{Z}^n$. Further, $u^T b \equiv \theta \pmod{k}$ if and only if $\mu^T b = \alpha k + \theta$ for some $\alpha \in \mathbb{Z}$. Thus we have

$$\lambda^T b - \left\lfloor \lambda^T b \right\rfloor = \alpha + \frac{\theta}{k} - \left\lfloor \alpha + \frac{\theta}{k} \right\rfloor = \alpha + \frac{\theta}{k} - \left(\alpha + \left\lfloor \frac{\theta}{k} \right\rfloor \right) = \frac{\theta}{k},$$

and hence

$$\lambda^T Ax' > \lambda^T b - \theta/k = \lambda^T b - \left(\lambda^T b - \left\lfloor \lambda^T b \right\rfloor \right) = \left\lfloor \lambda^T b \right\rfloor,$$

finishing the proof. □

Mirroring the work of [15], we will prove NP-completeness of kCSEP by connecting these conditions with a generalization of WBC. After some consideration, we formulate the following problem as the correct generalization:

Problem: Weighted k-ary Clutter (WkC)

Input: Matrix $Q \in \{0,1\}^{r \times t}$, vector $d \in \{0,1\}^r$, and nonnegative weight vector $w \in \mathbb{Q}^t_{\geq 0}$.

Question: Is there a vector $z \in \{0,1,...,k-1\}^t$ and integer $\alpha \in \{1,...,k-1\}$ such that $Qz + \alpha d \equiv 0 \pmod{k}$ and $w^T z < \alpha$?

Of course, we must first establish NP-completeness of WkC before using it in a reduction for kCSEP. We will do so in a coming section.

3.3.3 Linear codes modulo k

Our ultimate aim is to show that kCSEP is NP-complete for any k, even if we restrict $\{x : Ax \leq b\} \in [0,1]^n$. This will be done via a series of reductions. First, we use the classical NP-complete problem Three-Dimensional Matching to prove that the following generalization of DLC is NP-complete.

Problem: Decoding Linear Codes Modulo k (DLCk)

Input: $Q \in \{0,1\}^{r \times t}$, $d \in \{0,1\}^r$, $K \in \mathbb{Z}_+$.

Question: Is there $z \in \{0,1,...,k-1\}^t$ and $\alpha \in \{1,...,k-1\}$ such that $z^T \mathbf{1} \leq \alpha K$ and $Qz + \alpha d \equiv 0 \pmod{k}$?

This represents most of the work, as the reduction from DLCk to WkC is quite simple, and the reductions from WkC to kCSEP are merely straightforward modifications of the proofs found in [15] and [52].

To show that DLCk is NP-complete, we follow the lead of [7], who were the first to show that DLC is NP-complete. Their reduction goes from the Three-Dimensional Matching problem, which we describe here.

Problem: Three-Dimensional Matching (3DM)

Input: Finite set T, set $U \subseteq T^3$.

Question: Is there a set $W \subseteq U$ such that $|W| = |T|$ and no two elements of W agree on any coordinate?

3DM is a well known problem, and in fact is one of Karp's 21 NP-complete problems ([44]). The following is an example of a 3DM instance (from [7]):

$$T = \{1,2,3,4\}, \qquad U = \{(1,2,1),(1,3,2),(2,1,4),(2,2,3),(3,1,1),(4,4,4)\}.$$

In this case, the answer to the question is "yes" (take the second, fourth, fifth, and sixth points in U).

3DM is equivalent to solving an integer linear system with certain requirements, as we illustrate below. Consider the matrix $A(T,U) \in \{0,1\}^{3|T|\times|U|}$ constructed in the following manner: Suppose $u^1, ..., u^n$ are the elements of U. For $i \in \{1, ..., |T|\}$ and $k \in \{1,2,3\}$ set

$$A(T,U)_{(k-1)|T|+i,j} = \begin{cases} 1 & \text{if } u^j_k = i, \\ 0 & \text{otherwise.} \end{cases}$$

Each column corresponds to an element u^j of U, and each group of $|T|$ rows codes the entries of u^j. Below is the matrix arising from our previous example.

$$A(T,U) = \left[\begin{array}{cccccc} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \hdashline 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \hdashline 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right]$$

By construction, a 3DM instance has a solution if and only if there is some $y \in \{0,1\}^{|U|}$ with $y^T\mathbf{1} = |W| = |T|$ and $A(T,U)y = \mathbf{1}$ (the desired elements of U correspond to the positive entries in y). Note that every column of $A(T,U)$ has exactly 3 entries equal to one, and in particular $\mathbf{1}^T A(T,U) = 3 \cdot \mathbf{1}$.

3.3.4 The mod-3 case

By adapting the proof of [7], we can use 3DM to show NP-completeness of DLCk for any k. For illustrative purposes, we begin by proving this only in the case of $k = 3$. For reference, this special case of DLCk is described here:

Problem: Decoding Linear Codes modulo 3 (DLC3)

Input: $Q \in \{0,1\}^{r\times t}$, $d \in \{0,1\}^r$, $K \in \mathbb{Z}_+$.

Question: Is there $z \in \{0,1,2\}^t$ and $\alpha \in \{1,2\}$ such that $z^T\mathbf{1} \leq \alpha K$ and $Qz + \alpha d \equiv 0 \pmod 3$?

The completeness proof follows by a reduction from 3DM. The particular reduction differs slightly depending on the parity of $|T|$. We handle the two cases separately, using the two proofs to illustrate different ingredients of the general proof in Section 3.3.5.

The next lemma covers the simplest case, and illustrates the main reduction technique. Given a 3DM instance, we create a DLC3 instance with $Q = A(T,U)$, $d = \mathbf{1}$, and $K \approx |T|/2$. Going from a 3DM solution to a DLC3 solution is simple, but the other direction is less obvious. As we will see, for any solution z, α to the DLC3 instance, the value of α gives simultaneous upper and lower bounds on the value $\|A(T,U)z\|_1$. The bounds will be such that they can only be satisfied when $\alpha = 2$ (or $\alpha = k-1$ in the general case), which means that $Qz + \alpha d \equiv 0 \pmod 3$ reduces to $A(T,U)z \equiv \mathbf{1} \pmod 3$. This, combined with the upper bound on $\|A(T,U)z\|_1$, will imply that in fact $A(T,U)z = \mathbf{1}$, and hence 3DM also has a solution. The formal proof follows.

Lemma 3.3.2. *Let T, U be an instance of 3DM with $|T|$ even. Create an instance of DLC3 defined by $Q = A(T,U)$, $d = \mathbf{1}$, and $K = |T|/2$. Then the 3DM instance has a solution if and only if the DLC3 instance has a solution.*

Proof. One direction is immediate: if 3DM has a solution, i.e. there exists $y \in \{0,1\}^{|U|}$ with $y^T\mathbf{1} = |T|$ and $A(T,U)y = \mathbf{1}$, then setting $z = y$ and $\alpha = 2$ solves the DLC3 instance. For the other direction, suppose the DLC3 instance has a solution. We claim that the solution must have $\alpha = 2$, for if $\alpha = 1$ we have that $\mathbf{1}^T z \leq |T|/2$ which implies $\mathbf{1}^T A(T,U)z = (3 \cdot \mathbf{1})z \leq 3|T|/2$. But $\alpha = 1$ also implies $A(T,U)z \equiv 2 \cdot \mathbf{1} \pmod 3$, and so as $A(T,U)z$ has dimension $3|T|$ this implies $\mathbf{1}^T A(T,U)z \geq 6|T|$, a contradiction.

So the DLC3 solution has $\alpha = 2$, implying that $\mathbf{1}^T z \leq 2(|T|/2) = |T|$ and $A(T,U)z \equiv \mathbf{1} \pmod 3$. The former implies $\mathbf{1}^T A(T,U)z = (3 \cdot \mathbf{1})z \leq 3|T|$, while the latter implies $\mathbf{1}^T A(T,U)z \geq 3|T|$. Thus $\mathbf{1}^T A(T,U)z = 3|T|$ and $\mathbf{1}^T z = |T|$.

Further, the only way to satisfy $A(T,U)^T z \equiv \mathbf{1} \pmod 3$ and $\mathbf{1}^T A(T,U)z = 3|T|$ is to have $A(T,U)z = \mathbf{1}$, which also implies $z \in \{0,1\}$. Thus setting $y = z$ solves the 3DM instance. □

In the case that $|T|$ is odd, we must modify the above slightly since $|T|/2$ is not an integer. The fix is to set $K = (|T|+1)/2$ and add an extra row and column to Q. We do this in such a way as to force any z solving DLC3 to have a 1 in the index corresponding to the extra column (without affecting the rest of the matrix).

Lemma 3.3.3. *Let T, U be an instance of 3DM with $|T| \geq 2$. Create an instance of DLC3 defined by*

$$Q = \begin{bmatrix} A(T,U) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}, \qquad d = \mathbf{1}, \qquad \textit{and} \qquad K = \frac{|T|+1}{2}$$

Then the 3DM instance has a solution if and only if the DLC3 instance has a solution.

Proof. If the 3DM instance has a solution, setting $\alpha = 2$ and $z = (y^T, 1)^T$ solves the DLC3 instance. If the DLC3 instance has a solution, then we can show as in Lemma 3.3.2 that we have $\alpha = 2$ and further the final component of z (corresponding to the extra column in Q) must be equal to 1. Let $z' \in \{0,1,2\}^{|U|}$ be equal to z on its first $|U|$ components (i.e. ignoring the final component corresponding to the extra column in Q). We have $\mathbf{1}^T z \leq |T| + 1$ and hence $\mathbf{1}^T z' \leq |T|$, plus $Qz \equiv \mathbf{1} \pmod 3$ and so $A(T,U)z' \equiv \mathbf{1} \pmod 3$. Proceeding as in the proof of Lemma 3.3.2, we see that setting $y = z'$ solves the 3DM instance. □

The previous two lemmas combine to give the main result.

Theorem 3.3.4. *DLC3 is NP-complete.*

Proof. Let T, U be an instance of 3DM. The reduction to DLC3 depends on the parity of $|T|$, which can be determined in polynomial time. If $|T|$ is even, use the reduction from Lemma 3.3.2. If $|T|$ is odd, use the reduction from Lemma 3.3.3. □

3.3.5 The general case

We are now ready to prove that DLCk is NP-complete for any k, which we will do by generalizing Lemma 3.3.3's proof.

Theorem 3.3.5. *DLCk is NP-complete.*

Proof. Let T, U describe an instance of 3DM. For the DLCk instance, let $m = |T| \pmod{k-1}$ and set

$$Q = \begin{bmatrix} A(T,U) & \mathbf{0} \\ \mathbf{0} & I_{k-1-m} \end{bmatrix}, \qquad d = \mathbf{1}, \qquad \text{and} \qquad K = \frac{|T| + k - 1 - m}{k-1}.$$

If the 3DM instance has a solution y, then setting $\alpha = k-1$ and $z = (y^T, \mathbf{1}^T)^T$ solves the DLCk instance. If the DLCk instance has a solution, we will show that we must have $\alpha = k-1$. Let $z' \in \{0, 1, ..., k-1\}^{|U|}$ be equal to z on its first $|U|$ coordinates (ignoring the coordinates corresponding to the identity matrix in the bottom-right of Q), and z'' contain the remaining coordinates of z. For any value of α, we have $A(T,U)z' \equiv (k-\alpha)\mathbf{1} \pmod{k}$, implying that $\mathbf{1}^T A(T,U)z' \geq 3|T|(k-\alpha)$. Further, we have $\mathbf{1}^T z \leq \alpha \frac{|T|+k-1-m}{k-1}$, and since every entry of z'' must be positive, we get

$$\mathbf{1}^T z' \leq \alpha \frac{|T| + k - 1 - m}{k-1} - \mathbf{1}^T z'' \leq \alpha \frac{|T| + k - 1 - m}{k-1} - (k-1-m).$$

This implies that $\mathbf{1}^T A(T,U)z' = (3 \cdot \mathbf{1})^T z' \leq 3\alpha \frac{|T|+k-1-m}{k-1} - (k-1-m)$. For $\mathbf{1}^T A(T,U)z'$ to satisfy both bounds simultaneously, we need

$$\begin{aligned} & \alpha \left(\frac{|T| + k - 1 - m}{k-1} \right) - k + 1 + m \geq |T|(k-\alpha) \\ \Leftrightarrow \quad & \alpha \left(\frac{|T|(k-1) + |T| + k - 1 - m}{k-1} \right) \geq (|T|+1)k - 1 - m \\ \Leftrightarrow \quad & \alpha \left(\frac{(|T|+1)k - 1 - m}{k-1} \right) \geq (|T|+1)k - 1 - m \\ \Leftrightarrow \quad & \alpha \geq k-1, \end{aligned}$$

and hence $\alpha = k - 1$. This implies that $z'' = \mathbf{1}$, so then $A(T,U)z' \equiv \mathbf{1} \pmod{k}$ and $\mathbf{1}^T z' \leq |T|$. Hence (proceeding as in the proof of Lemma 3.3.2) we know that setting $y = z'$ solves the 3DM instance. □

3.3.6 Separation for the mod-k closure

We have now established that DLCk is NP-complete for any k. Given this, it is straightforward to reduce to WkC, which is the problem we will use to show completeness of kCSEP. In fact, WkC stays NP-complete even when we restrict $w \in [0,1]^t$, as the reduction will show. This fact will be used later.

