New Integer Programming Approaches to Open-Pit Mining and Metabolic Engineering Problems

by

Amanda G. Smith

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Industrial and Systems Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2019

Date of final oral examination: December 17, 2018

The dissertation is approved by the following members of the Final Oral Committee:

Jim Luedtke, Associate Professor, Industrial and Systems Engineering
Laura Albert, Associate Professor, Industrial and Systems Engineering
Jeff Linderoth, Professor, Industrial and Systems Engineering
Alexandra Newman, Professor, Department of Mechanical Engineering (Colorado School of Mines)
Victor Zavala, Associate Professor, Chemical and Biological Engineering

To Mom and Dad. I wouldn't be here without you, in so many ways.

ACKNOWLEDGMENTS

I am and always will be indebted to my PhD advisor, Dr. Jim Luedtke. I would like to express my sincere gratitude for his continuous guidance, patience, and encouragement.

I would also like to thank the rest of my thesis committee, Drs. Jeff Linderoth, Laura Albert, Alexandra Newman, and Victor Zavala. Their insightful questions and feedback throughout my research and writing process have been invaluable in the production of this thesis.

I must also thank my labmates and fellow graduate students who have helped me debug endless lines of code, caught errors in my logic, and convinced me to have fun from time to time. My thanks extend further to my family, friends, and faith community, who have supported me and believed in me when I didn't believe in myself. I wouldn't have finished this without all of you.

And thank you, God, for giving me strength when I had none.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

We study new optimization-driven approaches to two engineering problems, employing techniques from integer, bilevel, stochastic, and multi-objective programming. We first present an approach to the open-pit mining truck dispatching problem that utilizes mixed-integer programming (MIP). The truck dispatching problem seeks to determine how trucks should be routed through the mine as they become available. We describe an optimization-driven approach to solving the dispatching problem in the form of a MIP model. The model is difficult to solve directly, so we present a heuristic that quickly produces high-quality feasible solutions to the model. We give computational results demonstrating the effectiveness of the proposed heuristics and several key model components. To show that our model finds solutions that meet the open-pit mining objectives while accounting for key problem components in novel ways, we embed the MIP-based dispatching policy in a discrete-event simulation of an open-pit mine. We further create two additional heuristic dispatching policies that rely on a new nonlinear rate-setting model that treats queueing at each site as an M/G/1 queue. We present a full computational study of the three policies in which we perform output analysis on key metrics of the open-pit mine simulation. We show that the MIP-based dispatching policy consistently outperforms the heuristic dispatching policies on open-pit mines with a variety of characteristics. The second problem we study is selecting metabolic network changes in cellular organisms. In this problem, enzymes are used to alter the rates at which reactions occur in cellular organisms, causing the cell to increase the output of a desired biochemical product. In existing bilevel MIP models, the lower-level cellular objective is modeled as either maximizing cellular growth or minimizing the biochemical output. We combine these perspectives with two new bilevel MIP models: a single-objective maximum productivity model and a bi-objective maximum yield and maximum growth model. We finally present two-stage stochastic extensions of both models in which we maximize the expected values of productivity, yield, and growth when the planned changes to the metabolic network are uncertain. Because the

stochastic bi-objective model contains a complicating budget constraint that lacks parallel structure, we describe a heuristic that alternates between a scenario decomposition-based algorithm and allocating the budget to individual scenarios. Ultimately, we show that this methodology can be implemented to find solutions that meet the metabolic engineering objectives but which are less sensitive to uncertainty than solutions to existing models.

1 INTRODUCTION

We investigate two research projects that apply integer, bilevel, and multi-objective programming to engineering problems. In this introductory chapter, we start by providing an overview of the following chapters in Section 1.1. Then, in Sections 1.2-1.5, we provide background information on several key methodologies used in this thesis.

1.1 Overview

We start by describing a new approach to truck dispatching in open-pit mining. The dispatching problem involves making real-time truck dispatching decisions in a stochastic dynamic environment. Historically, the dispatching problem has been solved with a linear flow-rate-setting model, followed by a heuristic that matches decisions to target rates (Elbrond and Soumis (1987a); White (1991); Temeng (1998); Subtil et al. (2011); Ahangaran et al. (2012)). Existing dispatching heuristics tend to be myopic, focusing only on the immediate dispatching decisions and ignoring the future state of the system. We propose a time-discretized mixed-integer programming (MIP) model that captures all the necessary problem characteristics and yields real-time dispatching decisions. We find a trade-off between competing objectives by treating the multi-objective function as a three-phase hierarchy. To solve the challenging third phase, we describe an implementation of the relaxation-induced neighborhood search algorithm from Danna et al. (2005) that produces good feasible solutions quickly. Finally, we present a computational study that demonstrates our model and solution methods work as expected. We find that solutions to our model meet open-pit mining objectives while accounting for key problem components in novel ways. We present the MIP model of the dispatching problem in its entirety in Chapter 2.

In Chapter 3, we test and demonstrate the performance of the MIP dispatching model by incorporating it into a discrete-event simulation of an open-pit mine and comparing it to two competing policies. As part of our proposed dispatching policies, we describe a

nonlinear model that sets target flow rates for trucks moving between locations in the mine. We then give two heuristics that match decisions to the target rates. The first is a simple greedy heuristic which selects the next destination as the route with the greatest total deviation from the target flow rates and deviation from target flow rates on the possible return trips. The second heuristic considers all possible sequences of two dispatching decisions for the trucks becoming available in the next T time periods and solves a small MIP model to match trucks to trips. This method has the benefit of including information about how current dispatching decision will affect future decisions. The final component of this study is evaluating these heuristic policies and comparing them to the policy obtained from the MIP model described in Chapter 2 by running the open-pit mine simulation over a 10-hour period and collecting metrics on how well each policy meets processing, mining, and ore quality targets. We vary characteristics of the mine, such as the variance of the travel times and the number of trucks available, to evaluate how each policy performs under different conditions. We show that the discrete-time MIP dispatching policy consistently results in higher volume and quality of ore than the target rate-based policies. Based on the results of this study, we recommend that the discrete-time MIP dispatching model be used to make truck-dispatching decisions in open-pit mines with a variety of different characteristics.

In Chapter 4. we present new bilevel MIP models for metabolic engineering problems. Historically, these problems have been solved with bilevel programming models, in which the decision maker changes some reaction rate (flux) bounds with a top-level objective to maximize certain biochemical outputs. The lower level cellular behavior is typically modeled as minimizing the biochemical output or maximizing cellular growth (Burgard et al. (2003); Cotten and Reed (2013)). Because the true flux bound changes and cellular responses are uncertain, Cotten and Reed (2013) find the expected output by simulating the success of the suggested bound changes. Our models build on the existing models by producing sets of alternative solutions that are less sensitive to the uncertainty inherent

in implementing the bound changes. The first model maximizes the worst-case cellular productivity, which we approximate as the product of cellular growth and biochemical yield, but the model only results in a slight reduction of solution sensitivity. The second model makes a more significant improvement in solution sensitivity through the addition of a second objective to the top level of maximizing cellular growth. The third and fourth models are stochastic extensions of the maximum productivity and bi-objective models, respectively. We demonstrate the behavior of solutions to each of these four models with data from *Escherichia coli* cells. The first two models can be solved directly with commercial MIP solvers but the stochastic models require use of a heuristic algorithm to find good feasible solutions. We suggest a scenario decomposition algorithm – such as dual decomposition or progressive hedging – to find solutions to the maximum productivity model. The stochastic bi-objective model has additional complexity in the form of a budget constraint that cannot be decomposed by scenario using the standard dual decomposition framework. Thus, we propose a heuristic for allocating the budget across different scenarios in alternation with a decomposition-based algorithm. Ultimately, we show that both the deterministic and stochastic versions of the bi-objective model – and, to a lesser extent, the maximum productivity model – have the potential to discover solutions that meet the metabolic engineering objectives but which are less sensitive to uncertainty than solutions to existing models.

1.2 Integer Programming

We begin with a general overview of a class of optimization known as *integer programming*.[1] Integer programming involves decision making where one or more of the decisions must take an integer value. If all decisions are integral, the problem is a *pure integer program*. Much more often, problems contain a mix of integer and continuous decisions and this class of problems is referred to as *mixed-integer programs.* We provide a very brief introduction

[1]Notation and information in this section are adapted from Conforti et al. (2014)

to methods used to solve integer programs, as well as a discussion of the complexity of such problems and the current state of integer programming solvers.

Integer Programming Basics

We begin by introducing the general structure of a mixed-integer program (MIP) with a linear objective and constraints. Let $c \in \mathbb{R}^n$, $d \in \mathbb{R}^p$, and $b \in \mathbb{R}^m$. Further, let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m \times p}$. We formulate a MIP in the following way:

$$\begin{aligned} &\min c^T x + d^T y \\ &\text{s.t. } Ax + By \geq b \\ &\quad x \in \mathbb{R}^n \\ &\quad y \in \mathbb{Z}^p. \end{aligned} \tag{MIP}$$

Note that this is also the form for a pure integer linear program if $n = 0$. In addition, a more general integer program could include a nonlinear objective function or nonlinear constraints.

In general, MIPs are challenging to solve. In fact, integer programs are classified as $\mathcal{NP}$-hard, meaning every problem in $\mathcal{NP}$ (non-deterministic polynomial time complexity class) can be reduced in polynomial time to an integer program.

Methods for Solving Integer Programs

Two general methods exist for solving any instance of (MIP): *branch-and-bound* and *cutting plane* algorithms. We give a short overview of these two methods in this section but refer the reader to Conforti et al. (2014) for a more detailed description. Both of these methods rely heavily on the linear programming (LP) relaxation of the given instance of (MIP), which removes the constraints $y \in \mathbb{Z}^p$ but is otherwise identical to the instance of (MIP).

Because the problem is less constrained, the LP relaxation provides a lower bound on the optimal MIP objective value.

Branch-and-Bound Algorithm

As its name indicates, the branch-and-bound algorithm involves two major steps: *branching* on an individual integer variable and computing *bounds* on the objective value at the optimal integer solution. In particular, the LP relaxation is solved to obtain an initial solution. An integer variable with a fractional solution $y_i' = f$ is selected on which to branch. Next, two subproblems are created, one in which the constraint $y_i' \geq \lceil f \rceil$ is added to the LP relaxation and the other in which the constraint $y_i' \leq \lfloor f \rfloor$ is added. The resulting LPs are then solved and the process is repeated, resulting in a *branch-and-bound tree*, where each node represents an LP relaxation with additional constraints. The objective value at each node provides a lower bound for all subproblems of that node. Nodes are *pruned* – meaning they are no longer branched on – when the LP is infeasible, the objective value at the node is bigger than the best known upper bound, or the node's solution is feasible to the integer program (yielding a new upper bound). The process is repeated until the optimal MIP solution is found. Key decisions that must be made in the algorithm are which variable to branch on and in what order subproblems should be solved.

Cutting Plane Algorithm

Much like the branch-and-bound algorithm, a cutting plane algorithm relies on an initial solution to the LP relaxation of (MIP). If the solution obtained to the LP relaxation is feasible to (MIP), the solution is optimal. If, however, the vector of integer variables y' contains at least one fractional component (i.e., $y_i' = f_i$), a *separation problem* must be solved. The separation problem requires finding a cutting plane of the form $\alpha x + \gamma y \geq \beta$ that is satisfied by any point feasible to (MIP), but such that $\alpha x' + \gamma y' < \beta$. This constraint is added to the LP relaxation and the process is repeated until a point is found which is feasible to (MIP).

The key challenge of the cutting plane algorithm is finding a good choice of cutting plane at each iteration.

Integer Programming in Practice

As discussed, MIPs are hard to solve in general. However, there has been significant progress in making good decisions in the branch-and-bound algorithm and cutting plane algorithm and combining the two into a *branch-and-cut* algorithm. Branch-and-cut follows a procedure similar to that of branch-and-bound, but with the additional step at each node in the branch-and-bound tree of adding a set of cutting planes to the LP subproblem. Typically, the addition of cutting planes results in improved bounds at the subproblem nodes, allowing nodes to be pruned earlier in the branch-and-bound tree and potentially reducing the number of nodes that must be explored. Many MIPs also have structure that can be exploited through decomposition methods, initial cut generation, or reformulations that might greatly reduce the solution time. As a result, although theoretically challenging, commercial MIP solvers can solve many types of MIPs quickly. For example, the popular commercial MIP solver Gurobi demonstrated over 20x speed-up in solving MIP models in the four years between its release in 2009 and 2013 (Bixby (2012); Gurobi Optimization (2018)). Between 1991 and 2010, there was an overall MIP solver speed-up due to algorithmic improvements of about 580,000x (Bixby (2012)). Combined with machine improvements in that same time, the overall speed increase for solving MIPs is over 10^9 (Bertsimas (2014)). Many MIPs that were unsolvable in the early 2000's can now be solved almost instantaneously. The key conclusion to draw from these facts is that although originally impractical to solve real problems with MIP models, in practice many problems can in fact be modeled as MIPs and solved efficiently.

1.3 Stochastic Programming

We provide a brief introduction to stochastic programming (also known as stochastic optimization), which involves making decisions under uncertain data. By using specialized modeling and solution techniques, we aim to optimize the expected outcome of decision problems given the uncertainty of the future.

Stochastic Programming Modeling

First, we describe the structure of a stochastic programming model.[2] We present a two-stage formulation; multi-stage stochastic problems present additional complexity and are beyond the scope of this introduction. In a two-stage stochastic program, a decision-maker makes some decisions (*first-stage decisions*), then a random event occurs, and finally the decision-maker makes additional decisions to account for the random event (*recourse* or *second-stage decisions*).

Let $c \in \mathbb{R}^n$ be a vector of deterministic costs. Let $\boldsymbol{\xi}$ be a random vector on a probability space $(\Omega, \mathbb{P}, \Sigma)$ such that $\boldsymbol{\xi} : \Omega \to \mathbb{R}^r$. Let $X \subseteq \mathbb{R}^n$ be the set of deterministic (i.e., first-stage) constraints. Following is the formulation of a two-stage stochastic programming problem:

$$\min_{x \in X} \mathbb{E}[f(x, \boldsymbol{\xi})], \tag{SP}$$

where $f(x, \boldsymbol{\xi}) := c^T x + Q(x, \boldsymbol{\xi})$ and for a given outcome ξ and a given first-stage solution x, the *recourse function* or *value function* Q is defined as

$$\begin{aligned} Q(x, \xi) := \min\ & q(\xi)^T y \\ \text{s.t } & W(\xi) y = h(\xi) - T(\xi) x \\ & y \in Y(\xi). \end{aligned}$$

[2] Notation and information in this section are adapted from Kall et al. (1994) and Shapiro et al. (2009).

$Y(\xi) \subseteq \mathbb{R}^m$ is the feasible region defined by all the second-stage constraints, which could depend on ξ. Note that X or $Y(\xi)$ could include integer restrictions.

A useful metric that gives an indication of how much value is gained by using a stochastic model rather than a deterministic model is the *value of the stochastic solution (VSS)*, which is defined as the difference between the expected cost of the *mean-value solution* and the optimal objective value of SP. Suppose z_S is the optimal objective value of (SP). The mean value solution (the solution to the problem where all data are given as their expected values) is defined as

$$x_{MV} \in \arg\min_{x \in X} f(x, \mathbb{E}[\boldsymbol{\xi}]).$$

When the objective function of (SP) is evaluated using the solution x_{MV}, the objective value is

$$z_{MV} := \mathbb{E}[f(x_{MV}, \boldsymbol{\xi})].$$

Then the VSS is defined as

$$VSS := z_{MV} - z_S.$$

A solution to the mean-value problem provides an upper bound on z_S, so $VSS \geq 0$.

Another useful metric is the *expected value of perfect information (EVPI)*, which gives a measure of how much a decision maker would be willing to pay in order to obtain data which is closer to the true data (i.e., how much is gained by knowing the random outcomes in advance). The *perfect information problem* (also known as the *wait-and-see* problem) is defined as

$$z_{PI} := \mathbb{E}[\min_{x \in X} f(x, \boldsymbol{\xi})]$$

and the EVPI is

$$EVPI := z_S - z_{PI}.$$

A solution to the perfect information problem provides a lower bound on z_S, so $EVPI \geq 0$.

These metrics give some understanding of how valuable the stochastic formulation is relative the the deterministic formulation and the available data. The VSS indicates the relative value of the stochastic model over the (much more efficiently solved) deterministic model. The EVPI indicates how much the stochastic solution could be improved with more certainty about the realizations of the random data (Kall et al. (1994)).

Solving Stochastic Programming Problems

In general, SP can be hard to solve, as evaluating the expected value may require high-dimensional integration and there are possibly infinitely many realizations of $\boldsymbol{\xi}$. However, a common special case of the distribution of ξ is a finite list of *scenarios*, where each scenario is a realization of ξ. Then $\Omega = \{\xi^1, \ldots, \xi^K\} \subseteq \mathbb{R}^r$ with $\mathbb{P}(\boldsymbol{\xi} = \xi^k) = p_k$ for $k = 1, \ldots, K$. In this case, there exists an explicit reformulation of (SP) called the *extensive form*.

Because both the first and second stage problems are minimization problems, we create a separate copy of the vector y^k for each scenario $k = 1, \ldots, K$ and include all K recourse problems in the extensive form as follows:

$$\begin{aligned}
&\min c^T x + \sum_{k=1}^{K} p_k q(\xi^k)^T y^k \\
&\text{s.t. } x \in X \\
&\quad T(\xi^k)x + W(\xi^k)y^k = h(\xi^k)\ \forall k = 1, \ldots, K \\
&\quad y^k \in Y^k \qquad\qquad\qquad\qquad \forall k = 1, \ldots, K.
\end{aligned}$$

This problem can then be solved using any deterministic optimization methods relevant

to the problem structure, such as LP, MIP, or nonlinear methods. However, when K is large, the extensive form may become intractable. A number of techniques exist to solve or obtain bounds on stochastic programming problems that are too large to solve directly. Many solution techniques, such as Benders decomposition or dual decomposition algorithms, involve decomposing the problem into smaller subproblems. We do not give details of these algorithms here, as they are not necessary for the comprehension of the material in the chapters to follow. For more information, we direct the reader to Kall et al. (1994) and Shapiro et al. (2009).