Lemma 3.3.6. *WkC is NP-complete.*

Proof. Let Q, d, and K be an instance of DLCk. For the WkC instance, use Q and d unmodified, and set

$$w = \frac{1}{K + \frac{1}{k-1}} \cdot \mathbf{1}.$$

Suppose that z, α satisfy $Qz + \alpha d \equiv 0 \pmod{k}$. If $z^T\mathbf{1} \leq \alpha K$ then we have

$$\alpha \geq \frac{1}{K} z^T \mathbf{1} > \frac{1}{K + \frac{1}{k-1}} z^T \mathbf{1} = w^T z,$$

and if $w^T z < \alpha$ then

$$z^T\mathbf{1} < \left(K + \frac{1}{k-1}\right)\alpha = K\alpha + \frac{\alpha}{k-1} \leq K\alpha + 1,$$

and hence $z^T\mathbf{1} \leq K\alpha$. □

We now have all the ingredients necessary to show that kCSEP is NP-complete for any k, both in the general case and when $P \subseteq [0,1]^n$. The first proof is essentially a copy of [15], swapping ks for 2s.

Theorem 3.3.7. *The mod-k Closure Separation problem is NP-complete.*

Proof. The problem is clearly in NP. To show completeness, we reduce from WkC. Let Q, d, and w give an instance of WkC. Define

$$A = \begin{bmatrix} Q^T & \\ & kI_{t+1} \\ d^T & \end{bmatrix}, \quad b = \begin{bmatrix} k\mathbf{1}_{t\times 1} \\ \\ 1 \end{bmatrix}, \quad x' = \begin{bmatrix} \mathbf{0}_{r\times 1} \\ \\ \mathbf{1}_{t\times 1} - \frac{1}{k}w \\ \\ \frac{1}{k} \end{bmatrix},$$

and notice that $b - Ax' = (w^T, 0)^T$, and in particular $x' \in P := \{x : Ax \leq b\}$.

Suppose that x' violates some mod-k cut, and hence by Lemma 3.3.1 there exists $\mu \in \{0, 1, ..., k-1\}^{t+1}$ and $\theta \in \{1, ..., k-1\}$ so that $\mu^T A \equiv 0 \pmod k$, $\mu^T b \equiv \theta \pmod k$ and $\theta > \mu^T(b - Ax')$. By construction of b, we must have $\mu_{t+1} = \theta$. Thus, by choosing $z = (\mu_1, ..., \mu_t)$ and $\alpha = \theta = \mu_{t+1}$, we also have $Qz + \alpha d \equiv 0 \pmod k$ and $w^T z = \mu^T(b - Ax') < \theta = \alpha$. Then the W$k$C problem has a solution.

Conversely, suppose there exists $z \in \{0, 1, ..., k-1\}^t$ and $\alpha \in \{1, ..., k-1\}$ such that $Qz + \alpha d \equiv 0 \pmod k$ and $w^T z < \alpha$. Then, setting $\mu = (z^T, \alpha)^T$ and $\theta = \alpha$, we have $\mu^T A \equiv 0 \pmod k$, $\mu^T b \equiv \theta \pmod k$, and $\theta = \alpha > w^T z = \mu^T(b - Ax')$.

Thus the WkC instance has a solution if and only if the kCSEP instance has a solution, completing the proof. □

A similar variation to the proof of [52] shows that kCSEP remains NP-complete, even if the polyhedron of interest is contained in the 0-1 hypercube. We present this proof below.

Theorem 3.3.8. *The mod-k Closure Separation problem is NP-complete, even if we restrict* $P := \{x : Ax \leq b\} \subseteq [0, 1]^n$.

Proof. We again reduce from WkC, this time taking care to assure that the polytope we construct is contained in the cube. Further, we use the special case of WkC where we restrict $w \in [0, 1]^t$, which is still NP-complete as shown in the proof of

Lemma 3.3.6. Let Q, d, and w define an instance of WkC with $w \in [0,1]^t$, and define

$$A = \begin{bmatrix} \begin{matrix} Q^T \\ d^T \end{matrix} & kI_{t+1} \\ kI_r & \mathbf{0}_{r\times t+1} \\ -kI_r & \mathbf{0}_{r\times t+1} \\ \mathbf{0}_{t+1\times r} & -kI_{t+1} \end{bmatrix}, \quad b = \begin{bmatrix} k\mathbf{1}_{t\times 1} \\ 1 \\ k\mathbf{1}_{r\times 1} \\ \mathbf{0}_{r\times 1} \\ \mathbf{0}_{t+1\times 1} \end{bmatrix}, \quad x' = \begin{bmatrix} \mathbf{0}_{r\times 1} \\ \mathbf{1}_{t\times 1} - \frac{1}{k}w \\ \frac{1}{k} \end{bmatrix}.$$

Here, we have that $b - Ax' = (w^T, 0, k\mathbf{1}_{1\times r}, \mathbf{0}_{1\times r}, k\mathbf{1}_{1\times t} - w^T, 1)^T \geq 0$, and hence $x' \in P$. Further, one can easily verify that $P \subseteq [0,1]^{r+t+1}$.

Suppose that x' violates some mod-k cut, and so by Lemma 3.3.1 there exists $\mu \in \{0, 1, ..., k-1\}^{2(t+1)+2r}$ and $\theta \in \{1, ..., k-1\}$ so that $\mu^T A \equiv 0 \pmod{k}$, $\mu^T b \equiv \theta \pmod{k}$ and $\theta > \mu^T(b - Ax')$. By the construction of b it must be that $\theta = \mu_{t+1}$. Thus, by choosing $z = (\mu_1, ..., \mu_t)$ and $\alpha = \theta = \mu_{t+1}$, we clearly have $Qz + \alpha d \equiv 0 \pmod{k}$. Further, $w^T z$ is equal to the product $\mu^T(b - Ax')$ restricted to the first t indices, so by the nonnegativity of μ and $(b - Ax')$ we have $w^T z \leq \mu^T(b - Ax') < \theta = \alpha$. So W$k$C has a solution.

Next, assume that there exists $z \in \{0, 1, ..., k-1\}^t$, $\alpha \in \{1, ..., k-1\}$ so that $Qz + \alpha d \equiv 0 \pmod{k}$ and $w^T z < \alpha$. By choosing $(\mu_1, ..., \mu_t) = z$, $\mu_{t+1} = \alpha$, and $\theta = \alpha$ we get $\mu^T A \equiv 0 \pmod{k}$ and $\mu^T b \equiv \theta \pmod{k}$ irrespective of the selection of the other indices of μ. Further, if we choose $\mu_i = 0$ for $i > t+1$, we get $\mu^T(b - Ax') = w^T z < \alpha = \theta$. Thus x' violates some mod-k cut.

Then the WkC instance has a solution if and only if the kCSEP instance has a solution, completing the proof. □

3.4 *Remarks and future work*

Recently, [48] has indicated that the separation problem for CG cuts is NP-complete, even in the case that $P \subseteq [0,1]^n$, using techniques different from those used here for mod-k cuts. This essentially completes the picture of the work started by [52] and continued here in Section 3.3.

The work in Section 3.2 leaves open the important question of closing the $\log(n)$ gap between upper and lower bounds for CG rank when $P \subseteq [0,1]^n$. In the mean time, there is room for more work along the lines of that found in Section 3.2.3: other classes of polyhedra could obtain new CG rank bound by applying these methods.

It may be interesting to apply more attention to reducing bounds for polyhedra with a limited number of integral points. Other methods for bounding $\|c\|_\infty$ may be more fruitful. Alternatively, strategies different from those used in Section 3.2.3 may also apply. One idea is to try to relate the number of integral points to the "pitch" value defined in [6], then use their other results to provide a new bound.

CHAPTER IV

OPPORTUNISTIC REPLENISHMENTS IN INVENTORY MODELING

4.1 Introduction

Inventory control is one of the classic applications of optimization theory, and popular models such as EOQ (see [39]) and the newsvendor model (see [3]) have been in use for decades. Different models vary in assumptions and system dynamics, but the motivation remains constant: how much of a good should a company keep in stock in order to best meet the needs of the business?

In this chapter, we study management of an inventory system where replenishments do not occur according to a known schedule. Instead, replenishment opportunities arise from time to time due to influences outside of our control. To see where this may apply in the real world, consider the following: Imagine the fulfillment center of an online retailer, which is in charge of picking and shipping items ordered by the customers. Picking is a manual operation, i.e. human workers must be hired for the task. Staffing decisions are made ahead of time based on anticipated order volumes. Periodically, forecasts may overestimate the number of workers necessary, such that a large portion of the workforce is left idle.

Most retailers carry a number of items that are consistently ordered in high volumes. To give these idle workers a value-adding task, it is decided to allow them to pick these high-volume items and prep them for shipment, in anticipation of a likely future order. Maintaining this prepackaged inventory is analogous to the situation outlined above, as these prepackaging opportunities only arise when an order forecast significantly overestimates staffing needs. Indeed, this work was inspired by a large online retailer facing exactly this scenario. A main contribution of this work is the development of a model to be deployed in such a situation.

Furthermore, most retailers store a vast amount of information on the items they sell and the orders made for them. Given the recent explosion in interest in machine learning and big data, it makes sense to ask how one can leverage this information to help drive inventory decisions and ultimately increase profits. A second focus of this work is exploring how data-augmented decisions can help an inventory controller make better decisions and improve their bottom line.

4.1.1 Outline

This chapter is structured as follows: we begin in Section 4.2 by reviewing some preliminaries on stochastic processes. In Section 4.3, we develop a basic model for the inventory problem with opportunistic replenishments. The assumptions underlying this model are too restrictive to apply in many cases, but its simplicity allows for some interesting analyses. In particular, we are able to quantify the cost of uncertainty in its parameters (Section 4.3.2).

Next, in Section 4.4, we present an extended model which is more fit for application in real-life scenarios. The model is very similar in spirit to the first, but we allow a broader set of assumptions. We derive an expression for expected costs under this scenario, and discuss how to use the model to determine optimal inventory levels.

Section 4.5 explores decision making in the case where multiple items require replenishment while competing for shared resources. We show that the problem may be modeled as a binary integer program. A computational study explores the feasibility of using such models in practice.

Lastly, in Section 4.6 we use real data from an online retailer to test the usefulness of the model in practice. After establishing that the model can be used successfully with access to good order predictions, we turn the the task of predicting order rates from historical data. These data-driven predictions are then fed to the optimization model. We find that certain machine learning algorithms outperform classical forecasting techniques in terms of achieving higher revenue.

4.2 Preliminaries

Analyzing our new inventory models will require some background in stochastic processes, which we cover here. The results we describe are standard, and can be found for example in [63] or [61].

The *exponential distribution* is a well-studied continuous probability distribution, characterized by the density function

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

for some $\lambda > 0$ (often referred to as the *rate*). The structure of the density function makes the exponential distribution particularly easy to analyze. Due to this and other factors, this distribution is a popular choice in stochastic modeling.

A few properties of the exponential distribution are relevant to us. First, if X is an exponential random variable with rate λ, then it is easy to show that $\mathbb{E}[X] = \frac{1}{\lambda}$. Second, and more interesting, is the so-called *memorylessness property* of the exponential distribution, which states that $\mathbb{P}[X > s + t | X > s] = \mathbb{P}[X > t]$. In words, this property implies that the occurrence of past events has no impact on the timing of the next event. In particular, a long passage of time since the last event does not imply that the next event is imminent.

The *Poisson distribution* is a discrete probability distribution over the nonnegative integers. In modeling, this distribution is often used to count the number of events that occur in a certain time period. The distribution is governed by a single parameter $\mu > 0$. If X is a random variable with a Poisson distribution, then $\mathbb{E}[X] = \mu$. Further, its probability mass function is given by

$$\mathbb{P}[X = k] = \frac{\mu^k e^{-\mu}}{k!},$$

and the cumulative distribution function is

$$\mathbb{P}[X \leq k] = \frac{\Gamma(\lfloor k+1 \rfloor, \mu)}{\lfloor k \rfloor !}. \tag{7}$$

Appearing in the distribution function is the well-studied gamma function $\Gamma(\cdot)$, which is defined as

$$\Gamma(s) = \int_0^\infty t^{s-1} e^{-t} \, dt.$$

Specifically, the form found above is known as the upper incomplete gamma function $\Gamma(\cdot, \cdot)$, given by

$$\Gamma(s, x) = \int_x^\infty t^{s-1} e^{-t} \, dt. \tag{8}$$

The (one-dimensional) *Poisson process* is a stochastic process that is often used in probabilistic models to govern occurrences of a particular event over time. The Poisson process is a special type of *counting process*, a stochastic process $\{N(t), t \geq 0\}$ which satisfies

1. $N(t)$ is a positive integer for all $t \geq 0$.

2. If $s < t$, then $N(s) \leq N(t)$.

In modeling, the quantity $N(t)$ usually represents the number of events that have occurred since the start of the process at time 0. As a natural extension of this notation, we let $N(s, t)$ represent the number of events that have occurred on the time interval $(s, t]$.

In order to be a Poisson process, a counting process must also satisfy a few extra conditions. Specifically, for some $\lambda > 0$, we have:

1. N(0)=0.
2. If $(s_1, t_1] \cap (s_2, t_2] = \emptyset$, then $N(s_1, t_1)$ and $N(s_2, t_2)$ are independent random variables (the *independent increments* property).
3. For all $s, t \geq 0$, the quantity $N(t+s) - N(s)$ follows a Poisson distribution with parameter $\mu = \lambda t$.

We refer to λ as the *rate* or *intensity* of the process.

When examining a Poisson process, we are often interested in more than just the number of events within an interval. Sometimes we are interested in the times at which events occur. With this in mind, let X^j be the time of the jth event. We call each X^j an *arrival time*. The arrival times are connected to the function N by the following relation:

Lemma 4.2.1. *Consider a Poisson process with* $\{N(t), t \geq 0\}$ *and* $\{X^j, j \in \mathbb{Z}_+\}$ *as defined above. Then*

$$\mathbb{P}\left[X^j > t\right] = \mathbb{P}\left[N(t) < j-1\right] = \sum_{k=0}^{j-1} \mathbb{P}\left[N(t) = k\right]$$

The relation holds because the events $\{X^j > t\}$ and $\{N(t) < j-1\}$ are equivalent: at least j events have occurred by time t (i.e. $N(t) \geq j$) if and only if the jth event occurred before (or precisely at) time t (that is, $X^j \leq t$).