Sample Average Approximation

For the case in which Ω is not finite (or is too large to handle directly), a technique called sample average approximation (SAA) can be used to obtain a finite list of samples of Ω and a confidence interval on the optimal solution to (SP).[3] Suppose we sample K independent identically distributed (iid) realizations of the random variable $\boldsymbol{\xi}^1, \cdots, \boldsymbol{\xi}^K$. The SAA problem is defined as

$$z_K^* = \min_{x \in X} \frac{1}{K} \sum_{j=1}^{K} f(x, \xi^j). \tag{SP_K}$$

It is possible to make some important statements about how closely a solution to (SP_K) maps to a solution to the true problem (SP). To do so, the following assumptions are required:

A1: X is nonempty and compact

A2: $\mathbb{E}[f(\cdot, \boldsymbol{\xi})]$ is lower semicontinuous.

Theorem 1.1. *Assume A1, A2, and* $\mathbb{E}\left[\frac{1}{K}\sum_{j=1}^{K} f(x, \xi^j)\right] = \mathbb{E}[f(x, \boldsymbol{\xi})]$ *for all* $x \in X$. *Then* $\mathbb{E}[z_K^*] \leq z^*$. *If* ξ^i *for* $i = 1, \ldots, K$ *are iid, then* $\mathbb{E}[z_K^*] \leq \mathbb{E}[z_{K+1}^*]$.

We refer the reader to Shapiro et al. (2009) for the proof of this theorem.

[3] The notation and information in this section are adapted from Shapiro et al. (2009).

Thus, solving (SP_K) yields a lower bound in expectation on the optimal solution to (SP) and the gap $z^* - \mathbb{E}[z_K^*]$ is a non-increasing function of K. Furthermore, under some additional assumptions, Shapiro et al. (2009) show that the Law of Large Numbers implies $z_K^* \to z^*$ as $K \to \infty$. The authors further show that if the feasible set S is finite, the number of samples needed to find an objective value within ϵ of the optimal objective with probability at least $1 - \alpha$ is on the order of $(1/\epsilon^2)\log(|S|/\alpha)$. Mak et al. (1999) show that the optimal objective can be estimated statistically with a confidence interval by replicating SAA multiple times. For more details on what statements can be made regarding the SAA problem, we refer the reader to Shapiro et al. (2009).

1.4 Bilevel Programming

Bilevel programming is a special class of optimization that involves embedded optimization problems. Typically, bilevel programs represent a setting in which two or more decision makers are all working toward individual (possibly differing) objectives. A common type of bilevel program is the Stackelberg game. A Stackelberg game consists of a leader (P1) and a follower (P2). P1 makes the first decision, knowing that P2 will respond in a way that is optimal for P2 given P1's decision. After P1 makes his or her optimal choice, P2 makes the second decision, with knowledge of P1's decision (Von Stackelberg (1952)).

Bilevel programs present unique challenges both in modeling and solution techniques. Jeroslow (1985) showed that even the simplest bilevel programs with linear constraints and objectives are $\mathcal{NP}$-hard. In this section, we discuss a few key aspects of bilevel programs, including modeling perspectives and linear reformulations of certain classes of bilevel programs.

Modeling Bilevel Programs

The first challenge in working with bilevel programming problems is deciding how to model them.[4] Most bilevel programs have a consistent underlying structure. First, one decision maker selects a solution $x \in X \subseteq \mathbb{R}^n$. A second decision maker then has a set of feasible decisions $Y(x) \subseteq \mathbb{R}$ that could depend on the solution x. The second decision maker chooses a solution $y \in \arg\min_z\{h(x,z) : z \in Y(x)\}$ in response to the first decision maker's solution. The first decision maker's goal is to minimize an objective $f(x,y)$ while remaining feasible to the constraints $g(x,y) \leq 0$.

We assume that $f, g,$ and h are real-valued functions with appropriate dimensions. We refer to the y variables as the *lower-level decisions* and the x variables as the *top-level decisions.* Similarly, the embedded optimization problem is the *lower-level problem* and the rest of the problem is the *top-level problem.* A key choice in modeling the bilevel setting is how to deal with alternative solutions to the lower-level problem. The two options for handling this difficulty are *pessimistic* models and *optimistic* models.

Pessimistic bilevel programs could also be considered *robust*, as they require the top-level constraints to hold for *all* possible lower-level optimal solutions. A solution to a pessimistic model typically overestimates the objective value the top-level decision maker will receive but provides a guaranteed upper bound on the top-level objective function.

In particular, the pessimistic bilevel problem is

$$\begin{aligned} &\min_{x,\eta} \ \eta \\ &\text{s.t. } g(x,y) \leq 0,\ \eta \geq f(x,y) \qquad \forall y \in \arg\min_z\{h(x,z) : z \in Y(x)\} \\ &\quad x \in X. \end{aligned}$$

On the other hand, optimistic bilevel programs only require the existence of a solution to the lower-level problem that satisfies the top-level constraints. This allows the top-level

[4] Notation and information in this section are adapted from Dempe (2002) and Colson et al. (2007).

decision maker to assume the lower-level decision maker will choose an optimal solution that is favorable to the top-level decision maker, and thus yields a lower bound on the objective value the top-level decision maker will receive.

The optimistic bilevel problem is

$$\begin{aligned}
&\min_x f(x,y)\\
&\text{s.t. } g(x,y) \leq 0\\
&\qquad y \in \arg\min_z \{h(x,z) : z \in Y(x)\}\\
&\qquad x \in X.
\end{aligned}$$

Typically, pessimistic bilevel programs are much more challenging to solve than optimistic bilevel programs. In fact, it can be shown that under certain conditions, optimistic bilevel programs are guaranteed to have optimal solutions, whereas no such guarantee can be made for pessimistic bilevel programs in general (Dempe (2002)).

A special case of bilevel programming known as *max-min* programming occurs when the lower-level objective is directly opposed to the top-level objective (i.e., the top-level objective is $\min_x f(x,y)$ and the lower-level objective is $\max_y f(x,y)$). That is to say, the top-level objective is to minimize the worst-case solution to the lower-level problem. A solution to this problem yields an upper bound on the objective value of the top-level decision maker, since the the top-level decision maker assumes the worst possible lower-level solution for his or her objective.

Reformulating Bilevel Problems as Linear Programs with Complementarity Constraints

If both the top- and lower-level problems have linear objectives and constraints and there are no lower-level integer variables, it is possible to reformulate an optimistic bilevel

program as a single-level mixed-integer linear program (MILP) using linear programming theory. Specifically, it is possible to use the Karush-Kuhn-Tucker conditions to replace the embedded lower-level problem with a set of linear necessary and sufficient optimality conditions (Labbé et al. (1998)). Let $f(x,y) = c_1^T x + d_1^T y$, $h(x,y) = c_2^T x + d_2^T y$, and $Y(x) = \{y \in \mathbb{R}^p : Ax + By \geq b\}$. Further, let λ be the dual variables associated with the lower-level constraints, $Ax + By \geq b$ for a fixed x. Then the lower level dual problem is

$$\begin{aligned} \max_{\lambda} \ & \lambda^T(b - Ax) \\ \text{s.t. } & \lambda B = d_2 \\ & \lambda \geq 0. \end{aligned}$$

The full single-level bilinear formulation of a bilevel program is

$$\min_{x,y,\lambda} c_1^T x + d_1^T y \tag{1.1}$$

$$\text{s.t. } Ax + By \geq b \tag{1.2}$$

$$\lambda B = d_2 \tag{1.3}$$

$$\lambda \geq 0 \tag{1.4}$$

$$\lambda(Ax + By - b) = 0. \tag{1.5}$$

Constraint (1.2) enforces primal feasibility and constraints (1.3)-(1.4) enforce dual feasibility. Constraint (1.5) is the complementary slackness condition for primal-dual optimality. Though this is a bilinear constraint set, because either λ_i or $(Ax + By - b)_i$ is 0 for every element i, it is possible to replace these nonlinear equations with linear inequalities by introducing binary variables indicating which term is 0. Note that this requires the dual variables to be bounded, but typically this requirement is not overly restrictive (Labbé and Violin (2013)).

1.5 Multi-Objective Programming

Multi-objective programming problems are problems that have two or more potentially competing objectives.[5] We represent such a problem as

$$\min_x \{f_1(x), f_2(x), \ldots, f_t(x)\}$$
$$\text{s.t. } x \in X \subseteq \mathbb{R}^n,$$

where each f_i corresponds to one of t different objective functions.

In these problems, decision makers are usually interested in discovering a full portfolio of optimal solutions demonstrating the trade-offs between the objectives rather than a single solution. In particular, decision makers aim to find a set of *Pareto optimal* or *Pareto efficient* points. For simplicity, we assume $t = 2$ unless otherwise specified, though all material presented in this section can be extended to cases in which $t > 2$. Pareto optimality for a point x^* is defined by the condition that there is no other point x in the feasible set such that $f_1(x) \leq f_1(x^*)$, $f_2(x) \leq f_2(x^*)$, and $f_i(x) < f_i(x^*)$ for at least one of $i = 1, 2$. That is to say, a point is Pareto optimal if we cannot decrease one objective without increasing the other.

Finding Solutions to Bi-Objective Problems

A common way to find Pareto solutions for a multi-objective problem is to minimize a weighted sum of the functions:

$$\min_x f_1(x) + \lambda f_2(x)$$
$$\text{s.t. } x \in X \subseteq \mathbb{R}^n.$$

[5] Notation and information in this section are taken and adapted from Chankong and Haimes (1982), Coverstone-Carroll et al. (2000), and Ehrgott (2006)

By varying the weight λ , a subset of the Pareto optimal points can be discovered. If the feasible region is non-empty, a solution can be found for any value of λ. Although simple, this modeling approach only yields solutions on convex envelope of the Pareto frontier. Thus, if the Pareto frontier is nonconvex, we might miss large regions of the frontier. The Pareto frontier is convex if the feasible region $X \subseteq \mathbb{R}^n$ and the objective functions f_i for $i = 1, 2$ are convex, which is unfortunately not the case if there are integer restrictions on any of the variables (Ehrgott (2006)).

To combat this drawback of the weighted-sum approach, an alternative technique (sometimes called the *ϵ-constraint technique*) can be used to constrain one of the objectives by a parameter ℓ:

$$\begin{aligned} &\min_x \ f_1(x) \\ &\text{s.t. } f_2(x) \leq \ell \\ &\qquad x \in X \subseteq \mathbb{R}^n. \end{aligned}$$

The bound ℓ can be varied to find points all along the efficient frontier, not just on the convex envelope. Chankong and Haimes (1982) and Ehrgott (2006) showed that the set of Pareto optimal points are the same for the constraint-based method and the original problem of minimizing both f_1 and f_2. Furthermore, *all* efficient points can be found by choosing appropriate values of ℓ. Of course, the roles of the functions f_1 and f_2 can be swapped, depending on which choice makes the model more easily solvable. Caution should be exercised to choose ℓ such that the problem remains feasible.

When the relative importance of the t objectives is known, one method which will find a single solution to a multi-objective problem is a hierarchical solution approach. In this method, only the most important objective is minimized, with no constraints on any other objectives. The solution is used to introduce a bound constraint on the first objective value, then the second-most important objective is minimized. The process is repeated

until all t objectives have been iteratively optimized and constrained. This method ensures that the most important objectives are prioritized. Bounds must be chosen such that each successive problem remains feasible and sufficient (but not too much) flexibility is given to each objective. Note however that this method only yields one solution, so if an efficient frontier is needed, hierarchical methods are not preferred.

Stochastic Multi-Objective Programming

Finally, we provide a short overview of stochastic multi-objective programming.[6] Stochastic multi-objective programs (SMOPs) present a unique challenge as both a stochastic and a multi-objective optimization problem. One major challenge is deciding how to model a SMOP in a well-defined manner.

When formulating a SMOP, the first decision to be made is whether to treat the SMOP as a multi-objective problem consisting of multiple stochastic objectives or a single stochastic problem that aggregates the multiple objectives in some way. In particular, the decision to be made is how to formulate a problem of the form

$$\begin{aligned} &\min_{x} \{f_1(x, \boldsymbol{\xi}), f_2(x, \boldsymbol{\xi}), \ldots, f_t(x, \boldsymbol{\xi})\} \\ &\text{s.t. } x \in X \subseteq \mathbb{R}^n, \end{aligned} \tag{SMOP}$$

where $\boldsymbol{\xi}$ is a random outcome (as defined in the prior section on stochastic programming) that could affect any number of the t objectives. As written, SMOP is not fully defined because each function f_i is itself a real-valued random variable, and minimizing t different random variables has little meaning. For the sake of this discussion, we assume all constraints are deterministic; in general, some constraints could also depend on $\boldsymbol{\xi}$. There are two ways of reducing SMOP into a well-defined problem – either a multi-objective deterministic equivalent or a single objective stochastic problem.

[6] Notation and information in this section are adapted from Gutjahr and Pichler (2016).

(i) The first reduction is known as the *multi-objective method*. This method relies on reducing the SMOP to a multi-objective problem so it can be solved by multi-objective programming techniques. In the multi-objective method, a function $\mathcal{F}_j$ is defined for the random variable $f_j(x, \boldsymbol{\xi})$, usually some summary statistic of the variable (such as expectation or variance). The decision maker then solves the deterministic problem

$$\begin{aligned} &\min_x \{Z_1(x), Z_2(x), \ldots, Z_t(x)\} \\ &\text{s.t. } x \in X \subseteq \mathbb{R}^n, \end{aligned}$$

where $Z_j(x) := \mathcal{F}_j(f_j(x, \boldsymbol{\xi}))$. The simplest example of the multi-objective method is the expected value approach, where $\mathcal{F}_j(f_j(x, \boldsymbol{\xi})) = \mathbb{E}[f_j(x, \boldsymbol{\xi})]$. One drawback of this method is that it requires reducing all the distributional information to a set of summary statistics and thus could oversimplify the given SMOP.

(ii) The second reduction is known as the *stochastic method*. This method relies on reducing the SMOP to a single-objective stochastic program, to which stochastic programming methods can be applied. In the stochastic method, the multiple objectives are aggregated by a single function $u : \mathbb{R}^t \to \mathbb{R}$. Then the decision maker solves the single objective stochastic program

$$\begin{aligned} &\min_x u(f_1(x, \boldsymbol{\xi}), \ldots, f_t(x, \boldsymbol{\xi})) \\ &\text{s.t. } x \in X \subseteq \mathbb{R}^n, \end{aligned}$$

As written, the objective function is still not fully defined. Instead, the decision maker could minimize the expected value $\mathbb{E}[u]$ (or some other deterministic real-valued function of the random variable). In this way, the dependencies between the objectives can be preserved, but has the rather significant drawback that the function u (e.g., the decision maker's utility function), must be known in advance.

2 OPTIMIZING TRUCK DISPATCHING DECISIONS IN OPEN-PIT MINES USING INTEGER PROGRAMMING

2.1 Introduction

We propose a MIP model of an open-pit mine truck dispatching problem that introduces new approaches for modeling the effects of queueing and blending. Our primary contribution is a demonstration that the model can yield implementable solutions in real-time when combined with a relaxation induced neighborhood search heuristic.

Truck Dispatching in Open-Pit Mines

An open-pit mine is a large area in which trucks move ore between different locations. The mine consists of mining sites and processing sites. Ore is extracted from the mining sites and placed in large trucks that carry the ore to the processors. When trucks arrive at the processors, they dump the ore into short-term stockpiles from which ore is fed into the processor at an (ideally) fixed rate. Once processed, the ore is taken outside the mine to be sold or converted into other products. The ore loaded into the processor should be of a certain quality, but keeping the processors fed and extracting a sufficient amount of ore from each mining site are the top priorities. If a processor must be shut down, revenue that would be generated from the processed ore is lost.

To make routing decisions during the daily operations of the mine, a dispatcher typically has access to the mine's processing, mining, and quality targets and knows the current state of the system. When a truck completes a task, it must receive a new destination from the dispatcher. Each dispatching decision should make progress in meeting the mine's mining, processing, and ore quality goals. These goals tend to compete, forcing decision makers to find an acceptable balance between them.

The dispatching problem has a number of complicating components. The open-pit

mine is a dynamic stochastic environment, meaning dispatching decisions can quickly become obsolete. Furthermore, decisions must be made in real time to prevent trucks from stopping their loading or unloading activities while they are awaiting their next assignment. Due to limited loading and unloading capacity at each site, trucks often enter a queue before they can be loaded or unloaded. In addition, because different mining sites have different ore qualities, decision makers must account for the effects of blending ore from different sites. Finally, the dispatching problem has three potentially competing objectives: meeting processing rate targets, meeting mining extraction targets, and meeting ore quality targets. Because decision makers must balance these competing goals, it is difficult to create a dispatching rule that optimizes all three objectives concurrently. The combination of these factors makes the dispatching problem difficult to model and challenging to solve.

Literature Survey

A number of different truck dispatching rules have been used for truck dispatching. One method is to solve a medium-term rate problem that assigns flow rates of trucks to different routes within the mine, then to use myopic simulation or assignment problems based on the the target rates to make dispatching decisions. The first part of this approach – the rate-setting problem – has been explored by a number of authors. One approach is to use simulation or queueing theory to produce schedules (Kappas and Yegulalp (1991); Oraee and Asi (2004); Mena et al. (2013)). Other authors use optimization models and techniques to solve the rate problem, including mixed-integer programming (MIP) models (Goodman and Sarin (1988); Naoum and Haidar (2000); Burt (2008); Faraji (2013); Smith and Wicks (2014)), stochastic optimization models (Ta et al. (2005); Topal and Ramazan (2012)), or rate-setting linear programming (LP) models (Ercelebi and Bascetin (2009); Choudhary (2015)). Finally, Weintraub et al. (1987); Rubio (2006) and McKenzie et al. (2008) build optimization models that treat the truck allocation problem as a network.

Given target flow rates, a number of different approaches exist to solve the dispatching

problem in real time. Early work (Elbrond and Soumis (1987a); Soumis et al. (1989); White (1991); White et al. (1993)) considers integrating the rate-setting problem with the immediate dispatching decision by using LP and assignment or matching problems for the real-time decisions. Temeng (1998) models dispatching with goal constraints to enforce ore quality. In this approach, goal programming is employed to meet both volume and quality targets. Temeng demonstrates significant improvement in ore quality compared to traditional LP models. Ahangaran et al. (2012) build a binary programming model with an objective to minimize the cost of loading and transportation and constraining all other aspects of the system. Their model does not account for the timing of decisions; a decision is made at a fixed point in time without regard to the future state of the system. The constraints include hard bound constraints on the acceptable ore quality and ore volume. The authors also present an algorithm for solving the model.