Define $Y^j := X^j - X^{j-1}$ for $j \geq 1$ (allow $X^0 = 0$). Then Y^j is the amount of time between events j and $j-1$. We call the times between successive events the *interarrival times* of the process. It is well known that for Poisson process, interarrival times are governed by an exponential distribution with the same rate λ. Hence, we also have $X^j = Y^1 + \cdots + Y^j$, and so by the linearity of expectation we know $\mathbb{E}\left[X^j\right] = \frac{j}{\lambda}$.

The interarrival exponential distribution is convenient for analyzing the Poisson process, but it highlights one of its modeling weaknesses. It implies that, at any point during the process, the expected waiting time until the next event is always the same. This may not be desirable. For example, consider modeling arrivals of customers to a restaurant throughout the day. One would expect that interarrival times are shorter during the lunch and dinner rushes than at other times of the day. Thus the standard Poisson process may not be a good choice for modeling this scenario.

This difficulty is remedied by the *nonhomogeneous Poisson process*, which behaves like the classic Poisson process but allow us to replace the rate $\lambda > 0$ by a positive-valued *rate function* $\{\lambda(t), t \geq 0\}$. This allows us to increase or decrease the likelihood of events over the course of the process. An important function for analyzing a nonhomogeneous Poisson processes is given by

$$m(t) = \int_0^t \lambda(s)\, ds. \tag{9}$$

Indeed, one can show the following:

Lemma 4.2.2. *Consider a nonhomogenous Poisson process with the functions N, m as defined above. For any $t, s \geq 0$, the random variable $N(t+s) - N(s)$ is a Poisson random variable with parameter $m(t+s) - m(s)$.*

In particular, $N(t)$ is a Poisson random variable with parameter $m(t)$.

4.3 A model for opportunistic replenishments

In this section, we outline the components of our inventory model, and discuss optimal replenishment levels in various scenarios.

4.3.1 The base model

In the simplest case, we suppose that inventory is kept for only one type of item, and we currently have none of this item in stock. At time 0, we make a replenishment decision by choosing a number $n \in \mathbb{Z}_+$ of items we want to hold in inventory. We assume the replenishment occurs instantaneously, and all n items are immediately available to satisfy potential orders from our customers. Sales are lost if an order is received but no inventory exists.

The model consists of three parameters:

1. *Sales profit* $s > 0$: The sale of an item ordered from inventory gains s units of profit.

2. *Holding cost* $h > 0$: Each item accrues holding costs of h units per time period while being held in stock.

3. *Order rate* $\lambda > 0$: Orders are received from our customers according to a Poisson process with constant rate parameter λ.

For now, we assume the replenishment at time 0 is the only replenishment opportunity we will ever have (or future replenishments occur so far in the future as to be irrelevant). If we let X^j denote the time until the jth customer order, then for $j \leq n$ the profit generated from the jth item is $s - hX^j$. Our goal is to find $n \in \mathbb{Z}$ which maximizes the expected total profit

$$v(n) := \mathbb{E}\left[\sum_{j=1}^{n} s - hX^j\right]$$

We saw in Section 4.2 that $\mathbb{E}\left[X^j\right] = \frac{j}{\lambda}$, allowing us to calculate

$$v(n) = \mathbb{E}\left[\sum_{j=1}^{n} s - hX^j\right] = ns - \frac{hn(n+1)}{2\lambda} = n\left(s - \frac{h}{2\lambda}\right) - n^2\frac{h}{2\lambda}. \tag{10}$$

From this, it is easy to show that the optimal replenishment level $n^* \in \mathbb{Z}$ is given by

$$n^* = \left\lfloor \frac{s\lambda}{h} \right\rfloor. \tag{11}$$

4.3.2 Stochastic order rates

As we can see, the model is easy to solve when all parameters are known. In many cases, firms will be able determine reasonable values for s and h in their applications. However, predicting customer activity is a different matter altogether, making it tricky to accurately estimate λ.

It may be more reasonable to relax the requirement of knowing λ beforehand. To that end, we assume that the rate is determined by the realization of a positive-valued random variable Λ, whose distribution is known to us. Once realized, the rate value stays fixed and does not change over time.

We must make our replenishment decision before the true value of Λ is revealed. We will denote the time until the jth order as $X^j(\Lambda)$ to emphasize its dependency on the random arrival rate. Then our task becomes to find $n \in \mathbb{Z}$ that maximizes

$$\mathbb{E}\left[\sum_{j=1}^{n} s - hX^j(\Lambda)\right] = \sum_{j=1}^{n} s - h\mathbb{E}\left[X^j(\Lambda)\right].$$

This quantity is strictly decreasing in j, thus it suffices to find the largest j for which the term inside the sum is positive. We have

$$\begin{aligned} s - h\mathbb{E}\left[X^j(\Lambda)\right] &= s - h\mathbb{E}\left[\mathbb{E}\left[X^j(\Lambda)|\Lambda\right]\right] \\ &= s - h\mathbb{E}\left[\frac{j}{\Lambda}\right] \\ &= s - hj\mathbb{E}\left[\Lambda^{-1}\right]. \end{aligned}$$

This quantity is positive whenever $j < \frac{s}{h\mathbb{E}\left[\Lambda^{-1}\right]}$, so the best replenishment level n^* is

$$n^* = \left\lfloor \frac{s}{h\mathbb{E}\left[\Lambda^{-1}\right]} \right\rfloor.$$

Comparing this to our previous results, we see that the value $\frac{1}{\mathbb{E}\left[\Lambda^{-1}\right]}$ appears to be taking the place of λ in our value equations. As such, the optimal policy behaves as though we "guess" a true rate of $\frac{1}{\mathbb{E}\left[\Lambda^{-1}\right]}$, and then make our decisions according to the analysis of Section 4.3.1 (we will continue to use this ”rate guessing” interpretation in what follows). It is interesting that the ”correct” guess is $\frac{1}{\mathbb{E}\left[\Lambda^{-1}\right]}$ instead of $\mathbb{E}\left[\Lambda\right]$, as one might initially surmise. Indeed, as we will see, the difference between these two quantities captures how much is lost by not knowing the order rates exactly.

4.3.2.1 Cost of imperfect information

One may assume that if Λ has a “tighter” distribution, our results should be better than if the distribution of Λ is more uncertain. In this section, we formalize this intuition by quantifying the cost of uncertainty in Λ. In particular, we calculate how much we gain in expectation if we can make our replenishment decision *after* the true value of Λ is revealed. In what follows, we drop the integrality restriction on n for ease of analysis. To further highlight the distinction, we use x in place of n to denote the decision variable.

Our goal here is to compare two values, which we will call the values of perfect and imperfect information, respectively. In practice, the difference between the two gives a bound on the amount of money a firm may want to spend to pinpoint the true value of Λ before making a decision. In our analysis, we adopt the ”rate guessing” interpretation introduced in the last section. If we guess that $\Lambda = \lambda$, then we would be wise to make the optimal decision assuming that rate, as described in Section 4.3.1. As we have relaxed the integrality restriction, the proper value of x is slightly different than the value n^* from (11). Luckily, recovering the correct value (which we denote by $x(\lambda)$) is a simple application of elementary calculus: differentiate (10) set the result equal to zero, obtaining

$$x(\lambda) = \frac{s\lambda}{h} - \frac{1}{2}.$$

Then a prediction of $\Lambda = \lambda$ corresponds to a value

$$\begin{aligned} v(x(\lambda)) &= \left(\frac{s\lambda}{h} - \frac{1}{2}\right)\left(s - \mathbb{E}\left[\Lambda^{-1}\right]\frac{h}{2}\right) - \left(\frac{s\lambda}{h} - \frac{1}{2}\right)^2 \mathbb{E}\left[\Lambda^{-1}\right]\frac{h}{2} \\ &= \frac{s}{2h}\left(s\lambda(2 - \lambda\mathbb{E}\left[\Lambda^{-1}\right]) - h\right) + \frac{\mathbb{E}\left[\Lambda^{-1}\right]h}{8}. \end{aligned}$$

In the *imperfect information* setting, we only know the distribution of Λ but not its realization. In order to maximize expected returns, we guess the best prediction $\lambda^* := \frac{1}{E[\Lambda^{-1}]}$ for the rate and make decision $x(\lambda^*)$, yielding the value of imperfect information:

$$v(x(\lambda^*)) = \frac{s}{2h}\left(\frac{s}{\mathbb{E}\left[\Lambda^{-1}\right]} - h\right) + \frac{\mathbb{E}\left[\Lambda^{-1}\right]h}{8}.$$

In contrast, in the *perfect information* scenario, we are able to predict exactly the true realization of Λ. The expected value of this unimplementable policy (with the expectation taken before Λ is revealed) is what we call the value of perfect information:

$$\begin{aligned} \mathbb{E}\left[v(x(\Lambda))\right] &:= \mathbb{E}\left[\mathbb{E}\left[v(x(\Lambda))|\Lambda\right]\right] \\ &= \mathbb{E}\left[\left(\frac{s\Lambda}{h} - \frac{1}{2}\right)\left(s - \frac{h}{2\Lambda}\right) - \left(\frac{s\Lambda}{h} - \frac{1}{2}\right)^2 \frac{h}{2\Lambda}\right] \\ &= \frac{s}{2h}\left(s\mathbb{E}\left[\Lambda\right] - h\right) + \frac{\mathbb{E}\left[\Lambda^{-1}\right]h}{8}. \end{aligned}$$

Then the cost of imperfect information is merely the difference between these two values:

$$\begin{aligned} \mathbb{E}\left[v(x(\Lambda))\right] - v(x(\lambda^*)) &= \frac{s}{2h}\left(s\mathbb{E}\left[\Lambda\right] - h\right) - \frac{s}{2h}\left(\frac{s}{\mathbb{E}\left[\Lambda^{-1}\right]} - h\right) \\ &= \frac{s^2}{2h}\left(\mathbb{E}\left[\Lambda\right] - \frac{1}{\mathbb{E}\left[\Lambda^{-1}\right]}\right). \end{aligned}$$

We see that making decisions under imperfect information has a cost that scales with the difference between the distribution's expectation $\mathbb{E}\left[\Lambda\right]$ and harmonic mean $\frac{1}{\mathbb{E}\left[\Lambda^{-1}\right]}$. This gives validation to our hypothesis that imperfect information is more

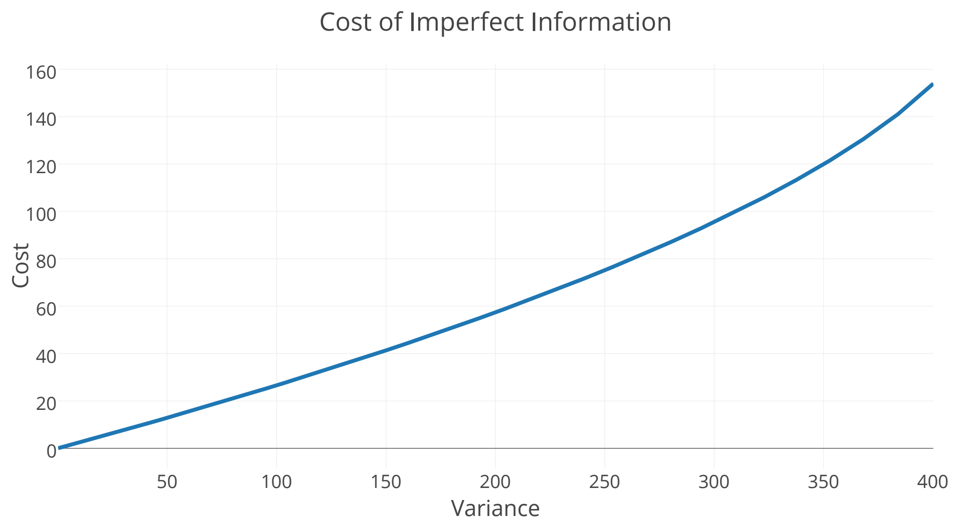


Figure 12: An example illustrating the role variance plays in affecting cost of imperfect information. We assume $s = 5, h = 1$, and Λ follows a symmetric triangular distribution with mode 50 (with *symmetric* meaning the mode is equidistant from the maximum and minimum).

costly when variance is high: as Λ is a positive random variable, the quantity $\mathbb{E}[\Lambda] - \frac{1}{\mathbb{E}[\Lambda^{-1}]}$ is always nonnegative, and strictly positive if and only if the distribution is nondegenerate. See Figure 12 for an illustration of how variance can affect the cost of imperfect information.

4.4 A more flexible model

We now outline a more flexible model which builds off the base model in Section 4.3.1. The analysis here is more complicated, but the reward is a more realistic model which is better suited to application in real-life scenarios. In particular, we will change two key assumptions of the base model, as explained below.

The first assumption we tackle regards order rates. Where in Section 4.3.1 we assumed that the order rate was constant, we now allow the rate to change over time. This is useful for applications where order levels are dependent on the day of the week (more orders on the weekend than on a week day), or time of day (more orders in the evening than early morning). In particular, we assume that the Poisson process driving orders is nonhomogeneous with intensity function $\lambda : \mathbb{R}_+ \to \mathbb{R}_+$.

Further, we assume that λ is piecewise constant, with a finite number of pieces. That is, the intensity function is of the form

$$\lambda(t) = \begin{cases} \lambda^1 & \text{if } b^0 \leq t < b^1 \\ \lambda^2 & \text{if } b^1 \leq t < b^2 \\ \vdots & \\ \lambda^\kappa & \text{if } b^{\kappa-1} \leq t < b^\kappa \end{cases} \tag{12}$$

where $\kappa \in \mathbb{Z}$, $\lambda_i > 0$ and $b^i \geq b^{i-1}$ for $i \in \{1, ..., \kappa\}$, $b^0 := 0$ and $b^\kappa := \infty$. It is easy to see that for such λ, the function $m(t)$ from (9) satisfies

$$m(b^\ell) = m(b^{\ell-1}) + (b^\ell - b^{\ell-1})\lambda^\ell.$$

More generally, letting $\ell(t)$ be the largest ℓ such that $b^\ell \leq t$, we have

$$m(t) = m(b^{\ell(t)}) + (t - b^{\ell(t)})\lambda^{\ell(t)+1}. \tag{13}$$

The second assumption regards future replenishment opportunities. In Section 4.3.1 we assumed no future replenishments would occur, and thus could afford to make inventory decisions greedily. In this case, we assume that we will have another replenishment opportunity at some unspecified time in the future, governed by nonnegative random variable T. For simplicity, we assume that T is a discrete random variable. This introduces a new dynamic to the model, in that it is possible that we expect a profit from the jth item if we stock it now, but we may prefer to wait instead until time T to lower holding costs.

In contrast to the base model, in order to calculate profits from a sale, we must now keep track of when an item was placed in inventory. For this purpose, let $\{I(t), t \geq 0\}$ denote the amount of inventory being held at time t, and let S^j denote the time at which we stock our jth item. We will find it convenient to slightly alter the dynamics of the inventory system in the case that a customer order arrives while no items exist in inventory. Instead of assuming a lost sale, we assume that we stock and immediately sell the item for a sales profit of 0. This modification has no material effect on the workings of the system, and is done simply to keep notation clean.

The manner in which the system evolves, then, is the following. At time 0 a replenishment decision is made - say n_1 items are added to inventory. Then $S^j = 0$ for $j \in \{1, ..., n_1\}$ and $I(t) = n_1$ for $t \in [0, X^1)$. At times X^j, $j \le n_1$, $I(t)$ is decreased by 1. If for some j, it happens that $I(X^j - \epsilon) = 0$ for small ϵ, then our altered dynamics demand that $S^j = X^j$. Further, we must have $I(X^j) = 1$ but $I(t)$ returns to 0 for t immediately following time X^j. Under this setup, the amount of profit generated from the jth order is

$$s \cdot \mathbb{1}_{\{X^j \neq S^j\}} - h(X^j - S^j) = (s - h(X^j - S^j))\mathbb{1}_{\{X^j > S^j\}}.$$

4.4.1 Determining replenishment levels

To make a replenishment decision at time 0, it is important to know how much profit we stand to gain for replenishing now as opposed to waiting until time T. To that end, let $Z^j(\tau)$ be the profit gained on the jth order received, assuming that $S^j = \tau$. That is,

$$Z^j(\tau) = (s - h(X^j - \tau))\mathbb{1}_{\{X^j > \tau\}}. \tag{14}$$

In this work, we replenish to level 0 if $\mathbb{E}\left[Z^1(0) < 0\right]$, and otherwise to the smallest $n \in \{0, 1, \dots\}$ with the property that for all items after the nth, it is more advantageous to wait to stock, i.e. $\mathbb{E}\left[Z^j(T)\right] > \mathbb{E}\left[Z^j(0)\right]$ for all $j \in \{n+1, n+2, \dots\}$. This number is guaranteed to exist, due to the following two results.

Lemma 4.4.1. $\mathbb{E}\left[Z_i^j(0) - Z_i^j(T)\right]$ *decreases in j.*

Proof. We have

$$\begin{aligned}\mathbb{E}\left[Z^j(0) - Z^j(T)\right] &= s - h\mathbb{E}\left[X^j\right] - \left((s + hT)\mathbb{P}\left[X^j > T\right] - h\mathbb{E}\left[X^j \mathbb{1}_{\{X^j > T\}}\right]\right) \\ &= s\left(1 - \mathbb{P}\left[X^j > T\right]\right) - h\left(\mathbb{E}\left[X^j + T\mathbb{1}_{\{X^j>T\}} - X^j\mathbb{1}_{\{X^j>T\}}\right]\right) \\ &= s\left(1 - \mathbb{P}\left[X^j > T\right]\right) - h\left(\mathbb{E}\left[\min(X^j, T)\right]\right)\end{aligned}$$

It is clear that this term decreases in increasing j, since we have that $s, h > 0$, that $P[X^j > T]$ increases as j increases, and that $\min(X^j, T)$ is non-decreasing in j on any sample path. □

Lemma 4.4.2. $\limsup_{j\to\infty} \mathbb{E}\left[Z^j(0) - Z^j(T)\right] < 0.$

Proof. Proceeding as in the proof of Lemma 4.4.1, we have

$$\begin{aligned}\limsup_{j\to\infty} \mathbb{E}\left[Z^j(0) - Z^j(T)\right] &\le \limsup_{j\to\infty} s\left(1 - \mathbb{P}\left[X^j > T\right]\right) - \limsup_{j\to\infty} h\left(\mathbb{E}\left[\min(X^j, T)\right]\right) \\ &= -\limsup_{j\to\infty} h\left(\mathbb{E}\left[\min(X^j, T)\right]\right) \\ &< 0,\end{aligned}$$

where the final line follows since $h > 0$ and both X^j and T are positive-valued. □

Computing the proper decision n is straightforward if the $\mathbb{E}\left[Z^j(\tau)\right]$ values are available (see Section 4.4.2 for details on computing this expectation). Letting $\lambda^{\max} = \max_{i\in\{1,\ldots,\kappa\}} \lambda^i$, the arguments of Section 4.3.1 imply that

$$n \le n^{\max} := \left\lfloor \frac{s\lambda^{\max}}{h} \right\rfloor.$$

Then the proper value of n may be determined by a simple bisection search over $\{0, \ldots, n^{\max}\}$.

While an improvement over the first model, this approach is still somewhat myopic in that we are only considering two time points 0 and T, instead of the full future of the system. However, the results of Section 4.6 suggest that the model can be useful in real-life applications.

4.4.2 Expressing expected profits

Making any decisions based on this model relies on knowing the values $\mathbb{E}\left[Z^j(\tau)\right]$ for $\tau = 0$ and any other $\tau \in \mathbb{R}$ such that T has positive mass at τ. Hence an efficient scheme for calculating this value is essential to the applicability of the model. Given the assumption of a piecewise constant rate function, we will now show that we can write $\mathbb{E}\left[Z^j(\tau)\right]$ in terms of algebraic operations on s, h, τ, the b^is and λ^is, as well as Poisson probabilities based on these numbers. Most importantly, the derived expression is fairly efficient to calculate in practice, opening up the model for use in real-world applications.

Before deriving our final expression for $\mathbb{E}\left[Z^j(\tau)\right]$, we will find it convenient to create some notation relating to the rate function λ over the domain $[\tau, \infty)$. Our goal is to write this restricted function in a form mirroring (12). To that end, let $b^0_\tau = \tau$ and let $i_\tau = \max\{i : b^i \leq \tau\}$. Then let

$$b^i_\tau = b^{i_\tau + i}$$

for $i \geq 1$ and while $i_\tau + i \leq \kappa$ (recall, κ is the number of "levels" in the piecewise-constant rate function (12)). Essentially, the values b^i_τ are the breakpoints of the function λ which are greater than τ, renumbered so that b^i_τ is the ith such breakpoint. Let $\kappa_\tau = \max\{i : i_\tau + i \leq \kappa\}$ be the number of these breakpoints. Lastly, let λ^i_τ be the values of the rate function λ corresponding to the intervals $[b^{i-1}_\tau, b^i_\tau)$ for $i \in \{1, ..., \kappa_\tau\}$. With this notation, we can write λ over the domain $[\tau, \infty)$ as

$$\lambda(t)\big|_{t \geq \tau} = \begin{cases} \lambda^1_\tau & \text{if } b^0_\tau \leq t < b^1_\tau \\ \lambda^2_\tau & \text{if } b^1_\tau \leq t < b^2_\tau \\ \vdots & \\ \lambda^\kappa_\tau & \text{if } b^{\kappa-1}_\tau \leq t < b^\kappa_\tau \end{cases}$$

We now proceed to deriving a more computationally-friendly expression for

$\mathbb{E}\left[Z^j(\tau)\right]$, involving just s, h, τ, the b^is and λ^is, and Poisson probabilities. To begin, we easily find

$$\mathbb{E}\left[Z^j(\tau)\right] = (s + h\tau)\mathbb{P}\left[X^j > \tau\right] - h\mathbb{E}\left[X^j \mathbb{1}_{\{X^j > \tau\}}\right].$$

We can already write $\mathbb{P}\left[X^j > \tau\right]$ as a Poisson probability due to Lemma 4.2.1. Thus we move our attention to the $\mathbb{E}\left[X^j \mathbb{1}_{\{X^j > \tau\}}\right]$ term. As $X^j \mathbb{1}_{\{X^j > \tau\}}$ is a nonnegative random variable, we may write

$$\begin{aligned}
\mathbb{E}\left[X^j \mathbb{1}_{\{X^j > \tau\}}\right] &= \int_0^\infty \mathbb{P}\left[X^j \mathbb{1}_{\{X^j > \tau\}} > t\right]\, dt \\
&= \int_0^\infty \mathbb{P}\left[X^j > t, X^j > \tau\right]\, dt \\
&= \int_0^\tau \mathbb{P}\left[X^j > \tau\right]\, dt + \int_\tau^\infty \mathbb{P}\left[X^j > t\right]\, dt \\
&= \tau\mathbb{P}\left[X^j > \tau\right] + \int_\tau^\infty \mathbb{P}\left[X^j > t\right]\, dt.
\end{aligned}$$

We've isolated another $\mathbb{P}\left[X^j > \tau\right]$ which, as before, we know how to convert to a Poisson probability. The only problematic bit is the integral term, which we turn to now. We can use Lemma 4.2.1 to derive

$$\begin{aligned}
\int_\tau^\infty \mathbb{P}\left[X^j > t\right]\, dt &= \int_\tau^\infty \sum_{k=0}^{j-1} \mathbb{P}\left[N(t) = k\right]\, dt \\
&= \int_\tau^\infty e^{-m(t)} \sum_{k=0}^{j-1} \frac{m(t)^k}{k!}\, dt.
\end{aligned}$$

Next, we use make use of (13) and (8), then rearrange to obtain.

$$\begin{aligned}
\int_\tau^\infty e^{-m(t)} \sum_{k=0}^{j-1} \frac{m(t)^k}{k!}\, dt &= \sum_{k=0}^{j-1} \frac{1}{k!} \int_\tau^\infty e^{-m(t)} m(t)^k \, dt \\
&= \sum_{k=0}^{j-1} \frac{1}{k!} \sum_{\ell=1}^{\kappa_\tau} \int_{b_\tau^{\ell-1}}^{b_\tau^\ell} e^{-m(t)} m(t)^k \, dt \\
&= \sum_{k=0}^{j-1} \frac{1}{k!} \sum_{\ell=1}^{\kappa_\tau} \int_{b_\tau^{\ell-1}}^{b_\tau^\ell} e^{-(m(b_\tau^{\ell-1})+(t-b_\tau^{\ell-1})\lambda_\tau^\ell)} \\
&\qquad\qquad \cdot (m(b_\tau^{\ell-1}) + (t-b_\tau^{\ell-1})\lambda_\tau^\ell)^k \, dt \\
&= \sum_{k=0}^{j-1} \frac{1}{k!} \sum_{\ell=1}^{\kappa_\tau} \frac{1}{\lambda_\tau^\ell} \Big[\Gamma\left(k+1, m(b_\tau^{\ell-1})\right) \\
&\qquad\qquad -\Gamma\left(k+1, (b_\tau^\ell - b_\tau^{\ell-1})\lambda_\tau^\ell + m(b_\tau^{\ell-1})\right)\Big] \\
&= \sum_{k=0}^{j-1} \frac{1}{k!} \sum_{\ell=1}^{\kappa_\tau} \frac{1}{\lambda_\tau^\ell} \left[\Gamma\left(k+1, m(b_\tau^{\ell-1})\right) - \Gamma\left(k+1, m(b_\tau^\ell)\right)\right] \\
&= \sum_{k=0}^{j-1} \frac{1}{k!} \Bigg(\frac{1}{\lambda_\tau^1} \Gamma(k+1, m(\tau)) \\
&\qquad\qquad + \sum_{\ell=1}^{\kappa_\tau - 1} \left(\frac{1}{\lambda_\tau^{\ell+1}} - \frac{1}{\lambda_\tau^\ell} \right) \Gamma\left(k+1, m(b_\tau^\ell)\right) \Bigg) \\
&= \sum_{k=0}^{j-1} \Bigg(\frac{\Gamma(k+1, m(\tau))}{\lambda_\tau^1 k!} + \\
&\qquad\qquad \sum_{\ell=1}^{\kappa_\tau - 1} \left(\frac{1}{\lambda_\tau^{\ell+1}} - \frac{1}{\lambda_\tau^\ell} \right) \frac{\Gamma\left(k+1, m(b_\tau^\ell)\right)}{k!} \Bigg).
\end{aligned}$$