Heuristic dispatching rules are commonly used to determine where a truck should go next. These rules are often embedded in a simulation in order to demonstrate how the current decision will affect the future state of the system. Subtil et al. (2011) find the optimal number of trucks within the mine and then use a simulation-based heuristic that combines simulation and a multi-criterion LP model. One heuristic, the 1-truck-for-N-shovels strategy, compares the benefit and cost of sending an available truck to any of the N mining sites but ignores the potential effects of the given truck's interactions with other trucks, such as queueing (Chatterjee and Brake (1981); Tu and Hucka (1985); Lizotte and Bonates (1987); Sadler (1988); Li (1990); Bonates (1992); Forsman et al. (1993); Panagiotou and Michalakopoulos (1995); Ataeepour and Baafi (1999)). A related strategy known as the M-trucks-for-1-shovel strategy ranks mine sites by how much additional material needs to be removed to meet the target and finds the next M trucks to be dispatched. This strategy ignores many of the interaction effects between system components over time (White et al. (1982); Arnold and White (1983)). Finally, it is possible to combine these two strategies into an M-trucks-for-N-shovels strategy to better account for the interconnected nature of

the mining system. Hauck (1973, 1979) and Munirathinam and Yingling (1994) suggest single-stage systems to determine a sequence of dispatching decisions by solving a series of assignment problems. Elbrond and Soumis (1987b) and Temeng et al. (1997) consider a multi-stage system but do not allow assigning multiple trucks to a single location and cost calculations do not account for future queueing effects if multiple trucks are routed to the same location.

Contributions

To solve the dispatching problem, we propose a deterministic MIP model which is solved over a medium-term planning horizon, such as 30 minutes. The planning horizon is discretized into many short intervals, usually 30 seconds long. By using a medium-term time-discretized MIP model as opposed to myopic heuristics or matching problems, we account for the future state of the system and reduce the effects of rounding in short- to medium-term decision making. Ultimately, our model finds truck dispatching decisions that meet the volume and quality targets while incorporating all the necessary problem characteristics. Our proposed model has components which are similar to those of many existing models but to the best of our knowledge is the first general mixed-integer real-time dispatching model that considers multiple sites of each type and multiple trucks.

We build our model in such a way as to account for as many of the challenges of the dispatching problem as possible. A drawback of the existing strategies is that they must ignore future effects of queueing at different locations in the mine. A contribution of our approach is the incorporation of a family of constraints which model the effects of queueing by approximating the maximum number of trucks that can be loaded or unloaded at each site within a window of time. In addition, due to the nonconvex nature of the standard blending model, we introduce a linear approximation of the effects of blending different ore qualities from different mining sites. We do this by penalizing the violation of the ore quality targets within multiple overlapping windows rather than tracking ore quality in

each period. With a hierarchical solution methodology to balance the three competing objectives of meeting processing rate, total extraction, and ore quality targets, our approach to constraining and optimizing ore quality and volume targets also differs from the existing dispatching models. We implement this hierarchy by solving three subproblems, iteratively optimizing and constraining the processing, mining, and ore quality objectives. Finally, this model can be used to make decisions in the dynamic stochastic environment of an open-pit mine. Through the use of a relaxation induced neighborhood search heuristic algorithm, we demonstrate that we can obtain good solutions to instances of the model quickly, much faster than simply solving the instances with standard optimization techniques but with similar solution quality. Ultimately, this model can be implemented in an ever-changing open-pit mining environment by re-solving it frequently and updating the data used in the model. A solution of the model yields a dispatching policy for each location within the mine by giving a list of the next locations to which available trucks should be sent. In Chapter 3, we test the dispatching policy derived from the MIP dispatching model in a discrete event simulation.

In Section 2.2, we give a detailed description of our proposed MIP model of the dispatching problem, including a hierarchical approach to optimizing and constraining the three competing objectives. In Section 2.3, we present the Relaxation Induced Neighborhood Search heuristic that is used to quickly obtain solutions to the challenging third phase of the dispatching problem. Finally, in Section 2.4, we give a comprehensive computational study of our proposed model and methods. The first part of the study evaluates different methods for solving the dispatching problem; the second part demonstrates how our novel approach to approximating the loading and unloading capacity at each site affects the solution.

2.2 An Integer Programming Model of the Dispatching Problem

Our goal is to create a medium-term decision model that can be solved at regular intervals to make dispatching decisions. Given target processing volumes, mining volumes, and ore qualities, solving the model yields decisions to meet the targets as closely as possible, while obeying physical constraints on the system. The horizon should be long enough that the current decisions take into account the future state of the system, but short enough that a solution can be obtained quickly. Decision variables in the dispatching model determine the violation of extraction, processing rate, and ore quality targets; the number of trucks queued, loaded and unloaded, and available at each site within the mine; the number of trucks traveling between pairs of locations in the mine; and the volume of ore in the short-term stockpiles. Constraints in the dispatching model measure violation of the target parameters, balance flow into and out of locations in the mine, bound the total trucks that can be loaded (unloaded) at each site in the mine in a given window of time, and track the short-term stockpile level over time. Decision variables associated with numbers of trucks assume integer values.

The objective is to minimize the violation of all three targets: processing rate, extraction, and ore quality, with the greatest emphasis given to meeting the processing rate targets. To model the three objectives, the MIP dispatching model is decomposed into three subproblems. The first subproblem (P1) minimizes deviation from processing rate targets. The second subproblem (P2) uses the solution to P1 to upper bound the processing rate violation and minimizes the maximum deviation from mining extraction targets. The third and final subproblem (P3) uses the (P1) and (P2) solutions to upper bound the processing rate violation and mining target violation (respectively) and minimizes the deviation from the target ore quality.

Sets and Parameters

We begin by describing the necessary notation of the model. The mine consists of a set of locations, N. The set N is partitioned into mining sites M, processing sites P, dump sites D, and long-term stockpile sites S. The set M is partitioned into mines where ore is extracted (ore mines) G and mines where overburden is extracted (waste mines) B. Material taken from ore mines is of high enough quality that it can be processed or taken to a long-term stockpile. In our model, we treat long-term stockpiles as storage sites for material from ore mines, so trucks at ore mine sites can be dispatched to either a processing site or a long-term stockpile. Instead of just viewing them as storage sites, long-term stockpiles could alternatively be used as sites from which material can be removed for processing. Our model could be extended to include the option to treat long-term stockpiles in this way, but we do not consider that option in this work. Material taken from waste mines is of poor quality or is waste that must be removed in order to reach the ore. Material from waste mines is taken to a dump site. The ore has a set of quality attributes R that are monitored. We are also given a set of truck sizes K.

We model an open-pit mine for certain length of time $T^{\max}$, using a set of discrete time periods $T := \{1, 2, \ldots, T^{\max}\}$. We extend the horizon to include some past time periods for ease of notation and denote the extended horizon by $T^H = \{-\max_{ij} \tau_{ij}, \ldots, 0, 1, , T^{\max}\}$, where τ_{ij} represents the number of time periods required to travel between location $i \in N$ and location $j \in N$.

The input data for the dispatching problem describe the current state of the system. Each truck type $k \in K$ can carry α_k tons of ore. Trucks travel between pairs of locations, and each route has a parameter representing travel time in number of periods, τ_{ij} for $i \in N$, $j \in N$. The average number of trucks that can be loaded or unloaded in each period is $\bar{p}_i$ for all $i \in N$. Note that we do not assume $\bar{p}_i$ is an integer for any $i \in N$. Each ore mine $i \in G$ has parameters Q^M_{ir} representing the quality attribute percentage for each $r \in R$. Although the true ore quality is uncertain, it is reasonable to assume this value can be estimated for

short-term planning horizons based on recent observations of ore quality. We also have total extraction targets for the entire planning horizon (in tons) for every mining site $i \in M$, Θ_i^M. Each processing site $j \in P$ has a processing rate Θ_{jt}^P, which should be a constant rate of depletion, in tons/period, unless the short-term stockpile becomes empty, in which case the processor must stop. For each processing site $j \in P$, the short-term stockpiles have an initial amount of ore, I_{j0}, and an initial attribute percentage, Q_{jr0}^I, for all $r \in R$. There are ore quality targets Q_{jr}^P at all $j \in P$ and for all $r \in R$. Trucks of type $k \in K$ en route from $i \in N$ to $j \in N$ that began their trip at period $t \in T^H \setminus T$ are given by $\hat{x}_{ijkt}$. Trucks of type $k \in K$ available for dispatching at site $i \in N$ at the start of the planning horizon are denoted by $\hat{g}_{ik}$. Finally, trucks that are in queue at period 0 at mining site $i \in M$ are given by $\hat{q}_{ik}^M$ and trucks in queue at processing site $j \in P$ that have arrived from mine $i \in M$ are given by $\hat{q}_{ijk}^P$. These sets and data are summarized in Tables 2.1 and 2.2 (respectively).

Set	Description
T	Time periods in the planning horizon ($t \in \{1, ..., T^{\max}\}$)
T^H	Time periods in the planning horizon with some history added to account for trucks being loaded or unloaded or en route at $t = 0$ ($t \in \{-\max_{ij} \tau_{ij}, ..., T^{\max}\}$)
M	Mining sites ($i \in M$)
G	ore mining sites
B	waste mining sites (note that $B \cup G = M$)
D	Dump sites for waste removed from a waste mining site $i \in B$ ($j \in D$)
P	Processing sites ($j \in P$)
S	Stockpiles where material taken from ore mines can be stored ($j \in S$)
N	$M \cup D \cup P \cup S$: all locations
R	Attributes to be monitored ($r \in R$)
K	Truck types ($k \in K$)

Table 2.1: A summary of all sets used in the MIP dispatching model

Parameter	Description	Units
α_k	Size of truck type $k \in K$	Tons/truck
$\bar{p}_i$	Maximum number of trucks that can be processed per period at site $i \in N$ (may be fractional)	Trucks/period
Θ_i^M	Amount that should be extracted from mine $i \in M$ during the planning horizon	Tons
Θ_{jt}^P	Amount that should be processed by processing site $j \in P$ during period t (typically the same for all t)	Tons
I_{j0}	Amount in short-term stockpile $j \in P$ at period 0	Tons
C_j	Maximum capacity of short-term stockpile $j \in P$	Tons
τ_{ij}	Number of periods needed to travel from $i \in N$ to $j \in N$	Time
Q_{ir}^M	Ore attribute $r \in R$ percentage at mine $i \in G$	(%)
Q_{jr}^P	Target ore attribute $r \in R$ percentage of the ore at processing site $j \in P$	(%)
Q_{jr0}^I	Ore attribute $r \in R$ percentage of the ore in short-term stockpile $j \in P$ at period 0	(%)
$\hat{x}_{ijkt}$	# of trucks of type k sent from $i \in N$ to $j \in N$ at period $t \in T^H \setminus T$	Trucks
$\hat{g}_{kj}$	# of trucks of type $k \in K$ available at $j \in N$ at $t = 0$	Trucks
$\hat{q}_{ik}^M$	# of trucks of type $k \in K$ waiting in a queue at mine $i \in M$ at period $t = 0$	Trucks
$\hat{q}_{ijk}^P$	# of trucks of type $k \in K$ waiting in a queue at $j \in P$ holding load from mine $i \in G$ at period $t = 0$	Trucks

Table 2.2: A summary of all parameters and data used in the MIP dispatching model

Decision Variables and Constraints

The dispatching problem includes decision variables to describe the new state of the system and variables indicating where trucks should be sent in each period. In particular, x_{ijkt} is the number of trucks sent from $i \in N$ to $j \in N$ of type $k \in K$ in period $t \in T^H$. The number of trucks loaded at $i \in M$ of type $k \in K$ in period $t \in T$ is represented by the variable y_{ikt}^M. The number of trucks of type $k \in K$ from mine $i \in G$ unloaded at $j \in P$ in period $t \in T$ is y_{ijkt}^P.

We balance queues of trucks both before and after each truck is loaded or unloaded at a site. In particular, at every mine, we have the structure depicted in Figure 2.1. All trucks that arrive enter a queue that they leave as soon as the loading site becomes available. A truck is loaded, then becomes available for dispatching. It is possible for the truck to wait

before it departs to another destination in the mine, so we also keep track of how many trucks are available at each site.

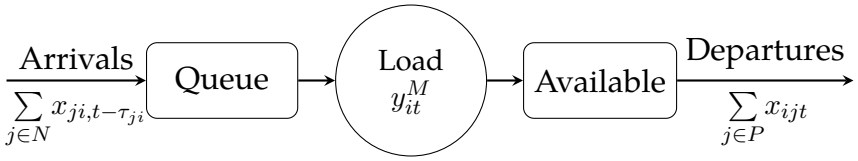


Figure 2.1: Example of the workflow at a mining site

Processing site queues have a similar structure, but there are separate queue and unloading variables for each possible origin (ore mining site), since each ore mining site could have different ore qualities.

The variables q_{ijkt}^{P} measure the number of trucks of type k queued at each site $j \in P$ from mining site $i \in M$ at time t. The following constraints are used to track the number of trucks queued at a processing site:

$$q_{ijkt}^{P} = q_{ijk,t-1}^{P} - y_{ijkt}^{P} + x_{ijk,t-\tau_{ij}} \qquad \forall k \in K, \forall i \in G, \forall j \in P, \forall t \in T^{H}. \tag{2.1}$$

Trucks waiting in a queue cannot be dispatched until after they are unloaded. Thus, the constraints require that for each processing site and for each mine, the number of trucks in queue at time t, q_{ijkt}^{P}, is the number of trucks in queue at time $t-1$ ($q_{ijk,t-1}^{P}$) minus the number of trucks unloaded at time t (y_{ijkt}^{P}), plus the number of trucks arriving at time t ($\sum_{i \in M} x_{ijk,t-\tau_{ij}}$). The subscript $t - \tau_{ij}$ on the flow variables accounts for travel time; if a truck leaves location i at time t, it will not arrive at its destination j until time $t + \tau_{ij}$.

For the mining sites, we do not need a separate queue variable for each truck's origin. We use q_{ikt}^{M} to track the number of trucks queued at mining site i of type k in time period t.

The queue balance constraints at the mining sites are then:

$$q^M_{ikt} = q^M_{ik,t-1} - y^M_{ikt} + \sum_{j \in N \setminus M} x_{jik,t-\tau_{ji}} \qquad \forall k \in K, \forall i \in M, \forall t \in T^H. \tag{2.2}$$

We initialize the model to fix the initial queue lengths to their current value.

$$q^P_{ijk,0} = \hat{q}^P_{ijkt} \qquad \forall i \in M, \forall j \in P, \forall k \in K \tag{2.3}$$

$$q^M_{ik,0} = \hat{q}^M_{ik} \qquad \forall i \in M, \forall k \in K. \tag{2.4}$$

We assume the unloading times at dump sites and long-term stockpiles are relatively small, so that queueing does not need to be modeled at these sites. If this were not true, the model could easily be modified to treat these sites the same as processing sites.

When a truck has finished being unloaded or loaded, it is ready to be dispatched to a new location. However, if there is no need for immediate dispatching, our model allows trucks to idle at a location until a later time. We use variables g_{ikt} to keep track of the number of trucks of type $k \in K$ at site $i \in N$ in period $t \in T$ that have completed their loading or unloading but have not yet been dispatched to a new location. The constraints to balance the number of available trucks at each site are as follows:

$$g_{jkt} = g_{jk,t-1} - \sum_{i \in M} x_{jikt} + \sum_{i \in G} y^P_{ijkt} \qquad \forall k \in K, j \in P, \forall t \in T^H \tag{2.5}$$

$$g_{jkt} = g_{jk,t-1} - \sum_{i \in M} x_{jikt} + \sum_{i \in B} x_{ijk,t-\tau_{ij}} \forall k \in K, j \in D, \forall t \in T^H \tag{2.6}$$

$$g_{jkt} = g_{jk,t-1} - \sum_{i \in M} x_{jikt} + \sum_{i \in G} x_{ijk,t-\tau_{ij}} \ \forall k \in K, j \in S, \forall t \in T^H \tag{2.7}$$

$$g_{ikt} = g_{ik,t-1} - \sum_{j \in P \cup S} x_{ijkt} + y^M_{ikt} \qquad \forall k \in K, i \in G, \forall t \in T^H \tag{2.8}$$

$$g_{ikt} = g_{ik,t-1} - \sum_{j \in D} x_{ijkt} + y^M_{ikt} \qquad \forall k \in K, i \in B, \forall t \in T^H. \tag{2.9}$$

At each processing site, the number of trucks available for dispatching at time t is the number of trucks available at time $t-1$ ($g_{jk,t-1}$), minus the number of trucks that are sent back to the mines at t ($\sum_{i \in M} x_{jikt}$), plus the number of trucks that are unloaded at time t ($\sum_{i \in G} y^P_{ijkt}$). Similar constraints exist for each of the different sites in the mine.

We initialize the model with parameters indicating the current availability of trucks.

$$g_{ik,0} = \hat{g}_{ik} \qquad \forall i \in N, \forall k \in K. \tag{2.10}$$

We track the amount of ore in the short-term stockpile at each processing site using variables I_{jt} for the tons of ore at processing site $j \in P$ at the end of time period $t \in T$. We monitor the amount of ore by keeping track of the ore added each period and the amount of ore removed (processed) in each period, which should be the parameter Θ^P_j. We introduce variables h^L_{jt} that measure by how much we violate the processing rate target at each site $j \in P$ and each time period $t \in T$. We model the amount of ore in the short-term stockpiles

with the following set of constraints:

$$I_{jt} = I_{j,t-1} - (\Theta^P_{jt} - h^L_{jt}) + \sum_{k \in K, i \in G} \alpha_k y^P_{ijkt} \forall t \in T, \forall j \in P \tag{2.11}$$

$$h^L_{jt} \leq \Theta^P_{jt} \qquad \forall t \in T, j \in P \tag{2.12}$$

$$I_{jt} \leq C_j \qquad \forall t \in T, j \in P \tag{2.13}$$

$$I_{jT^{\max}} \geq I_{j0} \qquad \forall j \in P \tag{2.14}$$

Constraints (2.11) say the amount in the short-term stockpile at time t must equal the amount in the short-term stockpile at time $t-1$ minus the amount removed plus the amount added. The desired rate of removal of material from the short-term stockpile is the parameter Θ^P. Ideally, this rate is always met and the h^L variables are 0. However, in some rare cases this might not be possible, so the variables h^L measure any negative deviation from the target rate. Constraints (2.12) enforce non-negative processing rates for every processing site and every time period. Finally, constraints (2.13) enforce the short-term stockpile capacity restrictions, and constraints (2.14) attempt to alleviate end-of-horizon effects by requiring the stockpile level at $T^{\max}$ to be at least as large as the initial short-term stockpile level.