Our final step involves relating the gamma function to Poisson probabilities. In fact, the cumulative distribution function of $N(t)$ (see (7) and Lemma 4.2.2) is given exactly by the final term of the product inside the sums. That is, we have

$$\begin{aligned}
\sum_{k=0}^{j-1} \frac{\Gamma\left(k+1, m(b_\tau^\ell)\right)}{k!} &= \sum_{k=0}^{j-1} \mathbb{P}\left[N(b_\tau^\ell) \le k\right] \\
&= \sum_{k=0}^{j-1} \sum_{r=0}^{k} \mathbb{P}\left[N(b_\tau^\ell) = r\right] \\
&= \sum_{k=0}^{j-1} (j-k) \mathbb{P}\left[N(b_\tau^\ell) = k\right] \\
&= j\mathbb{P}\left[N(b_\tau^\ell) \le j-1\right] - \sum_{k=1}^{j-1} k\mathbb{P}\left[N(b_\tau^\ell) = k\right] \\
&= j\mathbb{P}\left[N(b_\tau^\ell) \le j-1\right] - \sum_{k=1}^{j-1} k \frac{(m(b_\tau^\ell))^k e^{-m(b_\tau^\ell)}}{k!} \\
&= j\mathbb{P}\left[N(b_\tau^\ell) \le j-1\right] - \frac{j\Gamma(j, m(b_\tau^\ell)) - \Gamma(j+1, m(b_\tau^\ell))}{(j-1)!} \\
&\qquad - \frac{m(b_\tau^\ell)\left(e^{-m(b_\tau^\ell)}(m(b_\tau^\ell))^j + \Gamma(j+1, m(b_\tau^\ell))\right)}{j!} \\
&= (j - m(b_\tau^\ell))\mathbb{P}\left[N(b_\tau^\ell) \le j\right] + m(b_\tau^\ell)\mathbb{P}\left[N(b_\tau^\ell) = j\right] \\
&= j\mathbb{P}\left[N(b_\tau^\ell) \le j\right] - m(b_\tau^\ell)\mathbb{P}\left[N(b_\tau^\ell) \le j-1\right].
\end{aligned}$$

Bringing it all together, we get

$$\begin{aligned}
\mathbb{E}\left[X^j \mathbb{1}_{\{X^j > \tau\}}\right] = \tau\mathbb{P}\left[X^j > \tau\right] &+ \frac{1}{\lambda_\tau^1}\left(j\mathbb{P}\left[N(\tau) \le j\right] - m(\tau)\mathbb{P}\left[N(\tau) \le j-1\right]\right) \\
&+ \sum_{\ell=1}^{\kappa_\tau - 1} \left(\frac{1}{\lambda_\tau^{\ell+1}} - \frac{1}{\lambda_\tau^\ell}\right)\left(j\mathbb{P}\left[N(b_\tau^\ell) \le j\right]\right. \\
&\qquad \left. - m(b_\tau^\ell)\mathbb{P}\left[N(b_\tau^\ell) \le j-1\right]\right),
\end{aligned}$$

and so

$$\begin{aligned}
\mathbb{E}\left[Z^j(\tau)\right] = s\mathbb{P}\left[X^j > \tau\right] &- h\frac{1}{\lambda_\tau^1}\left(j\mathbb{P}\left[N(\tau) \le j\right] - m(\tau)\mathbb{P}\left[N(\tau) \le j-1\right]\right) \\
&- \sum_{\ell=1}^{\kappa_\tau - 1} h\left(\frac{1}{\lambda_\tau^{\ell+1}} - \frac{1}{\lambda_\tau^\ell}\right)\left(j\mathbb{P}\left[N(b_\tau^\ell) \le j\right]\right. \\
&\qquad \left. - m(b_\tau^\ell)\mathbb{P}\left[N(b_\tau^\ell) \le j-1\right]\right).
\end{aligned}$$

Thus, we get the main result of the section.

Theorem 4.4.3. *Let Z, b, λ, s, h be as defined in this section. Let N be the counting function for the associated Poisson process, and m the integral function of* (9)*. Then for any $j \in \mathbb{Z}_+$ and $\tau > 0$,*

$$\begin{aligned}\mathbb{E}\left[Z^j(\tau)\right] = s\mathbb{P}\left[N(\tau) \leq j-1\right] - h\frac{1}{\lambda_\tau^1}\left(j\mathbb{P}\left[N(\tau) \leq j\right] - m(\tau)\mathbb{P}\left[N(\tau) \leq j-1\right]\right) \\ - \sum_{\ell=1}^{\kappa_\tau - 1} h\left(\frac{1}{\lambda_\tau^{\ell+1}} - \frac{1}{\lambda_\tau^\ell}\right)\left(j\mathbb{P}\left[N(b_\tau^\ell) \leq j\right]\right. \\ \left. - m(b_\tau^\ell)\mathbb{P}\left[N(b_\tau^\ell) \leq j-1\right]\right).\end{aligned}$$

The above formula does not appear especially useful on first glance. However, the most computationally expensive operations involved are the evaluations of Poisson probabilities, which can be reduced to gamma function computations (see (7)). Given its wide application across many areas of mathematics, efficient approximation of the gamma function is a well studied topic, and in practice such evaluations are not computationally burdensome. Further, only $2\kappa_\tau + 1$ such computations are necessary, hence if the number of "pieces" in the rate function λ is small (which is the case in our application in Section 4.6), the total number of gamma function calculations necessary is also small. The number of terms in the final sum is also bounded by κ, thus the entire computation is quick to complete in many practical situations.

4.5 Multiple SKUs with shared resources

Up to now, we had assumed that only one type of item, or stock keeping unit (SKU), was a candidate for replenishment. Now we consider the situation where there are multiple SKUs which we would like to replenish, but the replenishment level of one SKU has an effect on the replenishment of others. For example, a warehouse stores its stock in a common area which is shared among all SKUs, so replenishing one SKU means less space for replenishments of another.

More concretely, suppose K is the number of SKUs that we consider replenishing. Each SKU $i \in \{1, 2, ..., K\}$ has an associated sales profit s_i, holding cost h_i, and order rate (function) λ_i. The jth item ordered of SKU type i has gain function Z_i^j, analogous to the functions Z^j of (14). Similarly, we denote by X_i^j the time until the jth order of item i. We must choose the replenishment vector $n \in \mathbb{Z}^K$, whose components n_i are the individual replenishment levels for each SKU. We assume that the shared resource constraints are linear, of the form $An \leq b$ for some $A \in \mathbb{Q}^{m \times K}$, $b \in \mathbb{Q}^m$.

Determining the "correct" problem to solve in such a scenario is somewhat subjection. For our purposes, we propose solving the problem of maximizing expected revenue subject to $An \leq b$ and replenishing only if the expected value of doing so now is higher than the expected value if replenishing at time T. That is, we seek to solve the problem

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{K} \sum_{j=1}^{n_i} \mathbb{E}\left[Z_i^j(0)\right] \\ \text{subject to} \quad & An \leq b \\ & \mathbb{E}\left[Z_i^j(0)\right] \geq \mathbb{E}\left[Z_i^j(T)\right] \quad \text{for } i \in \{1, ..., K\}, j \in \{1, ..., n_i\} \\ & n \in \mathbb{Z}^K. \end{aligned} \tag{15}$$

To make the model more tractable, we would like re-write it so that all constraints are linear. As a first step, we note that Lemma 4.4.1 and Lemma 4.4.2 imply that $\mathbb{E}\left[Z_i^j(0)\right] \geq \mathbb{E}\left[Z_i^j(T)\right]$ constraints need only be applied to $j = n_i$. This changes the model to

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{K} \sum_{j=1}^{n_i} \mathbb{E}\left[Z_i^j(0)\right] \\ \text{subject to} \quad & An \leq b \\ & \mathbb{E}\left[Z_i^{n_i}(0)\right] \geq \mathbb{E}\left[Z_i^{n_i}(T)\right] \quad \text{for } i \in \{1, ..., K\} \\ & n \in \mathbb{Z}^K. \end{aligned}$$

Furthermore, determining the maximum value M_i so that $\mathbb{E}\left[Z_i^{M_i}(0)\right] \geq \mathbb{E}\left[Z_i^{M_i}(T)\right]$ is precisely the focus of Section 4.4. Thus we can linearize these constrains to give an equivalent model

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{K}\sum_{j=1}^{n_i} \mathbb{E}\left[Z_i^j(0)\right] \\ \text{subject to} \quad & An \leq b \\ & n \leq M \\ & n \in \mathbb{Z}^K, \end{aligned} \tag{16}$$

where $M \in \mathbb{Z}^K$ is such that M_i is the replenishment amount from the single-SKU problem of Section 4.4, solved for SKU i. So with some precomputation, the original model (15) is transformed into the linearly-constrained model (16).

4.5.1 Binary integer programming formulation

We now propose a binary integer programming model for solving (16). This formulation will not be in the space of the original variables, but instead requires additional decision variables to linearize the cost function. In particular, we require a binary decision variable x_i^j for each SKU i and each value $j \in \{0, \ldots, M_i\}$. This is already bad news for the efficiency of creating the model, as M_i is dependent on s_i, h_i, and λ_i and hence may be arbitrarily large as compared to the size of the original problem. Nevertheless we carry on, as computational experience (see Section 4.5.2) tells us that the model can be practical for instances with reasonably-sized inputs.

Each variable x_i^j holds the interpretation that if $x_i^j = 1$ then we replenish SKU i up to level at least j. For this to make sense, one also needs to enforce that $x_i^j \geq x_i^{j+1}$ for $j \in \{1, ..., M_i - 1\}$. However, we will not need to do this explicitly in the model, as the objective function will require that an optimal solution is of this form. The full model is given below:

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{K}\sum_{j=1}^{M_i} \mathbb{E}\left[Z_i^j(0)\right] x_i^j \\
\text{subject to} \quad & An \leq b \\
& n_i = \sum_{j=1}^{M_i} x_i^j \quad \text{for } i \in \{1, ..., K\} \\
& x_i^j \in \{0,1\} \quad \text{for } i \in \{1,...,K\}, j \in \{1,...,M_i\}.
\end{aligned} \tag{17}$$

Theorem 4.5.1. *Problems* (16) *and* (17) *have the same optimal value.*

Proof. It is not hard to see that any solution n to (16) gives rise to a solution x to (17) with the same value: simply let $x_i^j = 1$ if $n_i \geq j$. Furthermore, for any $i \in \{1, ..., K\}$, an optimal solution to (17) must satisfy that $x_i^{j+1} = 0$ whenever $x_i^j = 0$ (otherwise, setting $x_i^{j+1} = 0$ and $x_i^j = 1$ gives a higher objective value). Hence setting

$$n_i = \begin{cases} 0 & \text{if } x_i^1 = 0 \\ \max\{j : x_i^j = 1\} & \text{otherwise} \end{cases}$$

gives a solution to (16) with an equivalent value. □

4.5.2 Computational results

Whether or not (17) can be used successfully in practice is a valid question. Given the inputs of (15), writing (17) requires not only calculating M but also creating $M_1 + \cdots + M_K$ variables and evaluating $\mathbb{E}\left[Z_i^j(0)\right]$ for each of them.

With this in mind, we present a computational study for problems of various sizes, the results of which are given in Table 4. For a problem with K skus and m constraints, 100 random instances were generated. For each SKU i, we randomly select $s_i \in [2,5]$ and $h_i \in [0.5,1]$. Each rate function λ_i is piecewise-linear with 5 pieces, and $5 \leq \lambda_i(t) \leq 15$ for all t. Coefficients of the constraint matrix A were taken from the range $[1,10]$. Particular selections of K and m varied between 10 to 25000 and 1 to 1000 respectively: these values were selected as we felt they would

Table 4: Solve times over 100 instances for model (17), reported as *mean* (±*std dev*). Number of replenishable SKUs is given by the rows, and number of constraints by the columns.

Constraints SKUs	1	10	50	100	500	1000
10	0.5 (±0.3)	0.3 (±0.2)	0.4 (±0.3)	0.5 (±0.4)	0.9 (±0.9)	1.2 (±0.6)
50	0.9 (±0.1)	0.8 (±0.1)	0.9 (±0.1)	0.8 (±0.1)	1.4 (±0.2)	1.5 (±0.2)
100	1.4 (±0.1)	1.1 (±0.1)	1.1 (±0.1)	1.3 (±0.1)	1.7 (±0.2)	3.2 (±0.2)
500	4.3 (±0.1)	4.3 (±0.1)	4.5 (±0.1)	4.6 (±0.1)	8.6 (±0.6)	14 (±0.9)
1000	8.5 (±0.1)	8.3 (±0.1)	8.8 (±0.1)	9.3 (±0.3)	18 (±1.3)	27 (±3.1)
5000	45 (±0.8)	42 (±0.6)	47 (±1.5)	53 (±2.6)	120 (±13)	227 (±18)
10000	98 (±3.4)	88 (±1.9)	98 (±3.7)	111 (±6.1)	249 (±34)	351 (±131)
25000	350 (±30)	249 (±6.0)	281 (±14)	317 (±24)	450 (±130)	550 (±79)

be relevant to our industry collaborator. Results show that the model can be solved under one minute for many problems sizes, while most instances solved in under 10 minutes.

4.6 Simulation study

We now return to the single-SKU extended model of Section 4.4. The aim of this section is to test the effectiveness of the model under a range of potential situations. For this purpose, we conduct a simulation study in which the model is deployed in inventory scenarios of increasing complexity.

In these simulation, we measure time in days. We do not treat time continuously but instead break the simulation into day-long chunks. Each day of the simulation, we execute the process outlined in Algorithm 3. In particular, we assume that replenishments occur at the beginning of the day, and that customer orders occur (and are immediately fulfilled) at midday. Thus for every item sold, we achieve the sales profit s but must also pay an extra half day of holding costs. Depending on the simulation scenario, daily customer orders are either determined by a Poisson random variable with appropriate rate, or pulled from historical data.