We fix each flow variable x_{ijkt} for $t \in T^H \setminus T$ to the number of trucks en route at that same time, $\hat{x}_{ijkt}$:

$$x_{ijkt} = \hat{x}_{ijkt} \qquad \forall i \in N, \forall j \in N, \forall k \in K, \forall t \in T^H \setminus T.$$

We also require that all decision variables are non-negative, and that truck availability,

loading/unloading, and flow variables are integral:

$$h^U_{it}, h^L_{it}, I_{jt}, q^M_{ikt}, q^P_{ijkt} \geq 0 \quad \forall i, j \in N, \forall k \in K, \forall t \in T, \forall r \in R \tag{2.15}$$

$$x_{ijkt}, g_{kjt}, y^P_{ijkt}, y^M_{ikt} \in \mathbb{Z}_+ \qquad \forall i, j \in N, \forall k \in K, \forall t \in T. \tag{2.16}$$

One of the difficulties of modeling the dispatching problem is incorporating the limited loading capacity at the mining sites and unloading capacity at the processing sites. We approximate the effect of the limited capacity by rounding up the maximum number of trucks that can be loaded (unloaded) at each site to the nearest integer:

$$\sum_{i \in G} \sum_{k \in K} y^P_{ijkt} \leq \lceil \bar{p}_j \rceil \quad \forall j \in P, \forall t \in T \tag{2.17}$$

$$\sum_{i \in M} \sum_{k \in K} x_{jikt} \leq \lceil \bar{p}_j \rceil \; \forall j \in D, \forall t \in T \tag{2.18}$$

$$\sum_{i \in M} \sum_{k \in K} x_{jikt} \leq \lceil \bar{p}_j \rceil \quad \forall j \in S, \forall t \in T \tag{2.19}$$

$$\sum_{k \in K} y^M_{ikt} \leq \lceil \bar{p}_i \rceil \qquad \forall i \in M, \forall t \in T. \tag{2.20}$$

We note that we could choose to round down instead, but this underestimates the capacity and if $\bar{p}_i < 1$ for some site $i \in N$ no trucks could ever be loaded or unloaded at that site. However, rounding up to the nearest integer in each period overestimates the rate at which trucks can be loaded or unloaded. We mitigate this overestimation by also implementing constraints over windows of time of length d_i, where $d_i \in \mathbb{Z}$ is chosen to minimize the rounding error of $\lceil d_i \bar{p}_i \rceil$ for each site $i \in N$. In particular, we choose d_i as follows:

$$d_i \in \arg\min \left\{ \frac{\lceil d\bar{p}_i \rceil - d\bar{p}_i}{d\bar{p}_i} : d \in \{1, \ldots, d^{\max}\} \right\}$$

for a fixed value $d^{\max}$. In our test instances, we choose $d^{\max} = 10$. Keeping the window

length d_i short ensures that the overlapping window constraints we describe next for modeling loading and unloading capacity are not dense.

Once we have a value of d for each site, we build a family of constraints that approximate capacity at different sites in the mine:

$$\sum_{s=t}^{t+d_i-1} \sum_{i\in G} \sum_{k\in K} y^P_{ijks} \leq \lceil d_j \bar{p}_j \rceil \quad \forall j \in P, \forall t \in \{1, .., T^{\max} - d_j + 1 : t > d_j\} \tag{2.21}$$

$$\sum_{s=t}^{t+d_j-1} \sum_{i\in M} \sum_{k\in K} x_{jiks} \leq \lceil d_j \bar{p}_j \rceil \forall j \in D, \forall t \in \{1, ..., T^{\max} - d_j + 1 : t > d_j\} \tag{2.22}$$

$$\sum_{s=t}^{t+d_j-1} \sum_{i\in M} \sum_{k\in K} x_{jiks} \leq \lceil d_j \bar{p}_j \rceil \ \forall j \in S, \forall t \in \{1, ..., T^{\max} - d_j + 1 : t > d_j\} \tag{2.23}$$

$$\sum_{s=t}^{t+d_i-1} \sum_{k\in K} y^M_{iks} \leq \lceil d_i \bar{p}_i \rceil \qquad \forall i \in M, \forall t \in \{1, ..., T^{\max} - d_i + 1 : t > d_i\}. \tag{2.24}$$

For $t = d_i, \cdots, T^{\max} - d_i + 1$, we require that the total number of trucks processed at site $i \in N$ in the time interval $[t, t + d_i - 1]$ is no more than $d_i \bar{p}_i$, rounded up to the nearest integer. Some slight modifications need to be made to these constraints at the beginning of the planning horizon:

$$\sum_{s=1}^{t} \sum_{i\in G} \sum_{k\in K} y^P_{ijks} \leq \lceil t \bar{p}_j \rceil \ \ \forall j \in P, \forall t \in T : t \leq d_j \tag{2.25}$$

$$\sum_{s=1}^{t} \sum_{i\in M} \sum_{k\in K} x_{jiks} \leq \lceil t \bar{p}_j \rceil \ \forall j \in D, \forall t \in T : t \leq d_j \tag{2.26}$$

$$\sum_{s=1}^{t} \sum_{i\in M} \sum_{k\in K} x_{jiks} \leq \lceil t \bar{p}_j \rceil \ \ \forall j \in S, \forall t \in T : t \leq d_j \tag{2.27}$$

$$\sum_{s=1}^{t} \sum_{k\in K} y^M_{iks} \leq \lceil t \bar{p}_j \rceil \qquad \forall i \in M, \forall t \in T : t \leq d_i. \tag{2.28}$$

For $t = 1, \ldots, d_i$, we require that the total number of trucks processed in the time interval $[1, t]$ is no more than $t \bar{p}_i$, rounded up to the nearest integer. By enforcing all constraints

(2.17) - (2.28), the model effectively approximates and constrains loading and unloading rate limits that are not integer-valued.

To simplify notation, let X be the set of solutions which satisfy all constraints:

$$X := \{v = (x,\ y^P,\ y^M\ q^M,\ q^P,\ g,\ h^L,\ I) : (2.1) - (2.28)\}.$$

Then a solution $v \in X$ is a feasible solution to the dispatching problem. The decision variables of the model are summarized in Table 2.3.

Variable	Description	Units
x_{ijkt}	# of trucks of type k that leave $i \in N$ traveling to $j \in N$ at time t	Trucks
y^P_{ijkt}	# of type $k \in K$ trucks holding load from $i \in G$ unloaded at $j \in P$ at time $t \in T$	Trucks
y^M_{ikt}	# of type $k \in K$ trucks loaded at $i \in M$ at time $t \in T$	Trucks
q^P_{ijkt}	# of trucks of type $k \in K$ waiting in a queue at $j \in P$ holding load from $i \in G$ at time $t \in T$	Trucks
q^M_{ikt}	# of trucks of type $k \in K$ waiting in a queue at mine $i \in M$ at time $t \in T$	Trucks
g_{kjt}	# of trucks of type $k \in K$ available at $j \in N$ at $t \in T$	Trucks
I_{jt}	Amount in short-term stockpile $j \in P$ at $t \in T$	Tons
h^L_{jt}	Lower violation of the amount extracted from short-term stockpile (and processed) at $j \in P$ at $t \in T$	Tons
w^U_i	Upper violation of the extraction targets for $i \in M$	Tons
w^L_i	Lower violation of the extraction targets for $i \in M$	Tons
$b^U_{jr\ell}$	Upper violation of the ore attribute percentage $r \in R$ at $j \in P$ during W_ℓ	Tons
$b^L_{jr\ell}$	Lower violation of the ore attribute percentage $r \in R$ at $j \in P$ during W_ℓ	Tons

Table 2.3: A summary of all decision variables used

Objective

A complicating factor of the dispatching problem is the competing objectives of meeting mining, processing rate, and quality targets during the planning horizon. The primary objective of the dispatching problem is to meet the processing rate targets. It is generally

expected that these targets are always met, but we start by considering this as an objective since it is possible in some rare cases they cannot be met. This occurs, for example, if many trucks are undergoing maintenance and there are insufficient trucks to supply the short-term stockpiles regularly. There are two additional competing goals for an open-pit mining operation: meeting mining targets and maintaining ore quality targets. One way to model these competing goals is to minimize a weighted sum of the amount each target is violated, varying the weights to emphasize different objective components. However, we found this model to be difficult to solve and it is challenging to control the trade-offs between the objectives. As an alternative, we use a multi-tiered hierarchy to model the competing objectives. In PHASE I, our objective is to meet processing rate targets. If we succeed, we add a constraint to require the processing rate targets to be met and try to meet mining extraction targets in PHASE II. Finally, in PHASE III we bound the deviation from mining targets to be not too far from the deviation in PHASE II, again constrain the processing rate targets to be met exactly, and try to meet the quality targets.

PHASE I

In PHASE I, we minimize the deviation from the processing rate target. Specifically, we solve the problem

$$\min_{v \in X} \sum_{j \in P} \sum_{t \in T} h^L_{jt}. \tag{P1}$$

Ideally, we achieve an objective value of 0 (no processing rate target violations). If not, we stop after this phase and report the best solution found. On the other hand, if we achieve 0 processing target violations, we proceed to PHASE II.

PHASE II

In PHASE II, we minimize the deviation from the mining extraction targets. If we reach PHASE II, we know there exists a feasible solution to P1 with $h^L = 0$, so we introduce

constraints fixing these variables to 0. We also need to track the progress in meeting mining targets.

We introduce variables w_i^U and w_i^L to measure the upper and lower violation of the mining extraction targets over the full planning horizon for each mine $i \in M$. We measure these violations as follows:

$$w_i^U \geq \sum_{t \in T, k \in K} \alpha_k y_{ikt}^M - \Theta_i^M \ \ \forall i \in M \tag{2.29}$$

$$w_i^L \geq \Theta_i^M - \sum_{t \in T, k \in K} \alpha_k y_{ikt}^M \ \ \forall i \in M \tag{2.30}$$

$$w_i^U, w_i^L \geq 0 \qquad \forall i \in M. \tag{2.31}$$

Constraints (2.29) - (2.31) measure the difference between the planned extraction amount ($\sum_{t \in T, k \in K} \alpha_k y_{ik}^M$) from a mine site $i \in M$ during the time horizon and the target extraction (Θ_i^M) by the end of the planning horizon. The difference is assigned to either the w^L or the w^U variables, depending on whether the extraction is below or above the target (respectively). We are concerned with the size of the greatest target violation rather than the sum total of all mines' target violations, so we introduce another variable z which is at least the maximum violation and minimize z in the objective function.

The full PHASE II model is then

$$\begin{aligned} &\min z && && \text{(P2)} \\ &\text{s.t. } (x,\ y^P,\ y^M,\ q^M,\ q^P,\ g,\ h^L,\ I) \in X \\ &\quad (y^M,\ w^U,\ w^L) && \text{satisfy } (2.29) - (2.31) \\ &\quad h_{jt}^L = 0 && \forall j \in P, t \in T \\ &\quad z \geq w_i^U,\ z \geq w_i^L,\ z \geq 0 && \forall i \in M. \end{aligned}$$

We record the actual mining target violation values for each site ($\hat{w}^U, \hat{w}^L$) after solving P2

and progress to PHASE III.

PHASE III

In PHASE III, we bound the mining target violation variables by the value $\hat{w}^U$ (or $\hat{w}^L$) plus a fraction of the target extraction for each site. The objective in PHASE III is to minimize the violation of the ore quality targets.

We choose β to be a small fraction, such as 1%, that allows some flexibility around the P2 solution values. The bounds on mining targets are:

$$w_i^U \leq \hat{w}_i^U + \beta\Theta_i^M \ \ \forall i \in M \tag{2.32}$$

$$w_i^L \leq \hat{w}_i^L + \beta\Theta_i^M \ \ \forall i \in M. \tag{2.33}$$

Now we measure the deviation from the quality targets. In general the goal is to monitor the quality of the ore in the short-term stockpile over a rolling horizon, but because the quality changes over time, the standard blending model in that approach results in nonconvex constraints. The actual ratio calculations in each period would be

$$\hat{C}_{jrt} = \hat{C}_{jrt-1} + \sum_{i \in G}(Q_{ir}^M \sum_{k \in K} \alpha_k y_{ijkt}^P) - \frac{\hat{C}_{jrt-1}}{I_{jt-1}}(\Theta_{jt}^P - h_{jt}^L), \tag{2.34}$$

where $\hat{C}_{jrt}$ is a new decision variable corresponding to the actual tons of attribute $r \in R$ unloaded at processing site $j \in P$ at time $t \in T$. Since $\hat{C}_{jrt}$, I_{jt}, and h_{jt}^L are all decision variables, the final term of this equation is the product of variables divided by another variable, making the equation nonconvex. Note that even this equation is an approximation, as it assumes perfect mixing of material once it enters the short-term stockpile.

To avoid introducing nonconvexity into our model, we use a family of overlapping constraints, each spanning a window of time of fixed length (a parameter L) and measure the violation of the ore quality *added* to the short-term stockpile within that window,

rather than the violation of ore quality *in* the short-term stockpile in a given period. Each subsequent constraint is written for a window of time that is D periods in the future. This structure is inspired by the L-average model proposed by Moreno et al. (2017). We set the window length L so that is possible to receive truckloads from mines with different qualities within the window, perhaps 10-20 minutes. This setting of L ensures there is a chance that the total ore added to the short-term stockpile in each window can meet the ore quality target. Let W be the largest integer such that $(W-1)D + L \leq T^{\max}$.

It is common for there to be an envelope of acceptable deviation from the quality targets that would be penalized very little or not at all. There are numerous options for modeling this piecewise-linear function. In this model, we introduce small bounds on the quality violation variables within which quality target violation is not penalized. We use a parameter ϵ to allow a window of quality target deviation before b^L or b^U become positive. In our test instances, we set ϵ to 0.005.

The constraints to track and measure ore quality deviation are as follows:

$$b^U_{jr\ell} \geq \sum_{\substack{\ell D+1 \leq t \leq \ell D+L \\ k \in K \\ i \in G}} \Big(Q^M_{ir} - Q^P_{jr} - \epsilon\Big)\alpha_k y^P_{ijkt} \qquad \forall j \in P, r \in R, \ell \in [1, W] \tag{2.35}$$

$$b^L_{jr\ell} \geq \sum_{\substack{\ell D+1 \leq t \leq \ell D+L \\ k \in K \\ i \in G}} \Big(Q^P_{jr} - \epsilon - Q^M_{ir}\Big)\alpha_k y^P_{ijkt} \qquad \forall j \in P, r \in R, \ell \in [1, W] \tag{2.36}$$

$$b^U_{jr,0} \geq \sum_{\substack{1 \leq t \leq L \\ k \in K \\ i \in G}} \Big(Q^M_{ir} - Q^P_{jr} - \epsilon\Big)\alpha_k y^P_{ijkt} + \Big(Q^I_{jr0} - Q^P_{jr} - \epsilon\Big) I_{j0} \qquad \forall j \in P, r \in R \tag{2.37}$$

$$b^L_{jr,0} \geq \sum_{\substack{1 \leq t \leq L \\ k \in K \\ i \in G}} \Big(Q^P_{jr} - \epsilon - Q^M_{ir}\Big)\alpha_k y^P_{ijkt} + \Big(Q^P_{jr} - \epsilon - Q^I_{jr0}\Big) I_{j0} \qquad \forall j \in P, r \in R \tag{2.38}$$

$$b^L_{jr\ell}, b^U_{jr\ell} \geq 0 \qquad \forall j \in P, r \in R, \ell \in [0, W]. \tag{2.39}$$

Constraints (2.35) - (2.39) measure the difference between the target ore quality and the

actual ore quality that is received by the short-term stockpile over W different time horizons. The constraints assign values to the variables b^L and b^U based on lower and upper violation of the quality targets, respectively. In constraints (2.35) and (2.36), the value is calculated as the absolute value of the difference between the quality of the $\sum_{\ell D+1\leq t\leq \ell D+L}\sum_{k\in K,i\in G}\alpha_k y^P_{ijkt}$ tons of ore added to short-term stockpile j during the window $[\ell D+1, \ell D+L]$ for $\ell \in [1,\cdots,W-1]$ and the target quality of that same volume of ore. Small modifications must be made for the first ($\ell = 0$) and last ($\ell = W$) constraints for each $j \in J$ and $r \in R$. When $\ell = W$, instead of covering L periods, the Wth constraint covers any remaining periods between $(W-1)D+L$ and $T^{\max}$. Finally, when $\ell = 0$ (constraints (2.37) and (2.38)), we add the absolute value of the difference between the target quality of the ore in the short-term stockpile ($Q^P_{jr}I_{j0}$) and the actual quality of the ore in the short-term stockpile at the beginning of the planning horizon ($Q^I_{jr0}I_{j0}$).

We again fix the processing rate violation variables to 0 and minimize the deviation from ore quality targets:

$$\begin{aligned}
&\min \sum_{j\in P}\sum_{r\in R}\sum_{\ell\in W}(b^L_{jr\ell}+b^U_{jr\ell}) && && \text{(P3)}\\
&\text{s.t. } (x,\ y^P,\ y^M,\ q^M,\ q^P,\ g,\ h^L,\ I)\in X\\
&\quad h^L_{jt}=0 && \forall j\in P, t\in T\\
&\quad (w^U,w^L) && \text{satisfy } (2.32)-(2.33)\\
&\quad (b^U,b^L,y^P) && \text{satisfy } (2.35)-(2.39).
\end{aligned}$$

2.3 Relaxation Induced Neighborhood Search Heuristic

For the three-phase hierarchical model to be used to make dispatching decisions in real time, we must be able to solve instances of P1, P2, and P3 quickly (ideally under a minute to solve all three). In particular, we have found P3 to be difficult to solve to optimality

because (a) for a planning horizon of only 20 minutes, the model can have over 10,000 integer variables and (b) the ore quality constraints have a complex structure.

We observe that P1 solves to optimality quickly enough in our test instances that we can solve it directly. P2 often solves quickly, but occasionally takes significant time to prove a solution is optimal. Thus, introducing a time limit (such as 20 seconds) on P2 is one way to ensure a bound on the mining target violations is obtained quickly. Similarly, a simple heuristic for solving P3 is to fix a time limit and record the best solution found. However, the solution quality could be bad, depending on the problem size and the choice of time limit. Thus, we implement a solution methodology that can find a good feasible solution to P3 quickly based on the solution to the LP relaxation of P3.