We test the model under four different scenarios of increasing complexity. The particular scenarios are:

Algorithm 3 Simulation process

```
initialize inventory $n \leftarrow 0$
initialize profits $p \leftarrow 0$
**repeat** for each day
    **if** replenishment occurs today **then**
        choose desired inventory level $n^*$
    **else**
        set $n^* \leftarrow n$
    **end if**
    set $n \leftarrow \max\{n, n^*\}$                    ▷ update inventory
    determine number of customer orders $r$
    $r \leftarrow \min\{r, n\}$                          ▷ determine fulfillable orders
    set $p \leftarrow p + r(s - h/2)$                    ▷ realize profit from orders
    set $n \leftarrow n - r$                             ▷ remove ordered items from inventory
    set $p \leftarrow p - nh$                            ▷ pay holding cost for remaining items
**until** end of simulation horizon
```

1. *Synthetic data:* We begin by testing the model under the assumptions it was built for. Namely, customer orders are governed by a Poisson process with nonhomogeneous rate function λ, and the timing of the next replenishment opportunity is determined by discrete random variable T. As in all simulations, we assume full knowledge of the distribution of T. Additionally, we assume full knowledge of the rate function λ in this first step.

2. *Real data:* Our industry collaborator provided us with several years worth of sales data. Using this data allows us to see how the model performs in a situation where day-to-day order numbers are volatile and do not follow a known distribution. We test multiple scenarios, varying in what order rates are used by the model.

    (a) *Orders as rates:* In this scenario, we assume knowledge of future order numbers and use these numbers as the values in the model's rate function. This is not implementable in practice, of course. The purpose of this exercise is to determine if a good prediction of future order numbers can be leveraged to create a model that yields high profits.

(b) *Noisy orders as rates:* This scenario is much like the last one, except that instead of giving the model the exact future order numbers, we add some "noise" to the numbers (as we describe later). The purpose of this exercise is to determine the quality of predictions necessary for the model to perform well.

(c) *No foresight:* In this scenario, we allow ourselves no access to future order numbers. We must determine proper order rates for the model based on historical data and other auxiliary data (time of year, items on sale, etc.).

For each scenario, we run the simulation over multiple replications (or using historical data from multiple SKUs, in the case that order numbers are pulled from real world data), and record the profit generated by using the model's prescriptions. These numbers do us no good in a vacuum, however; we need something to benchmark against.

For this purpose, in every scenario we also run the simulations under a *perfect foresight* assumption (note that numbers derived from this perfect foresight policy are not related to the *cost of perfect information* from Section 4.3.2.1). In this model, we assume exact knowledge of the future - that is, we know ahead of time daily order volumes and the times of all future replenishment opportunities. With this knowledge, one can deterministically calculate replenishment levels that yield the highest profit. Profits generated in this setting yield an upper bound on the profits achievable by any inventory policy. Thus performance similar to the perfect foresight policy implies a model with good performance.

Of course, comparison to a flawless benchmark may not always be fair. Thus also we benchmark against certain other, reasonable policies. The other benchmark policies will vary from scenario to scenario: we describe them as their usages arise. In each simulation, we (arbitrarily) set the sales profit $s = 10$ and daily holding cost $h = 1$. The times between replenishment opportunities are governed by i.i.d.

draws from a discrete random variable T satisfying

$$\mathbb{P}\left[T = \tau\right] = \begin{cases} 0.2 & \text{if } \tau \in \{1, 2, 3\} \\ 0.1 & \text{if } \tau \in \{4, 5, 6, 7\} \\ 0 & \text{otherwise.} \end{cases}$$

Recall that we measure time in days. So the times between replenishment opportunities are most likely to be between one and three days.

4.6.1 Synthetic data

In the first simulation scenario, we synthetically generate daily orders in a way that mimics the assumptions in Section 4.4. The simulation horizon lasts 365 days. We assume customer order rates vary from day to day, but hold constant within any particular day. For this scenario, each day's order rate is pulled uniformly-at-random from the set $\{15, \ldots, 40\}$. Total order numbers for a day are then determined from the realization of a Poisson random variable with the associated rate. For this scenario, we assume knowledge of the underlying rates, but no knowledge of the actual (future) order numbers.

Simulations follow the framework of Algorithm 3, with desired inventory levels determined by one of three different methods. First we run the perfect foresight model, which upper-bounds the amount of profit attainable. Second, we use the model of Section 4.4, where the rate function used in the model is taken directly from the rate function that governs customer orders.

Lastly, we run what we call the *expected best* policy. For this policy, we assume the same knowledge that is utilized by our model (i.e. known order rate function λ), but use this information in a natural, but perhaps less sophisticated way. First, we assume that the number of customer orders on future days will equal exactly the rates of each day (this is not an outrageous assumption, since e.g. the rate is the maximum likelihood estimate for a Poisson random variable). If we assume

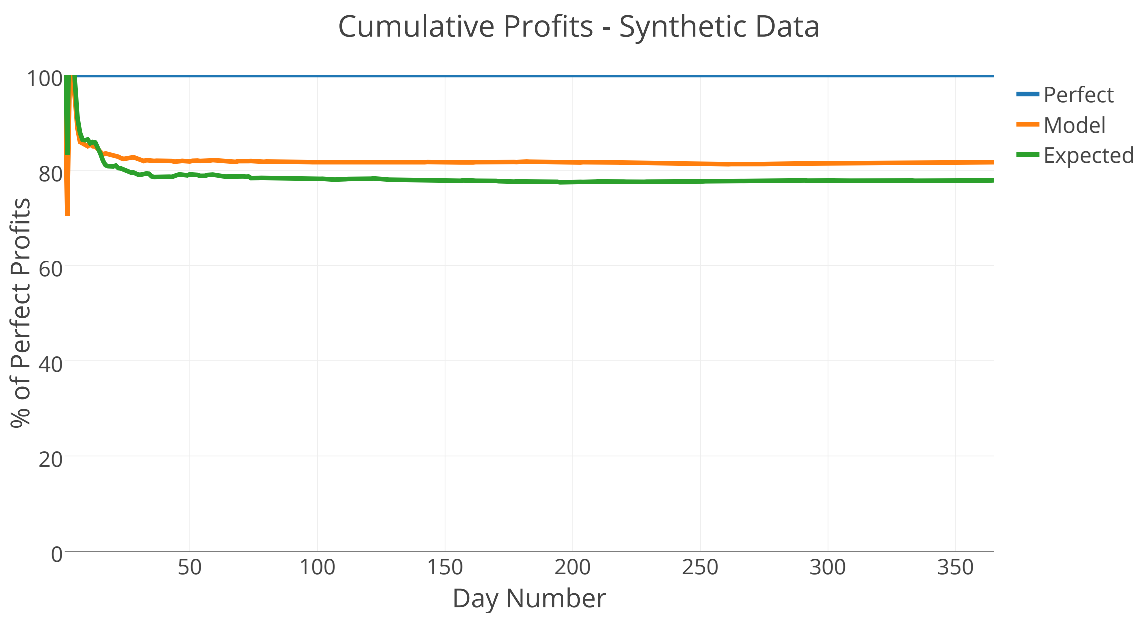


Figure 13: Cumulative profits achieved by the perfect foresight model, the model of Section 4.4, and the expected best model, under the synthetic data framework. Profits are displayed as a percentage of the maximum attainable value.

we know future orders, then we can easily calculate the correct number of items to prepackage if $T = \tau$. Denoting this value $c(\tau)$, the prescribed inventory level under the expected best policy is $\mathbb{E}\left[c(T)\right]$.

A total of 100 simulation scenarios were created, and the three models were run on each scenario. The results of these tests are displayed in Figure 13. For each model and a given day, we calculate the cumulative profits generated under the model's prescriptions since the start of the simulation. This value is divided by the value achieved by the perfect foresight policy to determine the percentage of attainable profits achieved by each model. Displayed in Figure 13 is the average value of this percentage over all replications.

By the end of the simulation horizon, the model of Section 4.4 is steadily achieving 82% of the possible profit. This is an improvement by four percentage points over the expected best policy, which by the end is averaging 78% of the possible profit. Thus one can say subjectively that the model does a decent job of realizing attainable profits in the face of uncertainty, while objectively stating that it performs better than a less sophisticated technique on the same task.

4.6.2 Real-world data

While the model's performance with synthetic data is encouraging, its true utility can only be shown via its performance on real data sets. In this section, we test the model using real order data from our industry collaborators.

Our dataset includes all orders fulfilled from a particular distribution center, with a time frame from late 2013 through the end of 2015. As expected, daily order volumes can vary greatly. Weekends typically bring more orders than weekdays. Some items are seasonal and are ordered more frequently at particular times of the year. Some items go on sale for a brief period and see increased sales volume due to improved value to the customer. Order numbers are generally greater during the holiday season from late November through December, and are particularly high on the Friday and Monday following the American Thanksgiving holiday - days known in the retail industry as "Black Friday" and "Cyber Monday."

Every retailer will have a number of consistently high-performing SKUs; ones that are reliably ordered in substantial quantity throughout the year. In our analysis, we focus on 150 of our collaborator's highest-selling SKUs. Further, the selected SKUs are nonseasonal, i.e. have consistent order volumes throughout the year.

The remaining simulations proceed once again according to Algorithm 3, where customer order numbers are determined from historical data. The simulation horizon is the entire calendar year 2015; historical data from 2013 and 2014 is reserved for training forecasting models.

The final determination is constructing the nonhomogeneous rate function λ to be used by the model. We once again will hold λ constant throughout each day, but allow it to vary over different days. In particular, we use predicted customer orders as the daily rates, with prediction methods varying for different scenarios.

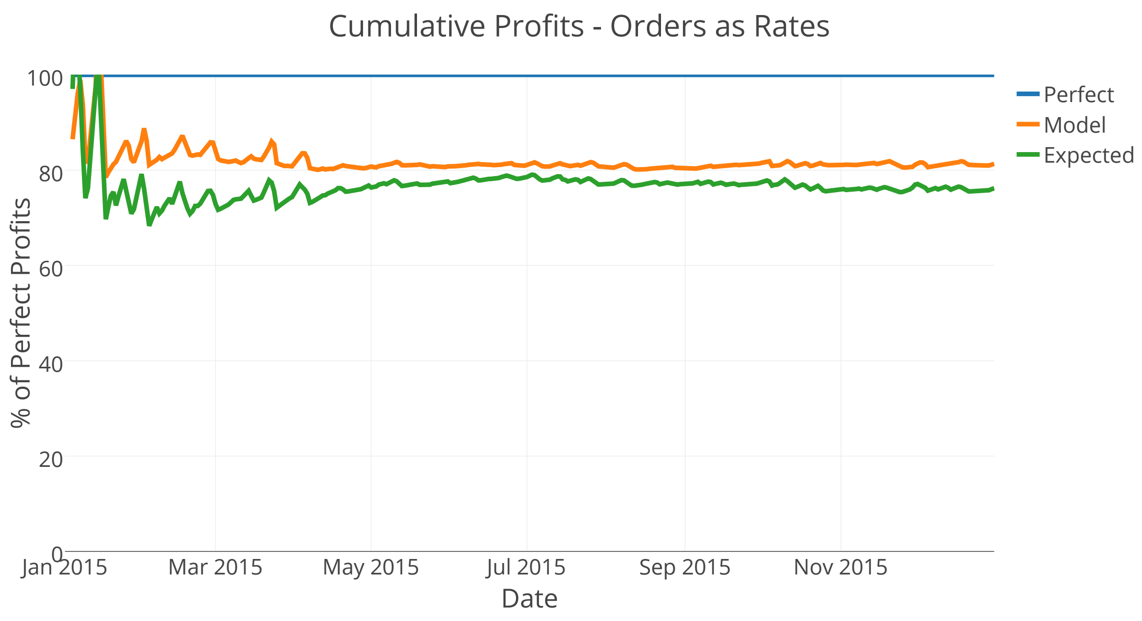


Figure 14: Cumulative profits achieved by the perfect foresight model, the model of Section 4.4, and the expected best model, under the orders-as-rates framework. Profits are displayed as a percentage of the maximum attainable value.

4.6.2.1 Orders as rates

As a first step, we test the performance of the model in the case that good order predictions are available. In this test, we assume knowledge of future orders and construct the rate function using exactly these numbers. If the model is to be applicable in any sense, its performance here should be in line with what was found in the synthetic data case.

The results of this study are displayed in Figure 14. We once again benchmark the model against both the perfect foresight policy and the expected best policy. As desired, we see performance very similar to what was found in the synthetic data framework. The model averages around 81% of the maximum attainable profit by the end of the simulation horizon. This is once again a noticeable improvement over the expected best model, which only averages 76% of attainable profits.

4.6.2.2 Noisy orders as rates

The results of the orders-as-rates simulations show that the model can perform well if we have access to good order predictions. In the next set of simulations, we explore just how "good" these predictions need to be.

We still assume access to future order numbers, but instead of using these values directly in the function λ, we first perturb them randomly. In particular, under the *noisy* p% policy ($p > 0$), we add or subtract a value up to p% of the actual order number. That is, for randomly determined $X \in (0,1)$ and $Y \in \{-1,1\}$, the rate used on a day with r orders is $\max\left\{0, r + Y\frac{rpX}{100}\right\}$[1]. In these simulations, we take X as the maximum of five uniform $(0,1)$ random variables and $\mathbb{P}[Y = 1] = \mathbb{P}[Y = -1] = 0.5$.

The results of these tests are given in Figure 15. Interestingly, adding noise up to 10% has no noticeable affect on the average performance of the model. The model achieves the same 81% average with under both the 10% noise and orders-as-rates frameworks. At 50% noise, the performance mirrors the expected best policy in the orders-as-rates framework, averaging 78% of perfect by the end of the simulation horizon. Performance degrades as noise increases (as expected), with the noisy 200% policy only obtaining around 50% of achievable profits.

Looking at the results, at appears that the model does not require extraordinarily accurate forecasts. Even at upwards of 50% noise, the model achieves over three quarters of attainable profits as compared to the perfect foresight scenario[2]. This encouraging result speaks well toward the potential applicability of the model.

[1]Actually, we don't allow λ to vanish, as this would induce division-by-zero errors. We instead use a small $\epsilon > 0$.

[2]It is worth mentioning that the noisy rates are still *unbiased* estimates of the true order numbers. Forecasts that stay within 50% of the true value but systematically over- or under-estimate would not perform as well.

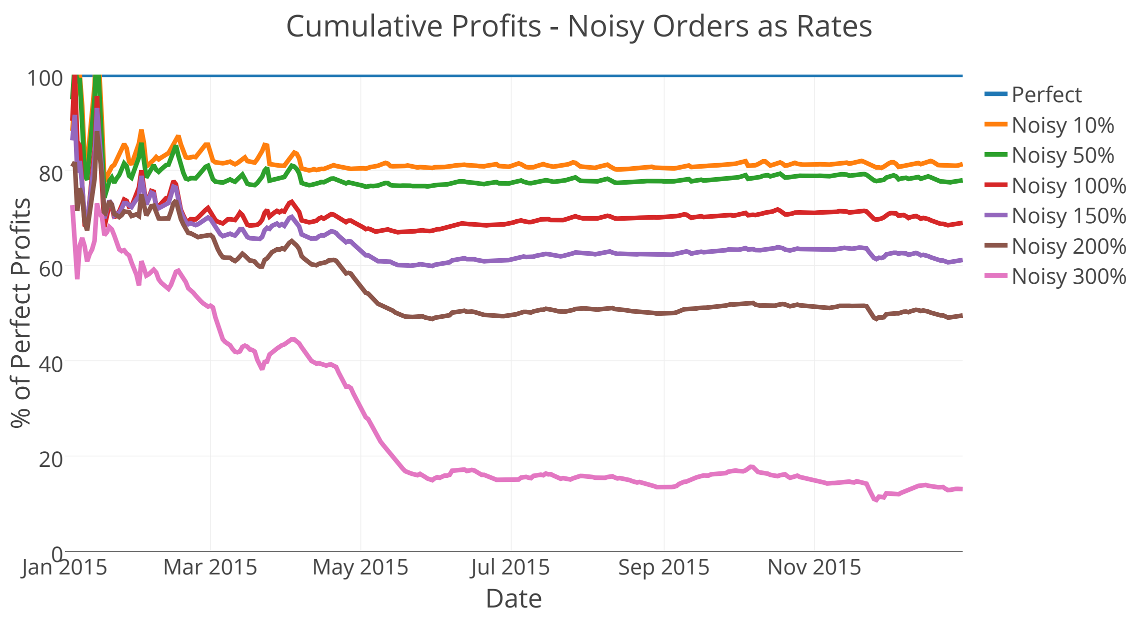


Figure 15: Cumulative profits achieved by noisy $p\%$ models, for p varying between 10 and 300. Profits are displayed as a percentage of the maximum attainable value.

4.6.2.3 Data-driven rate predictions

In the ultimate test of model applicability, we now dispense completely with any knowledge of future orders. In the next set of simulations, we must derive the model's rate function using only historical order numbers, and any (contemporaneous) exogenous data available to us.

We continue to build the model using forecasts of future order numbers. We use a wide range of forecasting techniques, from basic moving averages to more sophisticated machine learning techniques. In particular, the following approaches are applied:

- Sample average - The prediction is simply the average number of items sold per day since the beginning of the simulation horizon.
- Moving average - Like the sample average, but only considers the past $n \in \mathbb{Z}$ days of data.
- Exponential smoothing - A classic time-series analysis technique. Both single

and double exponential smoothing techniques are applied.

- Random forest regression - This well-known, ensemble-style classification technique makes use of decision trees built over random splits of the data (see [14]). Since we require numerical outputs, we use it here in its regression form.

- Neural network regression - The development of neural networks has taken a long, meandering path, perhaps starting [54]. The methods have been the subject of great fanfare due to the recent successes of deep learning.

Various models of the above forms were trained and tested via simulation. The results (once again benchmarked against the perfect foresight policy) are given in Figure 16. The figure displays results of the best method tested of the five families mentioned above.

A distinct advantage of the more sophisticated machine learning techniques (random forests and neural networks) is that by design they can incorporate a large body of information not available in simple order histories. In particular, the best neural network tested included indicators for the day of the week (both for the prediction day and historical data). The best random forest also included a "holiday season" indicator, which turns on during the high shopping period between Thanksgiving and Christmas (the same information was not as useful in the neural network). Other exogenous information was also gathered and tested (e.g. Google search volumes for key terms related to a product), but did not appear to improve the outcomes for either model.

Nonetheless, these two machine learning methods achieved the best results of all methods tested. The best neural network attains 61% of the possible profits on average, while the random forest achieves 58% of the perfect foresight policy. The random forest barely outperforms the best moving average method, however, with

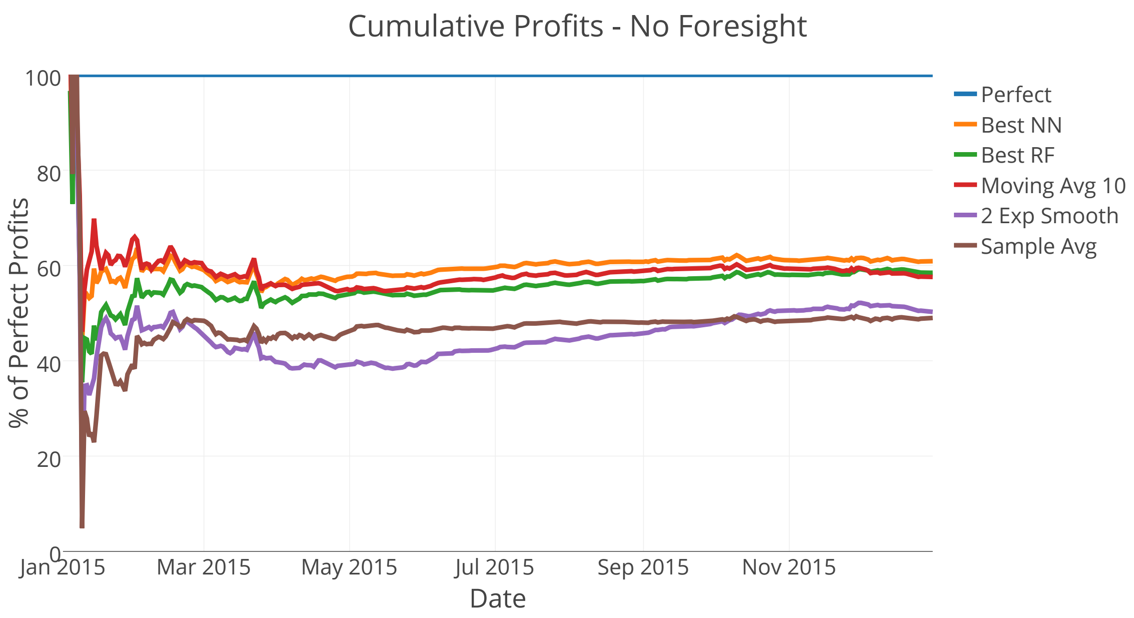


Figure 16: Cumulative profits achieved by various models under the no foresight framework. Profits are displayed as a percentage of the maximum attainable value.

only a single percentage point difference between them. The sample average and exponential smoothing lag behind noticeably. These results indicate that sophisticated regression techniques using auxiliary data can generate improved prescriptions when paired with an optimization model.

Figure 17 and Figure 18 combine to give a sense of how the machine learning models gain their advantage. Where Figure 16 plots the *average* percentage of possible profit attained by each algorithm, these new plots show the 90th and 10th *percentile* over all runs, respectively. We see that at the 90th percentile, almost all methods perform equally well. However, looking at the 10th percentiles, we see that the machine learning methods are noticeably better in the worst case.

4.6.2.4 Measuring the impact of exogenous information

We've established that auxiliary data can be useful for attaining improved profits, but suppose we want to quantify this impact. Recently, [9] proposed a natural way to quantify the usefulness of extra information via what they call the *coefficient of*

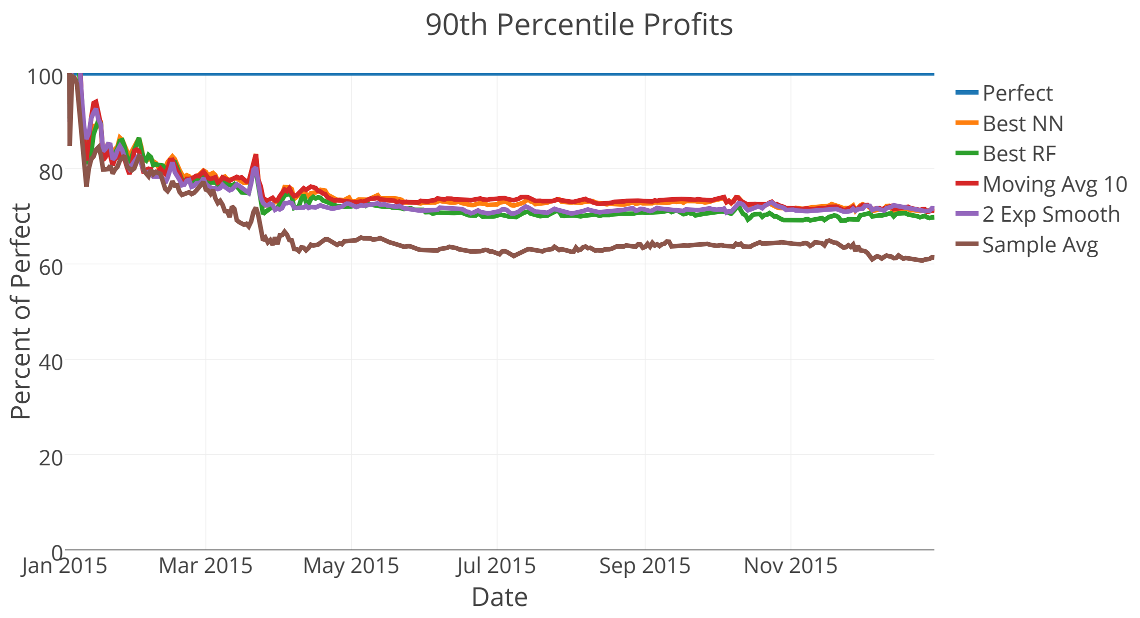


Figure 17: 90th percentile of cumulative profits achieved by various models under the no foresight framework. Profits are displayed as a percentage of the maximum attainable value.

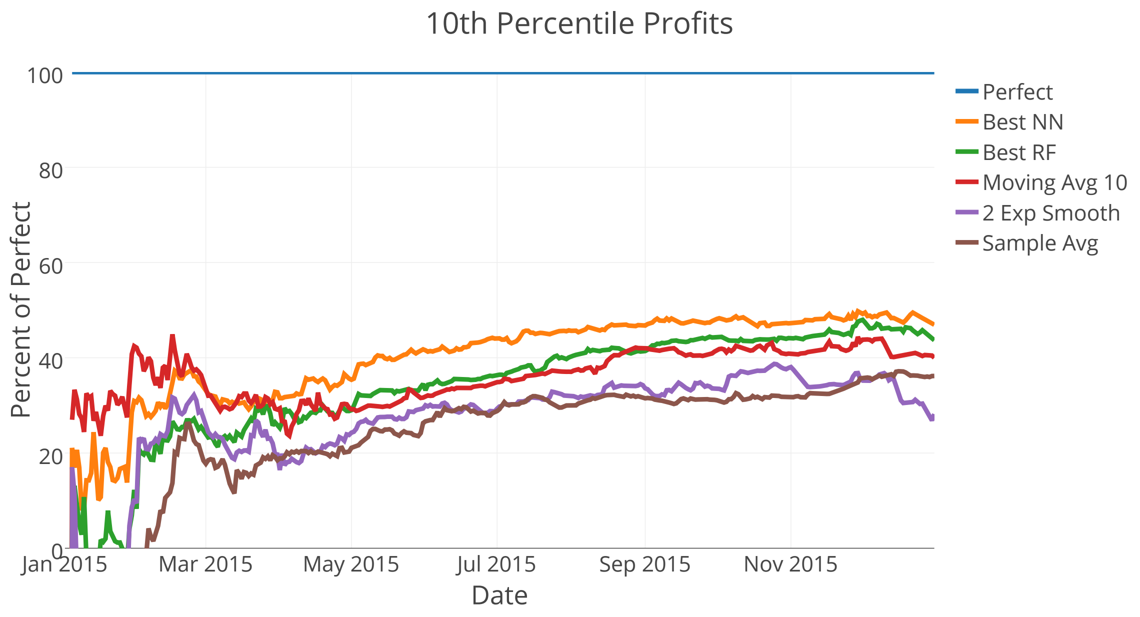


Figure 18: 10th percentile of cumulative profits achieved by various models under the no foresight framework. Profits are displayed as a percentage of the maximum attainable value.

prescriptiveness P. The first step to calculating this quantity is to determine the gap in profit between a low-information method and the perfect foresight policy. Then P can be described as the proportion of this gap that is recovered by the model of interest, using the new information. That is, if v_{perfect} is the maximum attainable profit, v_{LI} is the profit of the low-information method, and v_{model} is the profit generated by the model of interest, we have

$$P = 1 - \frac{v_{\text{model}} - v_{\text{perfect}}}{v_{\text{SA}} - v_{\text{perfect}}}.$$

Clearly, $P \in [0, 1]$ with a higher value indicating a better model. For the purposes of calculating P, the authors of [9] suggest using the sample average policy. Using this standard, the prescriptiveness of the best neural network policy is $P = 0.23$.

4.7 Conclusions

In this chapter, we've presented an inventory model for situations in which the times of future replenishment opportunities are not known exactly. We describe how to use the model to decide proper inventory levels, including the case where inventory must be managed for multiple SKUs which share resources. We test out the model using real data from a large distribution center, and demonstrate that the model, when provided with good order predictions, can be used to attain profits above 80% of what would be achievable with perfect foresight, and better returns than other policies using the same information. Hence we reduce the question of making good inventory decisions to the question of making good order volume predictions. We demonstrate that certain auxilliary data can be useful in driving these inventory decisions, with the best model closing 23% of the gap between the simplest prediction methods and the perfect foresight case. This leaves a significant gap still to close, which could be an avenue of future research.