We propose a heuristic based on the relaxation induced neighborhood search (RINS) algorithm described by Danna et al. (2005). In the RINS algorithm, a neighborhood that is likely to contain the MIP solution is constructed based on a solution to the LP relaxation of the MIP model. Then a MIP model is solved to explore the neighborhood and find the best solution within that neighborhood. In our implementation, we create a neighborhood by fixing a subset of the P3 integer variables to 0 to reduce the size of the variable space, creating a problem that is more easily solved, and explore the neighborhood by solving the restricted P3 MIP.

Our implementation of the RINS heuristic is inspired by the fact that many of the integer variables take value 0 in both the optimal MIP solution and the solution to the LP relaxation of P3. Motivated by this observation, we first solve the LP relaxation of P3, then we fix all integer variables which are 0 in both the LP solution and the solution to P2. In this way, the solution to P2 remains feasible to P3. Finally, we re-introduce all the integrality constraints and solve the restricted P3.

2.4 Computational Experiments

We conduct a set of computational experiments to better understand the impact of our modeling choices and the effectiveness of the RINS heuristic. We begin by examining the computational difficulty of the dispatching model by solving instances with different planning horizon lengths. We then solve instances of P3 with several methods to determine whether the RINS heuristic can quickly get solutions that are comparable to optimal solutions. Next we compare solutions of P2 to those of P3 to see whether our approximate ore quality constraints improve the true ore quality. We also evaluate the behavior of the solution for different values of the parameters used in the ore quality constraints (2.35)-(2.39) and the bounds on mining target violation (2.32)-(2.33). Finally, we evaluate the difference between solutions of P3 when we include all the approximate loading and unloading capacity constraints (2.21) - (2.28) and when we include only the per-period capacity constraints (2.21) - (2.24) to determine whether the additional constraints reduce our overestimation of available capacity.

Comparison of Solution Methods

We conduct a computational experiment using data representative of an open-pit mine. The mine used in this experiment has $|G| = 5$, $|B| = 5$, $|P| = 2$, $|S| = 2$, $|D| = 5$, and $|R| = 1$. We test planning horizon lengths of 20 minutes, 30 minutes, and 40 minutes using 30-second time periods. Each of the 40 total trucks is randomly assigned one of two possible sizes. All other parameters used to build the model are randomly generated between some fixed bounds. We generate 10 separate instances of the random input parameters for each planning horizon length, for a total of 30 distinct instances. For the ore quality constraints (constraints (2.35)-(2.39)), we choose the time covered by each constraint (L) to be 20 periods (10 minutes) and the distance between successive constraints (D) to be 4 periods (2 minutes). The data used in this experiment can be found in Appendix A.1.

We solve the dispatching problem using different methods with the goal of obtaining an optimal (or close-to-optimal) solution in near-real time. The models are implemented in Python 2.7 and solved with Gurobi 7.0.2 using a 1% optimality gap tolerance. This study is conducted on an Intel Xeon X5650 server with 12 cores at 2.66Ghz and 128GB of RAM using 24 threads. Excluding the parameters specified hereafter, all Gurobi parameters are left in their default state.

Computational Difficulty for Different Planning Horizons

We first solve all three phases of the dispatching model for three different planning horizons to determine how computational difficulty scales with problem size. We allow P1 to solve to optimality and limit P2 to 20 seconds. If a solution is not found in 20 seconds, we allow Gurobi to search for a feasible solution for up to 90 seconds before moving to the final phase. If no solution can be found in 90 seconds, we use the solution to P1 as the best available solution for solving P3. We first solve P3 by enabling a Gurobi parameter that emphasizes finding feasible solutions and setting a time limit of one hour. The computational results for P3 can be found in Table 2.4. We observe the 10 instances' remaining optimality gap at the time limit, the time it took to reach that gap, and the total number of branch-and-bound tree nodes explored. The optimality gap is computed by the MIP solver as (best feasible solution – best lower bound)/(best feasible solution).

From this experiment, we see that directly solving P3 scales poorly with problem size. Only five instances out of the 30 total instances are within a 1% optimality gap after the one-hour time limit, and three of those instances are in the 20-minute horizon. Many of the optimality gaps reported are 100%; this is due to the fact that when the time limit is reached, the best lower bound on these instances is 0. Thus, on these instances it is difficult to determine from the optimality gap alone how close the solution was to optimal, since even a very small upper bound would yield a 100% optimality gap. We also note that on all 10 instances, regardless of planning horizon, the processing rate was met in P1, so we

% Gap			Time (s)			# Nodes		
20-min	30-min	40-min	20-min	30-min	40-min	20-min	30-min	40-min
93.37	98.96	25.94	TL	TL	TL	83355	62748	152296
32.51	100	65	TL	TL	TL	461109	4840	16300
99.94	100	100	TL	TL	TL	78774	19382	7042
0	1.54	1.37	20.31	TL	TL	0	17976	7134
100	100	100	TL	TL	TL	38753	3651	457
100	100	100	TL	TL	TL	199584	14483	3095
100	100	100	TL	TL	TL	70637	12349	25800
7.13	8.63	26.12	TL	TL	TL	314519	54285	39444
0	0.19	0.11	31.25	37.15	503.67	0	0	3
0.9	2.36	9.96	15.14	TL	TL	0	185382	3113

Table 2.4: Summary of the results of solving P3 as close to optimality as possible on three different planning horizons. The '% Gap' column gives the remaining optimality gap at termination. The '# Nodes' column gives the total number of branch-and-bound tree nodes explored while solving the P3 problem. 'TL' indicates that the the time limit was reached before an optimal solution was found.

were able to continue to solve P2 and P3 with the processing rate violations fixed to 0.

Solution Quality with Competing Heuristics

In this section we compare the performance of two competing heuristics. Once again, we allow P1 to solve to optimality and enforce a 20-second time limit on P2. If a solution is not found in the 20-second time limit, we allow Gurobi to continue searching for a feasible solution for up to 90 seconds. If t' is the number of seconds required to solve P2, then we set a time limit of $90 - t'$ seconds for solving P3. Thus, a total time limit of 90 seconds is given to solving both P2 and P3. We apply the following heuristics to solve P3:

1. Enforce a time limit on P3 corresponding to the amount of the 90 second time limit remaining after solving P2 and record the best solution found in that time and

2. Use RINS to solve P3 with a time limit corresponding to the amount of the 90 second time limit remaining after solving P2 and record the best solution found in that time.

For the 10 sample instances selected, we solve each model for 40 periods, 60 periods, and 80 periods of length 30 seconds. Note that the RINS heuristic could also be applied to solving P2, although we do not explore that in this work.

We do not consider the computational difficulty of either of the heuristics, as they solve quickly by design and only obtain a feasible solution to P3, though it is worth noting the RINS-based heuristic occasionally terminates much more quickly than the time limit. For each planning horizon length, we observe the total quality target violation in the solution to P2, in the solution to P3 when solved as close to optimality as possible (OPT), in the solution to P3 using a time limit of 90 seconds, and in the solution to P3 using the RINS heuristic. The results of this experiment can be found in Table 2.5.

Total violation of quality targets (P3 objective)											
20-min horizon				30-min horizon				40-min horizon			
P2	OPT	90-SEC	RINS	P2	OPT	90-SEC	RINS	P2	OPT	90-SEC	RINS
265	0.7	1.1	7.6	559	5	55.5	14.7	754	25.9	-	17.4
429	5.4	5.4	19.5	968	5	7	2.6	2065	37	49.7	5.3
240	0.6	0.8	100	359.5	1	-	1.6	242.2	1.7	168.8	4.3
302	**155.1**	155.1	168.7	365	144.6	148	154.6	554	143	155.2	147.6
235	0.2	0.2	2.9	157	0	0	7.6	771	9.5	-	4.7
302	0.5	0.8	1.3	763	0.5	4.3	3	1164	1.5	2241.8	9
107	1.7	1.7	3.1	278.4	1.8	12.4	3.9	632	3.6	-	6.4
659	31.1	31.1	40.6	1525	31.3	35	47	2848	37.3	57.4	44.6
135	**42.7**	42.7	42.7	249.1	**28.3**	28.3	28.3	565	**28**	-	28.3
325	**86.4**	86.4	86.5	1684	80.3	-	72	2406.6	86.1	-	86.9

Table 2.5: Summary of the total quality target violation using different solution methods as compared to not solving P3. Each method is tested on 3 different planning horizons. Numbers in **bold** font indicate which instances were solved to optimality in an hour.

We observe that the ore quality objective value decreases significantly when solving P3 rather than stopping after solving P2. More notably, the ore quality objective is comparable for the 90-second time limit (90-SEC) heuristic and the RINS heuristic and, when a solution is found, both heuristics improve on the P2 value. However, when using the 90-SEC heuristic, no feasible solution was found in two of the 30-minute horizon instances and five of the 40-minute horizon instances. We were able to find a solution within 90 seconds for all instances using the RINS heuristic.

We also compare the true ore quality across different solution methods. The quality violation measurements in P3 are only an approximation of the true ore quality because of the nonconvex nature of the blending model. Because this approximation does not

accurately calculate the true quality violations, after solving P3 we use Equation (2.34) to calculate the true ore quality in the given solution. We demonstrate the solution quality of each instance by reporting the following values in Tables 2.6, 2.7, and 2.8 for planning horizons of 20 minutes, 30 minutes, and 40 minutes, respectively:

- **% of planning horizon in which quality is within target bounds (% of PH qual. within target):** Using the nonconvex blending model, we calculate the approximate true ore quality after solving P3 using each of the solution methods. Then we calculate the percent of periods in the planning horizon the true quality is within the ϵ window of allowable violation at each processing site.

- **% of planning horizon in which quality is within 2x target bounds (% of PH qual. within 2x target):** Using the nonconvex blending model, we calculate the approximate true ore quality after solving P3 using each of the solution methods. Then we calculate the percent of the periods in the planning horizon the true quality is within 2ϵ of the target values at each processing site.

20-minute horizon							
% of PH qual. within target				% of PH qual. within 2x target			
P2	OPT	90-SEC	RINS	P2	OPT	90-SEC	RINS
13.8	2.5	13.8	0	70	97.5	96.2	95
100	100	100	100	100	100	100	100
97.5	87.5	97.5	100	100	100	100	100
82.5	**88.8**	81.2	90	100	**100**	100	100
73.8	63.7	85	60	73.8	76.2	85	85
33.8	62.5	53.8	56.2	58.8	77.5	90	100
62.5	68.8	70	70	75	93.8	88.8	96.2
0	37.5	38.8	56.2	57.5	100	95	100
100	**100**	100	100	100	**100**	100	100
62.5	**78.8**	90	83.8	100	**100**	97.5	100

Table 2.6: The true post-solution ore quality evaluated after solving P2 as compared to applying different methods to P3. The percent of the 20-minute planning horizon the quality is within the allowable window and a window two times larger is compared between all solution methods. The OPT values in **bold** indicate that those instances solved to optimality in under one hour.

30-minute horizon							
% of PH qual. within target				% of PH qual. within 2x target			
P2	OPT	90-SEC	RINS	P2	OPT	90-SEC	RINS
0	5	3.8	0	96.2	100	100	100
100	98.8	100	98.8	100	100	100	100
100	95	-	93.8	100	100	-	100
86.2	77.5	57.5	90	100	100	100	100
71.2	86.2	75	87.5	96.2	97.5	93.8	88.8
27.5	70	55	68.8	65	93.8	70	92.5
63.7	75	73.8	85	98.8	100	88.8	97.5
0	62.5	40	62.5	46.2	100	100	100
100	**100**	100	100	100	**100**	100	100
13.8	83.8	-	92.5	78.8	100	-	100

Table 2.7: The true post-solution ore quality evaluated after solving P2 as compared to applying different methods to P3. The percent of the 30-minute planning horizon the quality is within the allowable window and a window two times larger is compared between all solution methods. The OPT values in **bold** indicate that those instances solved to optimality in under one hour.

40-minute horizon							
% of PH qual. within target				% of PH qual. within 2x target			
P2	OPT	90-SEC	RINS	P2	OPT	90-SEC	RINS
0	6.2	-	0	96.2	100	-	100
100	100	96.2	100	100	100	100	100
98.8	100	98.8	97.5	98.8	100	98.8	100
51.2	72.5	70	73.8	100	100	100	100
86.2	82.5	-	68.8	86.2	87.5	-	82.5
55	62.5	56.2	55	72.5	91.2	76.2	93.8
77.5	73.8	-	88.8	100	92.5	-	100
0	60	51.2	58.8	30	100	100	100
100	**100**	-	100	100	**100**	-	100
12.5	82.5	-	90	100	100	-	100

Table 2.8: The true post-solution ore quality evaluated after solving P2 as compared to applying different methods to P3. The percent of the 40-minute planning horizon the quality is within the allowable window and a window two times larger is compared between all solution methods. The OPT values in **bold** indicate that those instances solved to optimality in under one hour.

In 23 of the 30 total instances, the actual ore quality values improve (or stay approximately the same) when solving P3 instead of only P2. When stopping after solving P2, 10 of the 30 instances have ore quality in acceptable bounds less than 50% of the horizon, whereas only four instances do so in OPT. We also observe the solutions found by RINS are

comparable to the OPT solutions. When the target window is expanded, however, the RINS solutions actually tend to improve on the OPT solutions, since the OPT method is stopped well before reaching optimality. For example, in all but two instances in the 20-minute horizon, the ore quality values are in the 2x (1%) window more often than (or for the same amount of time as) the OPT values. The values found by the 90-second heuristic are similar to those found by the RINS heuristic, but the 90-second heuristic was unable to find a feasible solution in many of the instances in the 30- and 40-minute planning horizons. From these results, we see that RINS and OPT are comparable in terms of number of instances within the allowable quality windows. The 90-SEC heuristic tends to produce worse results than OPT, and is occasionally worse than just stopping after solving P2, particularly for the 40-minute planning horizon.

The results presented in this section allow us to draw some conclusions about our model. As expected, the P3 solution quality deteriorates somewhat when solved with the RINS heuristic, but overall the solutions are comparable to solving the instance as close to optimality as possible with a one-hour time limit. More significantly, these solutions are obtained in 90 seconds or less, making this model efficiently implementable. For horizons of 30 or 40 minutes, the solution obtained from RINS within 90 seconds is better than the solution obtained by simply enforcing a 90-second on the emphasize feasibility method, since for many instances a solution could not even be found in that time. Furthermore, we conclude that the inclusion of P3 is important for meeting the ore quality objective, in spite of the additional computational difficulty over P1 and P2.

Changing Ore Quality Constraint Parameters

In this section we compare the solution quality for differing values of the parameters L and D in the target ore quality violation constraints (constraints (2.35)-(2.39). L gives the number of periods covered by each constraint and D gives the number of periods between each successive constraint. Thus, small values of L and D result in many constraints covering

short windows of time, whereas a large value of L could result in as few as one constraint. We aim to understand the effect of our choice of L and D on the solution quality. We vary L between 10, 20, and 30 periods and D between 1, 2, 4, 10, and 20 periods. We solve P3 using the RINS heuristic. For this experiment, we run the same 10 instances from the previous experiments on a 20-minute planning horizon. We average the results of the ten instances and record the true ore quality target violations after solving P3 (Quality Obj), the percent of the planning horizon the true ore quality is within a 0.5% window of the quality targets in P3 (Qual in target), the percent of the planning horizon the true quality is within a 1% window of the targets in P3 (Qual in 2x target), and the average time it takes to solve P3. The results of this experiment are reported in Table 2.9.

L	D	Quality Obj	Qual in target	Qual in 2x target	Avg time (s)
10	1	4.11	75.76	97.51	64.10
10	2	3.99	77.5	97.38	43.74
10	4	6.93	72.86	98.51	38.28
10	10	10.39	74.13	97.38	22.55
10	20	18.29	70.62	94.13	33.53
20	1	3.77	71.75	97.88	38.22
20	2	5.46	75.12	97.51	23.53
20	4	7.60	71.62	95.25	42.79
20	10	12.65	60.37	93.63	36.84
20	20	13.89	67.13	91.73	1.72
30	1	7.23	70.87	90.63	30.26
30	2	7.61	71.88	95.00	31.81
30	4	13.91	73.13	94.75	24.14
30	10	13.14	61.76	88.25	1.50
30	20	26.07	62.51	88.37	1.56

Table 2.9: Comparison of solution quality for different values of L and D.

From this experiment, we see that the solution quality generally decreases as L and D increase, but for $L = 10$ and $L = 20$, the solution is not very sensitive to the choice of D. We further note that the problem is harder to solve as L and D decrease, but appears more sensitive to the choice of D than the choice of L. From these results, we conclude that for any choice of L, we should choose a value of D that is not too large. Thus, we conclude that a choice of $L = 20$ and $D = 4$ is a reasonable choice for these parameters.

β	P2 Mine	P3 Mine	Qual in target	Qual in 2x target	P3 Qual
0%	110.62	111.09	73.76	94.25	44.09
1%	110.62	129.1	72.50	97.51	40.87
2.5%	111.34	151.63	71.11	97.12	43.14
5%	110.62	201.8	73.75	95.38	37.97

Table 2.10: Comparison of solution quality for different values of β.

Changing Mining Extraction Bounds in P3

We next compare the solution quality for different values of the parameter β in constraints (2.32) and (2.33), which determines how much we are allowed to deviate from the P2 mining target violation solution values. We aim to understand the interaction between the ore quality and the mining target violation when solving P3. We vary β between 0%, 1%, 2.5%, and 5% and solve P3 using the RINS heuristic. For this experiment, we run the same 10 instances from the previous experiments on a 20-minute planning horizon. The extraction target for each mining site in each instance is on average about 1500 tons. We average the results of the ten instances and record the mining target violations after solving P2 (P2 Mine), the mining target violations after solving P3 (P3 Mine), the percent of the planning horizon the true ore quality is within a 0.5% window of the quality targets in P3, the percent of the planning horizon the true quality is within a 1% window of the targets in P3, and the ore quality objective in P3 (P3 Qual). The results of this experiment are reported in Table 2.10.

From this experiment, we see that the mining target violation increases with increasing values of β. As we expect, the ore quality objective in P3 decreases as we increase β, but the actual quality values do not change much as β increases. In fact, from these results, the true quality appears to stay about the same on average as β increases, so using a small value of β (such as 1%) allows us to find a solution that yields both good mining violation and good ore quality violation values.