References

[1] Achterberg, T., "SCIP: Solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 1–41, 2009.

[2] Arora, S., Hazan, E., and Kale, S., "The multiplicative weights update method: a meta-algorithm and applications.," *Theory of Computing*, vol. 8, no. 1, pp. 121–164, 2012.

[3] Arrow, K. J., Harris, T., and Marschak, J., "Optimal inventory policy," *Econometrica*, vol. 19, no. 3, pp. 250–272, 1951.

[4] Balas, E., "Intersection cuts - a new type of cutting planes for integer programming," *Operations Research*, vol. 19, no. 1, pp. 19–39, 1971.

[5] Ben-Tal, A. and Nemirovski, A., *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001.

[6] Benchetrit, Y., Fiorini, S., Huynh, T., and Weltge, S., "Characterizing polytopes contained in the 0/1-cube with bounded Chvátal-gomory rank," 2016.

[7] Berlekamp, E. R., McEliece, R. J., and Van Tilborg, H. C., "On the inherent intractability of certain coding problems," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, 1978.

[8] Berthold, T., "Measuring the impact of primal heuristics," *Operations Research Letters*, vol. 41, no. 6, pp. 611–614, 2013.

[9] Bertsimas, D. and Kallus, N., "From predictive to prescriptive analytics," 2014. arXiv:1402.5481.

[10] Bienstock, D., "Approximately solving large-scale linear programs. I. Strengthening lower bounds and accelerating convergence," *CORC Report*, 1999.

[11] Bienstock, D., *Potential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice*, vol. 53 of *International Series in Operations Research & Management Science*. Springer, 2002.

[12] Bockmayr, A., Eisenbrand, F., Hartmann, M., and Schulz, A. S., "On the Chv'atal rank of polytopes in the 0/1 cube," *Discrete Applied Mathematics*, vol. 98, no. 1 - 2, pp. 21 – 27, 1999.

[13] Bodur, M., Pia, A. D., Dey, S. S., Molinaro, M., and Pokutta, S., "Aggregation-based cutting-planes for packing and covering integer programs," 2016. arXiv:1606.08951.

[14] Breiman, L., "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[15] Caprara, A. and Fischetti, M., "$\{0, 1/2\}$-chvátal-gomory cuts," *Mathematical Programming*, vol. 74, no. 3, pp. 221–235, 1996.

[16] Caprara, A., Fischetti, M., and Letchford, A. N., "On the separation of maximally violated mod-k cuts," *Mathematical Programming*, vol. 87, no. 1, pp. 37–56, 2000.

[17] Chvátal, V., "Edmonds polytopes and a hierarchy of combinatorial problems," *Discrete Mathematics*, vol. 4, no. 4, pp. 305 – 337, 1973.

[18] Conn, A. R., Gould, N. I. M., and Toint, P. L., *Trust Region Methods*. SIAM, 2000.

[19] Cornuéjols, G. and Lee, D., "On some polytopes contained in the 0,1 hypercube that have a small Chvátal rank," in *Proceedings of the 18th International Conference on Integer Programming and Combinatorial Optimization - Volume 9682*, IPCO 2016, (New York, NY, USA), pp. 300–311, Springer-Verlag New York, Inc., 2016.

[20] Cornuéjols, G. and Li, Y., "Deciding emptiness of the gomory-chvátal closure is np-complete, even for a rational polyhedron containing no integer point," in *Proceedings of the 18th International Conference on Integer Programming and Combinatorial Optimization - Volume 9682*, IPCO 2016, (New York, NY, USA), pp. 387–397, Springer-Verlag New York, Inc., 2016.

[21] Dantzig, G., "Maximization of a linear function of variables subject to linear inequalities," in *Activity Analysis of Production and Allocation*, Cowles Commission Monograph No. 13, pp. 339–347, New York, N. Y.: John Wiley & Sons Inc., 1951.

[22] Dash, S., "A note on QUBO instances defined on Chimera graphs," *preprint arXiv:1306.1202*, 2013.

[23] Dash, S. and Puget, J.-F., "On quadratic unconstrained binary optimization problems defined on Chimera graphs," *Optima*, vol. 98, pp. 2–6, 2015.

[24] De Loera, J. A., Hemmecke, R., and Köppe, M., *Algebraic and geometric ideas in the theory of discrete optimization*, vol. 14 of *MOS-SIAM Series on Optimization*. SIAM, 2013.

[25] De Loera, J. A., Hemmecke, R., Köppe, M., and Weismantel, R., "FPTAS for optimizing polynomials over the mixed-integer points of polytopes in fixed dimension," *Math. Program.*, vol. 115, no. 2, pp. 273–290, 2008.

[26] De Loera, J. A., Hemmecke, R., and Lee, J., "Augmentation in linear and integer linear programming," *Preprint arXiv:1408.3518*, 2014.

[27] Edmonds, J. and Karp, R. M., "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of the ACM*, vol. 19, no. 2, pp. 248–264, 1972.

[28] Eisenbrand, F., "Note–on the membership problem for the elementary closure of a polyhedron," *Combinatorica*, vol. 19, no. 2, pp. 297–300, 1999.

[29] Eisenbrand, F. and Schulz, A. S., "Bounds on the Chvátal rank of polytopes in the 0/1-cube*," *Combinatorica*, vol. 23, no. 2, pp. 245–261, 2003.

[30] Fischetti, M. and Lodi, A., "Local branching," *Mathematical Programming*, vol. 98, no. 1-3, pp. 23–47, 2003.

[31] Fischetti, M. and Monaci, M., "Proximity search for 0-1 mixed-integer convex programming," *Journal of Heuristics*, vol. 20, no. 6, pp. 709–731, 2014.

[32] Frank, A. and Tardos, É., "An application of simultaneous Diophantine approximation in combinatorial optimization," *Combinatorica*, vol. 7, no. 1, pp. 49–65, 1987.

[33] Garg, N. and Koenemann, J., "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," *SIAM Journal on Computing*, vol. 37, no. 2, pp. 630–652, 2007.

[34] Gomory, R. E., "Outline of an algorithm for integer solutions to linear programs," *Bull. Amer. Math. Soc.*, vol. 64, pp. 275–278, 09 1958.

[35] Gomory, R. E., "Solving linear programming problems in integers," in *Proceedings of Symposia in Applied Mathematics X* (Bellman, R. and Hall, M., eds.), pp. 211–215, American Mathematical Society, 1960.

[36] Graham, R. L., Grötschel, M., and Lovász, L., *Handbook of combinatorics*, vol. 1. Elsevier, 1995.

[37] Graver, J. E., "On the foundations of linear and integer linear programming i," *Mathematical Programming*, vol. 9, no. 1, pp. 207–226, 1975.

[38] Hansen, P., Mladenović, N., and Urošević, D., "Variable neighborhood search and local branching," *Computers and Operations Research*, vol. 33, pp. 3034–3045, 2006.

[39] Harris, F. W., "How many parts to make at once," *Factory, The Magazine of Management*, vol. 10, no. 2, pp. 135–136, 1913.

[40] Hemmecke, R., Köppe, M., Lee, J., and Weismantel, R., "Nonlinear integer programming," in *50 Years of Integer Programming 1958–2008 – From the Early Years to the State-of-the-Art* (Jünger, M., Liebling, T. M., Naddef, D., Nemhauser, G. L., Pulleyblank, W. R., Reinelt, G., Rinaldi, G., and Wolsey, L. A., eds.), pp. 561–618, Springer, 2010.

[41] Hemmecke, R., Köppe, M., and Weismantel, R., "Graver basis and proximity techniques for block-structured separable convex integer minimization problems," *Mathematical Programming*, vol. 145, no. 1-2, pp. 1–18, 2014.

[42] Hemmecke, R., Onn, S., and Weismantel, R., "A polynomial oracle-time algorithm for convex integer minimization," *Math. Program.*, vol. 126, no. 1, pp. 97–117, 2011.

[43] Karmarkar, N., "A new polynomial-time algorithm for linear programming," in *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, (New York, NY, USA), pp. 302–311, ACM, 1984.

[44] Karp, R., "Reducibility among combinatorial problems. complexity of computer computations,(re miller and jm thatcher, eds.), 85–103," 1972.

[45] Khachiyan, L. G., "A polynomial algorithm in linear programming," *Dokl. Akad. Nauk SSSR*, vol. 244, no. 5, pp. 1093–1096, 1979.

[46] Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R. E., Danna, E., Gamrath, G., Gleixner, A. M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D. E., and Wolter, K., "MIPLIB 2010: mixed integer programming library version 5," *Math. Program. Comput.*, vol. 3, no. 2, pp. 103–163, 2011.

[47] Le Bodic, P., Pavelka, J. W., Pfetsch, M. E., and Pokutta, S., "Solving MIPs via Scaling-based Augmentation, year = 2015, howpublished = http://www.optimization-online.org/db_html/2015/09/5097.html, note = Accessed: 2015-10-17."

[48] Lee, D., Cornuejols, G. P., and Li, Y., "On the computational complexity of optimizing over the Chvátal closure of a polytope," in *INFORMS Annual Meeting 2016*, (Nashville, Tennessee, USA), 2016.

[49] Lee, J., Onn, S., Romanchuk, L., and Weismantel, R., "The quadratic Graver cone, quadratic integer minimization, and extensions," *Math. Program.*, vol. 136, no. 2, pp. 301–323, 2012.

[50] Lee, J., Onn, S., and Weismantel, R., "On test sets for nonlinear integer maximization," *Oper. Res. Lett.*, vol. 36, no. 4, pp. 439–443, 2008.

[51] Letchford, A. N. and Lodi, A., "An augment-and-branch-and-cut framework for mixed 0-1 programming," in *Combinatorial Optimization—Eureka, You Shrink!*, pp. 119–133, Springer, 2003.

[52] Letchford, A. N., Pokutta, S., and Schulz, A. S., "On the membership problem for the $\{0, 1/2\}$-closure," *Operations Research Letters*, vol. 39, no. 5, pp. 301–304, 2011.

[53] McCormick, S. T. and Shioura, A., "Minimum ratio canceling is oracle polynomial for linear programming, but not strongly polynomial, even for networks," *Operations Research Letters*, vol. 27, no. 5, pp. 199–207, 2000.

[54] McCulloch, W. S. and Pitts, W., "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[55] Nemhauser, G. and Wolsey, L., *Integer and Combinatorial Optimization*. Wiley, 1988.

[56] Onn, S., *Nonlinear Discrete Optimization: An Algorithmic Theory*. Zurich lectures in advanced mathematics, European Mathematical Society Publishing House, 2010.

[57] Orlin, J. B. and Ahuja, R. K., "New scaling algorithms for the assignment and minimum mean cycle problems," *Mathematical Programming*, vol. 54, no. 1-3, pp. 41–56, 1992.

[58] Padberg, M. W. and Grötschel, M., "Polyhedral computations," in *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization* (Lawler, E. L., Lenstra, J. K., and A. H. G. Rinnoy Kan) and D. B. Shmoys, Y. . ., eds.), pp. 307–360, John Wiley.

[59] Plotkin, S. A., Shmoys, D. B., and Tardos, É., "Fast approximation algorithms for fractional packing and covering problems," *Mathematics of Operations Research*, vol. 20, no. 2, pp. 257–301, 1995.

[60] Pokutta, S. and Stauffer, G., "Lower bounds for the chvátal–gomory rank in the 0/1 cube," *Operations Research Letters*, vol. 39, no. 3, pp. 200–203, 2011.

[61] Resnick, S., *Adventures in Stochastic Processes*. Birkhäuser Boston, 2013.

[62] Rockafellar, R. T., "Monotone operators and the proximal point algorithm," *SIAM Journal on Control and Optimization*, vol. 14, no. 5, pp. 877–898, 1976.

[63] Ross, S., *Stochastic processes*. Wiley series in probability and statistics: Probability and statistics, Wiley, 1996.

[64] Rothvoss, T. and Sanitá, L., "0/1 polytopes with quadratic Chvátal rank," in *Integer Programming and Combinatorial Optimization* (Goemans, M. and Correa, J., eds.), vol. 7801 of *Lecture Notes in Computer Science*, pp. 349–361, Springer Berlin Heidelberg, 2013.

[65] Scarf, H. E., "Test sets for integer programs," *Mathematical Programming*, vol. 79, no. 1-3, pp. 355–368, 1997.

[66] Schrijver, A., *Theory of linear and integer programming*. Wiley, 1986.

[67] Schulz, A. S. and Weismantel, R., "The complexity of generic primal algorithms for solving general integer programs," *Mathematics of Operations Research*, vol. 27, no. 4, pp. 681–692, 2002.

[68] Schulz, A. S., Weismantel, R., and Ziegler, G. M., "0/1-integer programming: Optimization and augmentation are equivalent," in *Algorithms – ESA '95, Proceedings*, pp. 473–483, 1995.

[69] "Solving Constraint Integer Programs, Version 3.2.0," 2015. http://scip.zib.de/.

[70] Wallacher, C. and Zimmermann, U., "A combinatorial interior point method for network flow problems," *Mathematical programming*, vol. 56, no. 1-3, pp. 321–335, 1992.

VITA

Jeffrey William Pavelka was born in Manhattan, Kansas on March 2, 1988. He graduated from Kansas State University in 2011 with dual BS/MS degrees in Industrial and Manufacturing Systems Engineering. The following fall, he enrolled as a Ph.D. student at the H. Milton Stewart School of Industrial and Systems Engineering at the Georgia Institue of Technology, where his research has focused on integer programming as well as applications of data science and machine learning in real-world optimization problems.