Linear Capacity Constraints

Finally, in this section, we report results on an experiment designed to study the effects of our linear approximation of the effects of limited loading and unloading capacity (constraints (2.17)-(2.28)). We conduct a computational experiment using data representative of an open-pit mine in order to determine if these constraints are an effective way to limit how much the true capacity is overestimated in each period. The data used in this experiment is nearly identical to that used in the experiments in the previous section and can be found in Appendix A.1.

We solve the dispatching model using only constraints (2.17) - (2.20), which impose per-period capacity restrictions, and again with constraints (2.17) - (2.28). We solve P1 to optimality and enforce a 20-second time limit on P2. If a solution is not found in the 20-second time limit, we allow Gurobi to continue searching for a feasible solution for up to 90 seconds. As before, if t' is the number of seconds required to solve P2, we set a time limit of $90 - t'$ seconds for solving P3 with the RINS heuristic.

Using 10 instances with a 20-minute planning horizon, we demonstrate the solution quality of each instance by choosing a site in the mine (in this study, we choose one of the processing sites) and reporting the following values:

- **Avg Over PP**: the percent of periods in which the given site exceeded the true fractional truck capacity using only the per-period capacity constraints
- **Avg Over PP+**: the percent of periods in which the given site exceeded the true fractional truck capacity using the per-period capacity constraints as well as the moving window constraints with optimal window length d
- **Max Cons. Over PP**: the greatest number of consecutive periods in which the given site exceeded the true fractional truck capacity using only the per-period capacity constraints

- **Max Cons. Over PP+**: the greatest number of consecutive periods in which the given site exceeded the true fractional truck capacity using the per-period capacity constraints as well as the moving window constraints with optimal window length d

By reporting the percent of time the capacity is overestimated, we gain an understanding of how often our model allows too many trucks to be loaded or unloaded in a period. However, this alone does not demonstrate how much we are overestimating capacity; more importantly, capacity should not be exceeded for too many periods in a row. If capacity is exceeded at a site, the solution gives a poor approximation of when trucks will become available. When capacity is exceeded for many periods in a row, the solution indicates that trucks will become available several periods before they actually do. However, if the solution has short sequences (1-2 periods) of exceeded capacity at each site, the trucks become available only 1-2 periods after the solution says they will. Thus, the goal of the additional constraints (PP+) is to prevent a long series of periods in which capacity is exceeded. The results of this study are in Table 2.11.

Avg % Over PP	Avg % Over PP+	Max Cons. Over PP	Max Cons. Over PP+
22.5	25.0	2	1
22.5	22.5	3	2
27.5	27.5	4	2
22.5	22.5	3	3
22.5	22.5	4	1
25.0	22.5	4	2
27.5	27.5	3	1
27.5	27.5	3	1
30.0	25.0	4	2
25.0	25.0	4	2

Table 2.11: Summary of the quality of our solution methods on a 20-minute planning horizon. This table compares the solution quality with a basic constraint implementation (PP) and our error-reducing constraint implementation (PP+).

From the results, we see the percent of periods where the actual trucks unloaded is above the maximum (fractional) capacity is approximately the same between PP and PP+. However, using all the constraints (PP+) limits overestimation of the number of trucks for

multiple periods in a row. In fact, the maximum number of consecutive periods over the capacity in the PP+ is three periods (although this only happens once – it is one or two periods for nine out of ten instances), as opposed to four (10% of the planning horizon) when using only the per-period constraints (PP). We claim that choosing a throughput window length wisely can significantly reduce the adverse effects of rounding to integer trucks.

2.5 Conclusion

We propose the use of a new time-discretized MIP model to solve the open-pit mine truck dispatching problem. This model can be solved using a three-phase hierarchy to iteratively optimize and constrain the competing objectives of meeting processing rate targets, mining extraction targets, and ore quality targets. Through the use of the RINS heuristic, instances of the model can be solved to good feasible solutions within a 90-second time limit, much better than the solutions obtained in that same time by simply imposing a time limit on the original problem. This is particularly notable in instances with planning horizons of thirty minutes or more. Furthermore, through the use of a new approach to modeling ore quality violation and truck capacity at each site, we are able to incorporate complicating aspects of the mining system in a tractable way. The capacity constraints approximate the true (fractional) capacity so that solutions do not exceed the capacity of any site for many consecutive periods. The ore quality constraints effectively approximate the nonlinear blending effects. Most importantly, given a solution to the dispatching problem, we can build a dispatching policy. For each location, the solution gives a list of the next locations to which trucks can be sent; every time a truck becomes available, we dispatch the truck to the next location on the list.

In Chapter 3, we discuss incorporating the dispatching problem into an open-pit mine simulation. The simulation integration allows this model to be compared to other policies

similar to what is found in current literature and provides additional data about the practical usefulness of our model. It would also be valuable to explore the creation of a hybrid model, combining both the short-term dispatching decisions and the longer-term daily target setting decisions. This model would allow decision makers to look further into the future to see how immediate decisions will affect longer-term targets, and alter target values accordingly.

3 DISPATCHING POLICIES IN OPEN-PIT MINING

3.1 Introduction

We present three new policies to make dispatching decisions in open-pit mining. Through the use of a discrete-event simulation, we demonstrate how the three policies perform on open-pit mines with different characteristics. Our primary contribution is a demonstration that the MIP-based dispatching model of Chapter 2 can yield implementable solutions when solved with existing methods that perform well in a discrete-event simulation of open-pit mines with different characteristics.

Literature Survey

A number of different rules have been used for truck dispatching. As described in Chapter 2, current literature on the truck dispatching problem discusses the use of a two-phase method. In the first phase, a linear programming model is solved to set target flow rates between each pair of locations. The second phase typically involves a small matching or assignment-type problem that matches the current decision to the target flow rates (Elbrond and Soumis (1987a); Temeng et al. (1997); White (1991); Subtil et al. (2011); Ahangaran et al. (2012)).

These rules are often embedded in a simulation in order to demonstrate how the current decision will affect the future state of the system. Subtil et al. (2011) find the optimal number of trucks within the mine and then solve a simulation-based heuristic that combines simulation and a multi-criterion LP model. One heuristic, the 1-truck-for-N-shovels strategy, compares the benefit and cost of sending an available truck to any of the N mining sites but ignores the potential effects of the given truck's interactions with other trucks, such as queueing (Chatterjee and Brake (1981); Tu and Hucka (1985); Lizotte and Bonates (1987); Sadler (1988); Li (1990); Bonates (1992); Forsman et al. (1993); Panagiotou and Michalakopoulos (1995); Ataeepour and Baafi (1999)). A related strategy known as the

M-trucks-for-1-shovel strategy ranks mine sites by how much additional material needs to be removed to meet the target and finds the next M trucks to be dispatched. This strategy ignores many of the interaction effects between system components over time (White et al. (1982); Arnold and White (1983)). Finally, it is possible to combine these two strategies into an M-trucks-for-N-shovels strategy to better account for the interconnected nature of the mining system. Hauck (1973, 1979), and Munirathinam and Yingling (1994) suggest single-stage systems to determine a sequence of dispatching decisions by solving a series of assignment problems. Elbrond and Soumis (1987b) and Temeng et al. (1997) consider a multi-stage system but do not allow assigning multiple trucks to a single location and cost calculations do not account for future queueing effects if multiple trucks are routed to the same location.

Contributions

To make real-time dispatching decision in an open-pit mine, we propose and evaluate three dispatching policies. Our main contributions include a detailed nonlinear model for setting target flow rates as well as two dispatching policies that map current decisions to target flow rates. We also use the discrete-time MIP dispatching model described in Chapter 2 to define a MIP-based dispatching policy. We conclude with a demonstration of the performance of each policy in a discrete-event simulation of an open-pit mine.

One limitation of existing two-phase dispatching methods is that they do not account for queueing from multiple trucks being sent to the same location. Our nonlinear flow-rate model, which sets target truck flow rates using data about production and quality targets, includes constraints to capture the effects of queueing at different sites. The flow-rate model uses an M/G/1 queueing formula and treats the arrival rate as a decision variable and is convex, as the nonlinear queueing formula can be reformulated as a rotated second-order cone constraint.

To make dispatching decisions, we propose a simple greedy target-matching dispatching

policy that matches available trucks to the sequence of two routes farthest from meeting their target flow rates. The greedy policy, like the heuristics described in literature, is myopic, only considering the next decision to be made without regard to how it will affect the overall system. Our second proposed target-matching policy is less myopic and involves solving a small MIP that computes the maximum possible progress toward meeting the target flow rates given the trucks becoming available in the next $\tilde{T}$ units of time and the list of possible round trips for each of those trucks. Finally, we compare the effectiveness of these two heuristic policies and the MIP-based policy by incorporating them into a discrete-event simulation of an open pit mine.

In Section 3.2, we describe the stochastic model that defines our discrete-event simulation of an open-pit mine. We present our three dispatching policies in Section 3.3. In Section 3.4, we give results for a comprehensive computational study in which we evaluate the three dispatching policies on mines with a variety of features, such as varying numbers of trucks, ore quality, travel times, and mine geography. We also examine how the modeling choices we make in both the discrete-time MIP and nonlinear flow-rate models influence the open pit mine simulation results.

3.2 Stochastic Model of Open-Pit Mine

We begin with a full description of the stochastic model of an open-pit mining system. The mine consists of a set of locations, N. The set N is partitioned into mining sites M, processing sites P, dump sites D, and long-term stockpile sites S. The set M is partitioned into mines where ore is extracted (ore mines) G and mines where overburden is extracted (waste mines) B. Material taken from ore mines is of high enough quality that it can be processed or taken to a long-term stockpile. In our model, we treat long-term stockpiles as storage sites for material from ore mines, so trucks at ore mine sites can be dispatched to either a processing site or a long-term stockpile. Instead of using them as storage sites, long-

term stockpiles could alternatively be used as sites from which material can be removed for processing. Our model could be extended to include this option, but we do not consider it in this work. Material taken from waste mines must be taken to a dump site since it must be removed before ore can be extracted. There is a set of attributes, R, that are tracked to measure ore quality. There is also a set of truck sizes K, and a given number of trucks of each type. The initial location of all trucks is assumed to be given.

Trucks are loaded at mining sites $i \in M$, and the time to load a truck is a random variable $\boldsymbol{p}_i$. Loading at each mine is modeled as a single-server system, and trucks waiting to be loaded queue in a first-in first-out (FIFO) basis. The amount of ore extracted from each mine $i \in M$ in each truckload is represented by a random variable $\boldsymbol{\alpha_k}$ tons of ore for $k \in K$. Similarly, the time to unload a truck at a processing site, long-term stockpile, or dump site i is is a random variable $\boldsymbol{p}_i$, and each of these sites is also treated as a single-server system with a FIFO queue. When a truck finishes loading or unloading at a site, its next destination is determined by calling a dispatching function, which takes as an input the state of the system and returns the next destination of the truck. The time it takes a truck to travel between each pair of locations is given by a random variable $\boldsymbol{\tau}_{ij}$ for $i \in N$, $j \in N$.

Each processing site $j \in P$ maintains a short-term stockpile where trucks dump material taken from the mines. Ore is removed from the stockpile for processing. We assume that the material added to the stockpile is instantaneously perfectly blended with any material remaining in the stockpile at the time the unloading is finished. Each processing site $j \in P$ has a processing rate Θ_j^P. We assume this to be a constant rate of depletion, in tons/minute, unless the total tons of ore in the short-term stockpile plus any truckloads currently en route to the processor falls below ℓ tons. If this threshold is reached, the processing rate slows to $0.6\Theta_j^P$. If the short-term stockpile is emptied, the processor must stop. Processors slowing and processors stopping are both considered system failures and are therefore undesirable. Each short-term stockpile j has an initial amount of ore I_{j0} and initial quality, Q_{jr0}^I for all $r \in R$. Each short-term stockpile j also has a maximum capacity of C_j tons for

$j \in P$. If a truck's load would exceed the capacity, it must wait at that site until there is room in the short-term stockpile for the load. If the simulation is to be run for t minutes, each mining site i has an extraction target during that time of Θ_i^M, which is calculated as $\Theta_i^M = \sum_{j \in P} \frac{t\Theta_j^P}{|G|}$.

Finally, we are given ore quality targets (in percentage of attribute $r \in R$) Q_{jr}^P at all $j \in P$ and for all $r \in R$. Each truckload removed from an ore mine $i \in G$ has associated percentages of ore attributes represented by a random variable, $\boldsymbol{Q_{ir}^M}$ for all $r \in R$. All random variables described in this section are assumed to be independent of each other.

The simulation length T represents a portion of a shift, possibly including some time for a shift change. Some other important components of the open-pit mine simulation are the ramp-up and ramp-down effects at the beginning and end of each shift, respectively. Although in this work we investigate an open-pit mine operating in a steady state, the stochastic model can easily be extended to account for ramp-down effects toward the end of a shift by (a) requiring trucks to be dispatched to the nearest bus stop if they would not have enough time to travel 1.5 times the distance to the farthest destination in the mine by the end of the shift and (b) requiring the short-term stockpiles to be almost full near the end of the shift so production does not shut down during the shift change. Note that the first requirement is a very conservative estimate of travel time to guarantee near certainty that all drivers are at a bus stop by the end of their shift. At the beginning of a shift, all trucks are at bus stops and must be dispatched to new destinations.

3.3 Dispatching Policies

The key choices made in the open-pit mine simulation are the truck dispatching decisions. As stated in the stochastic model in Section 3.2, each time a truck finishes being loaded or unloaded, the simulation needs to call a dispatching function to acquire the truck's next destination. In this section, we describe three different dispatching policies that can be used

to determine the next destination for any truck in the open-pit mine. The first two policies are both examples of the two-phase policy procedure described in current literature in which an average rate model is solved, yielding target flow rates, then a heuristic is used to match the actual dispatching decisions to the target flow rates. The third policy is based on the MIP dispatching model described in Chpater 2. We propose a new average flow rate model and two possible second-phase heuristics in addition to the MIP-based dispatching policy.

Two-Phase Dispatching Policy

We first propose two dispatching policies based on the two-phase flow-rate matching method from current literature. The first phase involves solving an aggregate rate-setting problem to obtain target flow rates over a specified horizon, such as one hour. The second phase is performed every time a dispatching decision must be made, and typically involves an assignment or matching-type problem to determine how best to make progress in meeting the target rates.

We propose a version of this two-phase approach involving a nonlinear (second-order cone constrained) first-phase problem that incorporates queueing effects. For the second phase, there are two competing options that we implement: a greedy target-matching heuristic and a MIP-based target-matching heuristic. Both policies match available trucks to possible round trips. In our implementation, we define a *round trip* as a sequence of two origin-destination pairs, though the starting and ending locations could be different.

Nonlinear Average Flow Rate Model

In this section, we describe an average flow-rate model that is solved at regular intervals or any time a significant disruption occurs in the system (such as an equipment failure). Given target processing and mining rates and target ore quality, solving the model yields the average flow rates of trucks to each location that meet the targets as closely as possible,

subject to physical constraints. Decision variables in the flow-rate model determine the violation of loading volume, unloading volume, and ore quality targets, as well as the rate of trucks traveling between pairs of locations in the mine. Constraints in this model measure violation of the target parameters, balance flow into and out of locations in the mine, bound the total trucks in use at any given time, and track the average queue lengths using a nonlinear M/G/1 queueing formula. The objective is to meet the processing rate, mining rate, and ore quality targets, with the greatest emphasis given to meeting the processing rate targets.

Parameters

The input data for the flow rate model describe how the system should be operating if it is in a steady state. There are a total of Γ_k trucks of each possible size $k \in K$. Trucks travel between pairs of locations, and each route has a parameter representing travel time, t_{ij} for $i \in N, j \in N$. Each site $i \in N$ has an upper ($\overline{p}_i$) and lower ($\underline{p}_i$) bound on the tons of ore that can be either extracted (at a mine) or processed (at a processing, dump, or stockpile site) each hour. If there is a single target rate at a site, the upper and lower bounds could be equal. There are upper ($\overline{g}_{jr}$) and lower ($\underline{g}_{jr}$) bounds on attribute percentages at all $j \in P$ and for all $r \in R$. Each ore mine $i \in G$ has parameters ρ_{ir} representing the average attribute percentage for each $r \in R$. Each site $i \in N$ can serve trucks with a mean service time of $1/\mu_i$ hours/truck and a standard deviation of σ_i hours/truck. We assume a general distribution for the service times. These data are summarized in Table 3.1.

Decision Variables and Constraints

For each site $i \in N$, variables v_i^ℓ measure the rate in tons per hour by which the lower bound $\underline{p}_i$ is violated. Similarly, v_i^u measures the tons/hour the upper bound $\overline{p}_i$ is violated. The variables x_{ijk} represent the trucks/hour of type $k \in K$ traveling from $i \in N$ to $j \in N$, so the value $\sum_{k \in K} \alpha_k x_{ijk}$ gives the total tons traveling from i to j in each hour. The full set

Parameter	Meaning	Units
α_k	Size of truck type $k \in K$	Tons/truck
Γ_k	# of trucks of type k	Trucks
t_{ij}	Time required to travel from $i \in N$ to $j \in N$	Hours
$\overline{p}_i$	Maximum production rate at site $i \in N$	Tons/hour
$\underline{p}_i$	Minimum production rate at site $i \in N$	Tons/hour
$\overline{g}_{jr}$	Upper bound on grade-element ratio of attribute $r \in R$ at production site $j \in P$	(%)
$\underline{g}_{jr}$	Lower bound on grade-element ratio of attribute $r \in R$ at production site $j \in P$	(%)
ρ_{ir}	grade-element ratio of attribute $r \in R$ at location $i \in G$	(%)
$1/\mu_i$	Mean service time at $i \in M \cup P$	Hours/truck
σ_i	Standard deviation of service time at site $i \in M \cup P$	Hours/truck

Table 3.1: A summary of all parameters and data used in the flow rate model.

of constraints measuring rate bound violations is

$$v_i^\ell \geq \underline{p}_i - \sum_{j \in P \cup S} \sum_{k \in K} \alpha_k x_{ijk} \qquad \forall i \in G \tag{3.1}$$

$$v_i^u \geq \sum_{j \in P \cup S} \sum_{k \in K} \alpha_k x_{ijk} - \overline{p}_i \qquad \forall i \in G \tag{3.2}$$

$$v_i^\ell \geq \underline{p}_i - \sum_{j \in D} \sum_{k \in K} \alpha_k x_{ijk} \qquad \forall i \in B \tag{3.3}$$

$$v_i^u \geq \sum_{j \in D} \sum_{k \in K} \alpha_k x_{ijk} - \overline{p}_i \qquad \forall i \in B \tag{3.4}$$

$$v_j^\ell \geq \underline{p}_j - \sum_{i \in M} \sum_{k \in K} \alpha_k x_{jik} \qquad \forall j \in P \cup S \cup D \tag{3.5}$$

$$v_j^u \geq \sum_{i \in M} \sum_{k \in K} \alpha_k x_{jik} - \overline{p}_j \qquad \forall j \in P \cup S \cup D. \tag{3.6}$$

We could incorporate the end-of-shift requirement that the short-term stockpiles at the processors are nearly full by modifying constraints (3.5) and (3.6) as follows:

$$v_j^\ell \geq \underline{p}_j - \sum_{i \in M} \sum_{k \in K} \alpha_k x_{jik} + \frac{C_j - I_{j0}}{T'} \qquad \forall j \in P$$

$$v_j^u \geq \sum_{i \in M} \sum_{k \in K} \alpha_k x_{jik} - \overline{p}_j - \frac{C_j - I_{j0}}{T'} \qquad \forall j \in P,$$

where T' is the time (in hours) remaining before the end of the shift and $C_j - I_{j0}$ is the amount of ore needed to fill the short-term stockpile at processor j. This would increase both the minimum and maximum processing rate by $\frac{C_j - I_{j0}}{T'}$ tons/hour.

Similarly, we measure the amount by which the lower and upper bounds on each attribute $r \in R$ are not met at each processing site $j \in P$. To do so, we introduce variables f_{jr}^{ℓ} and f_{jr}^{u} to represent the violation of the lower and upper bounds, respectively. The bound violation is calculated as the non-negative difference between the lower or upper bound on ore quality and the actual quality of the ore brought from all ore mining sites G:

$$f_{jr}^{\ell} \geq -\left(\sum_{i \in G}\sum_{k \in K} \alpha_k(\rho_{ir} - \underline{g}_{jr})x_{ijk}\right) \qquad \forall j \in J, \forall r \in R \tag{3.7}$$

$$f_{jr}^{u} \geq \sum_{i \in G}\sum_{k \in K} \alpha_k(\rho_{ir} - \overline{g}_{jr})x_{ijk} \qquad \forall j. \in J, \forall r \in R. \tag{3.8}$$

We balance the flow rate of trucks into each location $i \in N$ with the flow rate of trucks out of that location:

$$\sum_{j \in N \setminus M} x_{jik} = \sum_{j \in P \cup S} x_{ijk} \qquad \forall i \in G, \forall k \in K \tag{3.9}$$

$$\sum_{j \in N \setminus M} x_{jik} = \sum_{j \in D} x_{ijk} \qquad \forall i \in B, \forall k \in K \tag{3.10}$$

$$\sum_{j \in B} x_{jik} = \sum_{j \in M} x_{ijk} \qquad \forall i \in D, \forall k \in K \tag{3.11}$$

$$\sum_{j \in G} x_{jik} = \sum_{j \in M} x_{ijk} \qquad \forall i \in P, \forall i \in S, \forall k \in K. \tag{3.12}$$

To incorporate the effects of queueing at different sites, we introduce decision variables λ_i that represent the arrival rates at each location $i \in N$, written as a function of truck flow

rates:

$$\lambda_i = \sum_{j \in P \cup D \cup S} \sum_{k \in K} x_{jik} \qquad \forall i \in M \tag{3.13}$$

$$\lambda_j = \sum_{i \in G} \sum_{k \in K} x_{ijk} \qquad \forall j \in P \cup S \tag{3.14}$$

$$\lambda_j = \sum_{i \in B} \sum_{k \in K} x_{ijk} \qquad \forall j \in D. \tag{3.15}$$

Specifically, the arrival rate at $i \in M$ is the total flow in trucks/hour coming from a processing site, dump site, or long-term stockpile. The arrival rate at $j \in P \cup S$ is the total flow in trucks/hour coming from an ore mine and the arrival rate at $j \in D$ is the total flow in trucks/hour coming from a waste mine.

We choose to model queueing in an open-pit mine as an M/G/1 queue. The formula for the average number of customers in the system for an M/G/1 queue is

$$L_i \geq \frac{\lambda_i}{\mu_i} + \frac{\lambda_i^2 (1/\mu_i^2 + \sigma_i^2)}{2(1 - \lambda_i/\mu_i)} \qquad \forall i \in N.$$

The use of this queueing model is based on the assumption that, for each $i \in N$, service times follow a general distribution with mean service time $1/\mu_i$ and standard deviation σ_i and arrival rates follow a Poisson distribution with the parameter λ_i. Since arrivals at different sites depend on the other trucks and the operation of other sites, the Poisson distribution is an approximation. Although not necessarily representative of the true arrival rates, this approximation improves on the assumption that there is no queueing and does so in a computationally tractable way.

We can exploit the convexity of the M/G/1 queueing constraint by reformulating it as a second-order cone constraint, which can be handled efficiently with commercial solvers.

We rewrite the queueing constraint by introducing auxiliary variables s and t:

$$s_i = L_i - \frac{\lambda_i}{\mu_i} \qquad \forall i \in N \tag{3.16}$$

$$t_i = 2\left(1 - \frac{\lambda_i}{\mu_i}\right) \qquad \forall i \in N \tag{3.17}$$

$$s_i t_i \geq (1/\mu_i^2 + \sigma_i^2)\lambda_i^2 \qquad \forall i \in N \tag{3.18}$$

$$s_i, t_i \geq 0 \qquad \forall i \in NP. \tag{3.19}$$

We also bound the total number of trucks in the system at any given time:

$$\sum_{i \in N} L_i + \sum_{i \in N} \sum_{j \in N} \sum_{k \in K} t_{ij} x_{ijk} \leq \sum_{k \in K} \Gamma_k. \tag{3.20}$$

Inequality (3.20) states that the average number of trucks waiting or being served plus the average number of trucks traveling cannot exceed the total number of truck.

We also enforce non-negativity on all decision variables:

$$v_i^\ell, v_i^u, \lambda_i, L_i \geq 0 \qquad \forall i \in N \tag{3.21}$$

$$x_{ijk} \geq 0 \qquad \forall i \in N, \forall j \in N, \forall k \in K \tag{3.22}$$

$$f_{jr}^\ell, f_{jr}^u \geq 0 \qquad \forall j \in P, \forall r \in R. \tag{3.23}$$

All decision variables given in this section are summarized in Table 3.2. To simplify notation, let X be the set of solutions which satisfy all constraints:

$$X := \{y = (x,\ v^\ell,\ v^u\ \lambda,\ f^\ell,\ f^u,\ L) : (3.1) - (3.23)\}.$$

Then a solution $y \in X$ is a feasible solution to the average flow-rate model.

Variable	Meaning	Units
x_{ijk}	Flow rate of trucks of type $k \in K$ from $i \in N$ to $j \in N$	Trucks/hour
v_i^ℓ	Violation of minimum loading/unloading rate bound at site $i \in N$	Tons/hour
v_i^u	Violation of maximum loading/unloading rate bound at site $i \in N$	Tons/hour
λ_i	Arrival rate of trucks at location $i \in N$	Trucks/hour
f_{jr}^ℓ	Violation of minimum grade for attribute $r \in R$ at production site $j \in P$	Tons
f_{jr}^u	Violation of maximum grade for attribute $r \in R$ at production site $j \in P$	Tons
L_i	Average number of trucks waiting in queue or being loaded or unloaded at location $i \in N$	Trucks

Table 3.2: A summary of all decision variables used in the flow rate model

Objective

We use a three-phase hierarchical objective structure to model the relative importance of each objective. This hierarchy prioritizes meeting processing rates, followed by mining targets, and finally ore quality targets if possible. The most important goal is to keep the processor supplied with ore at all times, so we first solve the model with all constraints minimizing deviation from processing volume targets:

$$\min_{y \in X} \sum_{i \in P} (v_i^\ell + v_i^u).$$

Let $\hat{v_j^\ell}$ and $\hat{v_j^u}$ be the values of the processing rate variables in a solution to this problem. We bound the variables v_j^ℓ and v_j^u for $j \in P$ and minimize the maximum deviation from the mining rate targets:

$$\begin{aligned} &\min_{y \in X} \max_{i \in M} \{v_i^\ell, v_i^u\} \\ &\text{s.t. } v_j^\ell \leq \hat{v_j^\ell}, v_j^u \leq \hat{v_j^\ell} \qquad \forall j \in J. \end{aligned}$$

If any of the processing rate violation variables are positive, we terminate with the current

solution. Otherwise, after solving this problem, we store the values $(\hat{v}_i^\ell, \hat{v}_i^u)$ for all mining sites $i \in M$ and introduce bounds on the mining target violations for each mining site based on the solution. The bounds on mining targets are:

$$v_i^\ell \leq \hat{v}_i^\ell + 0.01\underline{p}_i \ \forall i \in M \tag{3.24}$$

$$v_i^u \leq \hat{v}_i^u + 0.01\underline{p}_i \ \forall i \in M. \tag{3.25}$$

We fix the processing rate violation variables to 0 and solve the final phase:

$$\begin{aligned} &\min_{y \in X} \sum_{j \in P} \sum_{r \in R} (f_{jr}^\ell + f_{jr}^u) \\ &\text{s.t.} v_j^\ell = 0, v_j^u = 0 \qquad\qquad \forall j \in P \\ &\qquad v_i^\ell, v_i^u \text{ satisfy } (3.24 - 3.25) \ \forall i \in M. \end{aligned}$$

We solve the average flow-rate model at regular intervals to find target flow rates for each pair of locations. We denote these target rates for each truck type $k \in K$ by r_{ijk} for an origin-destination pair (i, j). The time between successive re-solves of the nonlinear flow-rate model is a parameter that we tune as part of the computational study in Section 3.4.

Greedy Target-Matching Heuristic

Having found a solution to the average flow model with flow values r, we need a policy for making real-time dispatching decisions to match the target rates. Let the simulation length be T and denote the current time t, for $t \in [0, T]$. We denote the total flow of trucks in the simulation that have traveled between two locations by time t by f_{ij} for each origin-destination pair $(i, j) \in N \times N$.

We first consider a greedy target-matching heuristic dispatching policy. Each time a dispatching decision must be made, we check the list of possible round trips for the available truck starting at the truck's location, $i \in N$. For each possible sequence of destinations

$(j,k) \in N \times N$, we compute the difference between the target flow on that round trip at the current time t, $(r_{ij} + r_{jk})(t/T)$, and the actual flow $(f_{ij} + f_{jk})$. We choose $\hat{j}$ such that $\hat{j} \in \arg\max_j\{(r_{ij}(t/T) - f_{ij}) + (r_{jk}(t/T) - f_{jk})\}$ for all possible $(i,j,k) \in N \times N$ with $i \in N$ fixed, and dispatch the available truck to $\hat{j}$. Finally, we update actual flow for that route: $f_{i\hat{j}} \leftarrow f_{i\hat{j}} + 1$. If two or more locations satisfy the argmax requirement, we break the tie arbitrarily.

Although computationally simple, the greedy heuristic is myopic in that only a single truck's next round-trip is considered. Since no other trucks are considered, we cannot take advantage of combining strategies for different trucks. In fact, the heuristic could end up sending all trucks to a single location if all the routes leading to that location were to have large differences between target and actual flow rates, resulting in long queues and waiting times.

MIP-Based Target-Matching Heuristic

We next propose a target-matching heuristic that accounts for other trucks in the system and how the current decision affects the near future. This heuristic involves solving a small assignment-type integer program based on current and target flow rates over round trips.

When a truck becomes available, we also determine where other trucks in the system will become available in the next $\tilde{T}$ units of time and denote the total trucks by n. For each of the n trucks, we build a list of all the possible round trips that a truck could take (two origin-destination pairs: $(i,j),(j,k)$). We denote this list of possible round trips by A_p for $p \in [1,\cdots,n]$. We also define a set Q_a as the set of origin-destination pairs in round trip a for each $a \in A$. We then solve a small MIP to assign trucks to round trips. Let y_{ij} be the total number of trucks assigned to each origin-destination pair $(i,j) \in N \times N$ and let x_{pa} be a binary variable such that $x_{pa} = 1$ if truck p is assigned to route a and 0 otherwise. Let t be the current time plus $\tilde{T}$, so that $r_{ij}(t/T)$ is the target flow on route (i,j) by the time all

n trucks are available for dispatching. The full model is

$$\min_{x,y}\Big[\max\{\sum_{(i,j)\in N\times N}(r_{ij}(t/T)-y_{ij}),0\}\Big] \tag{3.26}$$

$$\text{s.t.}\sum_{a\in A_p}x_{pa}=1 \qquad \forall p=1,\cdots,n \tag{3.27}$$

$$\sum_{p=1}^{n}\sum_{a:(i,j)\in Q_a}x_{pa}+f_{ij}=y_{ij} \qquad \forall (i,j)\in N\times N \tag{3.28}$$

$$x_{pa}\in\{0,1\},\ y_{ij}\geq 0 \qquad \forall a\in A,\forall p=1,\cdots,n,\forall (i,j)\in N\times N. \tag{3.29}$$

The currently available truck (say $p=1$) should then be dispatched on the route a such that $x_{1a}=1$. The following trucks that become available could also be dispatched according to the solution, or the assignment problem could be solved again every time a truck becomes available. The look-ahead time $\tilde{T}$ affects the quality of the solutions found by this heuristic, so we explore possible choices of $\tilde{T}$ in the computational study given in Section 3.4.

This heuristic improves on some aspects of the myopic nature of the single-truck heuristic. An assignment made with this heuristic reduces total violation by accounting for more than one truck. In addition, we could easily introduce a constraint on the total number of trucks routed to a single location in a given period of time, limiting issues with queueing. One downside to using this heuristic rather than the single-truck heuristic is that it is more computationally intensive than the single-truck heuristic, since it requires solving a small MIP every time a dispatching decision must be made.

We note that as they were described, these heuristics assume $|K|=1$, but both can be easily extended to account for multiple truck types by creating a copy of the policy for each type.

Discrete Time MIP-Based Dispatching Policy

We next propose a dispatching policy based on the MIP dispatching model from Chapter 2. We solve the dispatching problem using the RINS heuristic detailed in Chapter 2.

Modifications to the MIP Dispatching Model

We make some small changes to the discrete-time MIP dispatching model to more closely match the requirements we impose on the open-pit mine simulation. First, although it is almost always possible to meet the processing rate targets when the mine is operating in a steady state, the same is not necessarily true when the mine is simulated as a dynamic stochastic environment. When this occurs, it is possible in some instances to have unreasonable mining strategies in the PHASE I solution. To account for this, we solve the PHASE II problem of minimizing the maximum deviation from the mining extraction targets regardless of whether the processing rate targets can be met. If the processing rate violation variables are non-zero after solving the PHASE I problem, we use their values as upper bounds on the processing rate violation and minimize the maximum mining target violation. However, if the processing targets are not met in PHASE I, we do not continue on to PHASE III where we minimize deviation of the ore quality targets. Not meeting the processing targets is considered an emergency state, so in this situation the volume of ore processed matters much more than the quality of ore. PHASE II is only solved to avoid unreasonable mining strategies in this situation.

Next, we have an additional goal of avoiding situations in which the processors must slow down. To incorporate this in the MIP dispatching model, we add the following new constraints for each $j \in P$:

$$I_{jt} \geq \ell \qquad \forall j \in P, t \geq \lfloor T^{\max}/2 \rfloor. \tag{3.30}$$

Constraint (3.30) requires that after the halfway point of the planning horizon, the ore in

each short-term stockpile is at least the slow-down threshold ℓ. Note that it might not be physically possible to meet these constraints in all cases, but we model the inventory and processing constraints in such a way that the model is always feasible. In particular, the constraint for the inventory level at time $t \in [0, T^{\max}]$ at processor $j \in P$ is

$$I_{jt} = I_{jt-1} - (\Theta^P_{jt} - h^L_{jt}) + \sum_{k \in K, i \in G} \alpha_k y^P_{ijkt} \qquad \forall t \in T, \forall j \in P. \tag{3.31}$$

In this constraint, we calculate the current inventory level as the previous period's inventory level minus the amount processed plus the amount added. If we remove the restriction that $\Theta^P_{jt} - h^L_{jt} \geq 0$, we can always increase h^L_{jt} enough to meet the inventory requirements in any period. Of course, if $\Theta^P_{jt} - h^L_{jt} < 0$ we claim to have negative production, which is not physically possible. However, because we minimize $\sum_{j \in P, t \in [0, T^{\max}]} h^L_{jt}$, anytime it is feasible to meet the processing rate targets and the inventory constraints, these variables will be 0. Thus, we model the processors slowing down as not meeting the target processing rate.

We also describe how we would modify the model to account for end-of-shift effects. A small modification could be made to incorporate the requirement that the short-term stockpiles are nearly full at the end of a shift. To do so, we replace the constraint that the ending inventory level at processor $j \in P$ at simulation time $t \in [0, T^{\text{shift}}]$ is at least as great as the starting inventory with the following constraint:

$$I_{j,T^{\max}} \geq 0.99(t/T^{\text{shift}})C_j \qquad \forall j \in P, \tag{3.32}$$

where $I_{j,T^{\max}}$ is the tons of ore in the inventory at processor j at the end of the planning horizon used in the discrete-time MIP dispatching model and C_j is the capacity of the short-term stockpile. The fraction t/T^{shift} represents the fraction of the shift that has transpired, so that when $t = T^{\text{shift}}$ the short-term stockpiles will need to be nearly full.

A MIP-Based Dispatching Policy

We next describe how the discrete-time MIP dispatching model from Chapter 2 can be used to derive a dispatching policy. At regular intervals, we solve the MIP model using the current data from the simulation. A solution to the the dispatching problem includes the flow variable values, which contain all the dispatching decisions for the planning horizon specified in the model; specifically, x_{ijkt} indicates how many trucks of type k from location $i \in N$ are sent to $j \in N$ in time period t. For each location $i \in N$, and each truck type $k \in K$, we create a list of destinations at each time period t in the planning horizon:

$$L_{ikt} = \underbrace{j_1, \ldots, j_1}_{x_{ij_1kt}}, \underbrace{j_2, \ldots, j_2}_{x_{ij_2kt}}, \ldots, \underbrace{j_{|N|}, \ldots, j_{|N|}}_{x_{ij_{|N|}kt}}.$$

Thus, each list contains exactly x_{ijkt} copies of each destination $j \in N$. Then for each site $i \in N$ and for each truck type k, we have an ordered list of all the destinations to which trucks should be sent from i:

$$\mathcal{L}_{ik} = \{L_{ik,0}, L_{ik,1}, \ldots, L_{ik,T^{\max}}\}.$$

An iterator at each location $i \in N$ for each truck type $k \in K$ increments every time a dispatching decision is made at that location and indicates which list index to reference for the next dispatching decision at i with truck type k.

A key parameter in the use of this policy is how often the MIP model should be re-solved. Certainly, any time a major event occurs in the system, such as an equipment failure, the data should be updated and the model should be solved again. In normal operation, however, the correct timing of the re-solves is less clear. We investigate possible choices for the re-solve frequency as part of the computational study described in Section 3.4.

3.4 Computational Experiments

We conduct a set of computational experiments to better understand how each of the three policies described in Section 3.3 performs on open-pit mine instances with different characteristics. We begin by tuning each policy individually, finding reasonable choices of method parameters. Next we evaluate the performance of the queueing constraints in the nonlinear flow-rate model and the ore quality and truck capacity constraints in the discrete-time MIP dispatching model. Finally, we compare the tuned policies by constructing instances of the open-pit mine simulation with different numbers of trucks, increasing variance in the ore quality across different mining sites, increasing variance in travel times between pairs of locations, and increasing variance in the distance between the mining sites and the processing sites.

The models are implemented in Python 2.7 and solved with Gurobi 7.0.2 using a 1% optimality gap tolerance. We solve the discrete-time MIP dispatching problem using the techniques described in Chapter 2. We impose a 90-second time limit on solving the small MIP in the MIP-based target-matching policy. The simulation is implemented in Python 2.7 using SimPy 3.0. This study is conducted on an Intel Xeon X5650 server with 12 cores at 2.66Ghz and 128GB of RAM using 24 threads. Excluding the parameters specified hereafter, all Gurobi parameters are left in their default state.

Policy Input Parameter tuning

We begin by conducting a computational experiment for each individual dispatching policy to determine good choices of method parameters. Specifically, we tune the following parameters:

Greedy target-matching policy (G)

1. **Re-solve interval:** This parameter determines how much simulation time passes between successive re-solves of the nonlinear flow-rate model. The values we test are

5, 10, 15, 30, 60, 120, and 240 minutes.

MIP-based target-matching policy (M)

1. **Re-solve interval:** This parameter determines how much simulation time passes between successive re-solves of the nonlinear flow-rate model. The values we test are 15, 30, 60, 120, and 240 minutes.

2. **Look-ahead time for truck-to-round-trip matching:** This parameter corresponds to how far into the future we look to build an instance of the small MIP described in Section 3.3 that matches trucks to round trips. Specifically, we check how many trucks will become available in this amount of time and thus must be matched to a possible round trip. The values we test are 5, 10, 15, and 20 minutes.

We tune these parameters in combination, so each possible pair is tested.

Discrete-time MIP-based dispatching policy (D)

1. **Re-solve interval:** This parameter determines how much simulation time passes between successive re-solves of the discrete-time MIP dispatching model. The values we test are 2, 5, 10, and 15 minutes.

2. **Planning horizon:** This parameter is the planning horizon used in each instance of the discrete-time MIP dispatching model. Because the MIP model is a time-discretized model with increments of 30 seconds, increasing horizon length corresponds to increasing model size (and difficulty). The values we test are 25, 30, and 35 minutes.

As with the MIP-based target-matching policy, we tune these parameters in combination.

For each policy and each combination of parameters, we run 30 replications of the open-pit mine simulation on an instance with $|G| = 5$, $|B| = 4$, $|P| = 2$, $|S| = 2$, $|D| = 2$, $|R| = 1$, and 35 trucks of a single size. The simulation is run over the first 10 hours of a 12-hour shift, thus excluding the end-of-shift effects. We denote the length of the simulation

by T. The target processing volume over the 10-hour simulation is fixed at 200,000 tons. All other parameters used to build an instance of the open-pit mine and the distributions used to generate random variables can be found in Appendix B.1. To evaluate each choice of parameter, we construct 95% confidence intervals on the several key metrics.

Metrics to measure processing target violation:

- For what percent of time did the processing sites have to be shut down? In a given replication, we calculate this value by denoting the total length of time in which a processor was shut down by S_j for $j \in P$. Then the average shut-down percent is:

$$P^{sh} := 100\% \times \frac{\sum_{j \in P} S_j}{|P|T}.$$

- For what percent of time did the processors need to be slowed to 60% of their normal rate? We calculate this by denoting the length of time in which a processor was slowed down by W_j for $j \in P$. Then the average slow-down percent is:

$$P^{sl} := 100\% \times \frac{\sum_{j \in P} W_j}{|P|T}.$$

- How much ore (in tons) was processed over the entire shift? Let $r_j(t)$ be the rate (in tons) ore is processed at time $t \in [0, T]$ at processor $j \in P$. This value is calculated as:

$$P^{vol} = \sum_{j \in P} \int_0^T r_j(t) dt.$$

Metrics to measure mining target violation:

- Across all mining sites, what is the maximum amount (in tons) by which we fall short of the target? If θ_j^T is the target extraction from mine $j \in M$ and θ_j^A is the actual tons

extracted from mine j, we calculate this value as:

$$V^U = \max_{j \in M} \frac{\max\{0, \theta_j^T - \theta_j^A\}}{\theta_j^T}.$$

- Across all mining sites, what is the maximum amount (in tons) by which we exceed the target? We calculate this value as:

$$V^O = \max_{j \in M} \frac{\max\{0, \theta_j^A - \theta_j^T\}}{\theta_j^T}.$$

Metrics to measure ore quality target violation:

- What percent of the total ore processed (at both processing sites) is in an interval of width 1.0 percentage point around the target ore quality? Let $r_j(t)$ be the rate ore is processed (in tons) at processor $j \in P$ at time $t \in [0, T]$. Let $\delta_j^{|Q^A - Q^T| < 0.005}(t)$ be an indicator function that is 1 if the absolute value of the difference between the actual ore quality (Q^A) and target ore quality (Q^T) is less that 0.005 at processor $j \in P$ at time $t \in [0, T]$. We calculate this value as:

$$Q_{0.005} = 100\% \times \sum_{j \in P} \frac{\int_0^T r_j(t) \delta_j^{|Q^A - Q^T| < 0.005}(t) dt}{|P| P^{vol}}.$$

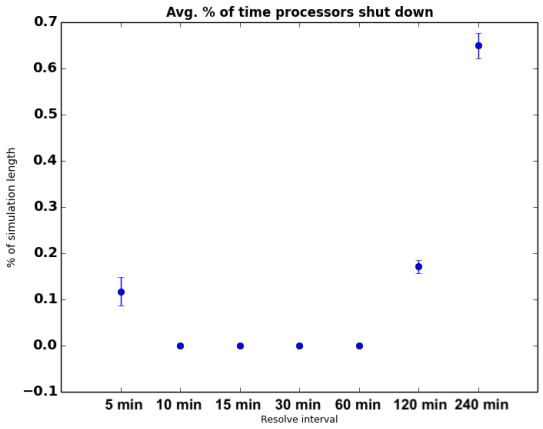
- What percent of the total ore processed (at both processing sites) is in an interval of width 2.0 percentage points around the target ore quality? This value is calculated in the same way as the previous one, but the indicator function is 1 as long as the absolute difference between the actual ore quality and target ore quality is less than 0.01:

$$Q_{0.01} = 100\% \times \sum_{j \in P} \frac{\int_0^T r_j(t) \delta_j^{|Q^A - Q^T| < 0.01}(t) dt}{|P| P^{vol}}.$$

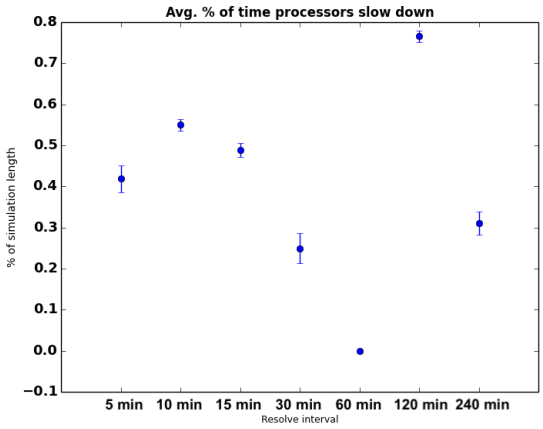
Because keeping the processors running at full speed is critically important to mining operations, we consider the ore quality metrics to be secondary to the processing metrics. Although it is preferable to process higher-quality ore, it would be better to process low-quality ore than no ore at all (or too little ore). Similarly, it is possible to have some mining sites exceed their mining extraction targets in order to keep the processors fed. Typically this means other mines will fall short of their extraction targets, but if the over-mine and under-mine values are not unreasonably high, some deviation is acceptable.

We first display the results for the greedy target-matching policy. Figure (3.1a) shows the percent of time processors are shut down and Figure (3.1b) shows the percent of time processors are slowed down for each choice of parameter. Figure (3.1c) displays the processing volume for each choice of parameter. Figures (3.2a)-(3.2b) display the amount under-mined and over-mined for each choice of parameter, respectively. Finally, the percent of the processed ore within half a percentage point of the target ore quality for each choice of parameter is shown in Figure (3.3a). The percent of ore within one percentage point of the target ore quality is shown in Figure (3.3b).

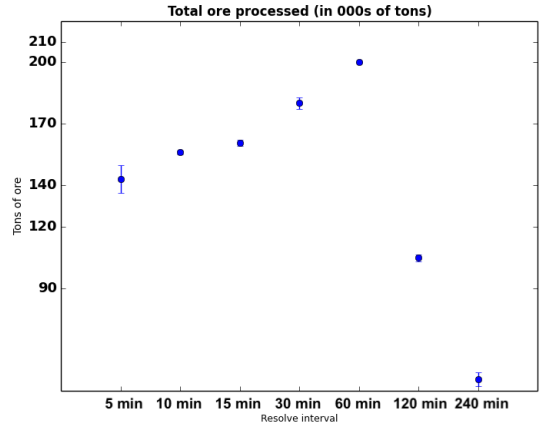
From Figures (3.1a)-(3.1c), we observe the smallest percent of time the processors must slow down or stop occurs with a 60-minute re-solve frequency. In Figures (3.2a)-(3.2b), the distinction between re-solve intervals is less clear. A 120-minute re-solve interval appears to under-mine the least and does not over-mine at all. However, a 120-minute re-solve interval produces the second-lowest processing volume.The ore quality metrics (Figures (3.3a)-(3.3b)) are of lesser importance but clearly reinforce the the choice of 60 minutes as the best re-solve interval. We note that a re-solve interval of 60 minutes falls between the lowest and highest values on the mining metrics. Thus, we conclude that 60 minutes is a reasonable choice of the re-solve frequency parameter.



(a) 95% confidence intervals on fraction of time processors shut down for 30 replications with different input parameters.

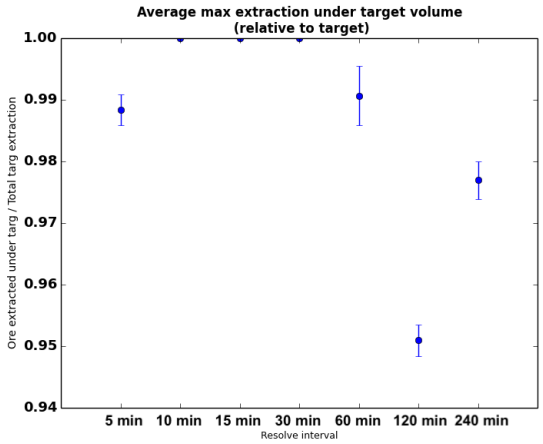


(b) 95% confidence intervals on fraction of time processors slowed down for 30 replications with different input parameters.

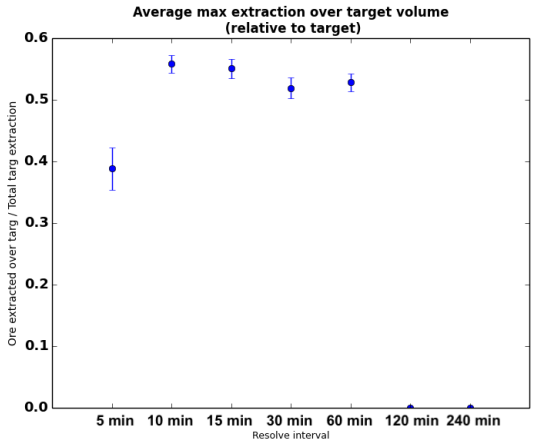


(c) 95% confidence intervals on total ore processed at one processing site for 30 replications with different input parameters. Target is 200,000 tons.

Figure 3.1: Parameter choices for greedy target-matching dispatching policy on processing metrics

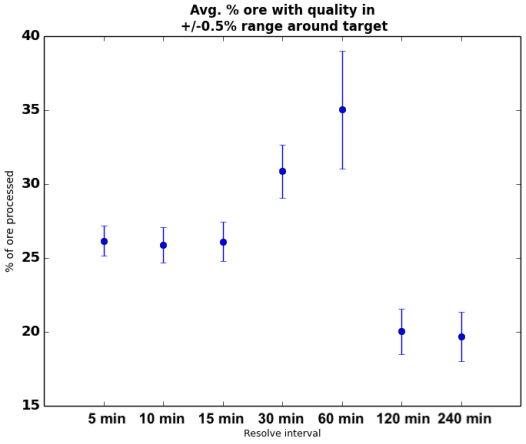


(a) 95% confidence intervals on maximum mining target shortfall across all mining sites (relative to target extraction) for 30 replications with different input parameters.

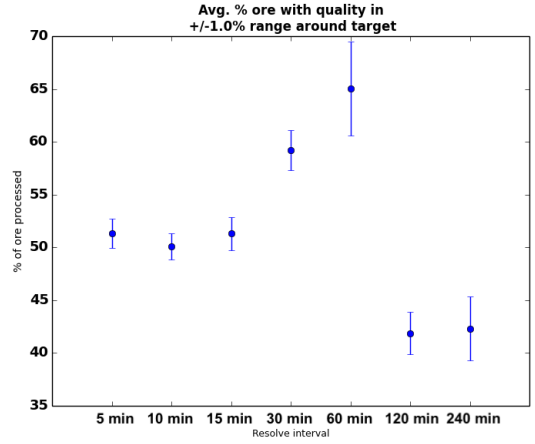


(b) 95% confidence intervals on maximum excess of mining targets across all mining sites (relative to target extraction) for 30 replications with different input parameters.

Figure 3.2: Parameter choices for greedy target-matching dispatching policy on mining metrics



(a) 95% confidence intervals on percent of ore processed within $\pm$ 0.5% of ore quality target for 30 replications with different input parameters.

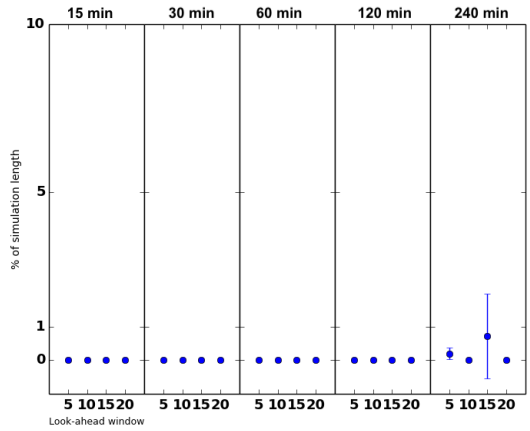


(b) 95% confidence intervals on percent of ore processed within $\pm$ 1.0% of ore quality target for 30 replications with different input parameters.

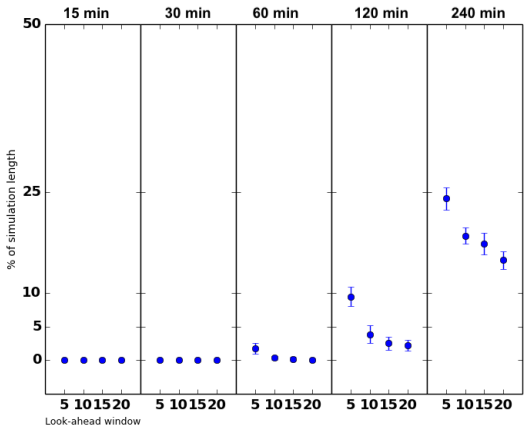
Figure 3.3: Parameter choices for greedy target-matching dispatching policy on ore quality metrics

Next we display the results for the MIP-based target-matching policy. Figures (3.4a)-(3.4b) show the percent of time processors are shut and slowed down, respectively. Figure (3.4c) displays the processing volume for each choice of parameter. Figures (3.5a)-(3.5b) display the amount under-mined and over-mined for each choice of parameter, respectively. Finally, the percent of the processed ore within half a percentage point of the target ore quality for each choice of parameter is shown in Figure (3.6a). The percent of ore within one percentage point of the target ore quality is shown in Figure (3.6b).

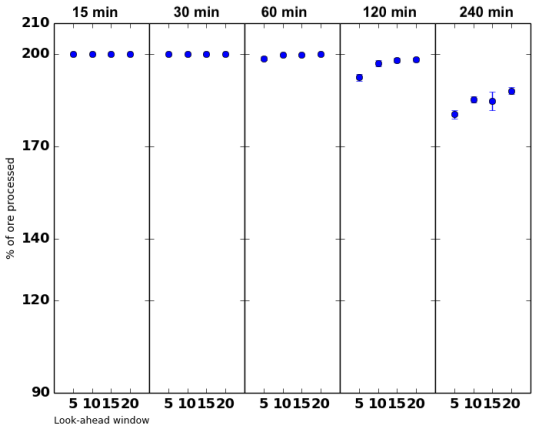
When evaluating parameters for the MIP-based target-matching policy, we first observe in Figures (3.4c)-(3.4b) that shorter re-solve intervals combined with longer look-ahead windows meet processing targets best. For example, a re-solve interval of 60 minutes with a 15- or 20-minute look ahead window is able to meet processing targets. At a re-solve interval of 120 minutes, there is a slight decrease in processing volume. The amount of time the processors must be slowed increases with increasing re-solve interval length. In general, longer look-ahead windows correspond to higher processing volume. However, from Figures (3.5a)-(3.5b), we observe that the mining targets can be better met with longer re-solve intervals and shorter look-ahead windows. Finally, from Figures (3.6a)-(3.6b) we observe that ore quality increases with increasing re-solve interval length. There is no clear difference in the ore quality metrics between different lengths of the look-ahead window. From these results, we conclude that a re-solve interval of 60 minutes and a look-ahead window of 20 minutes offers a reasonable trade-off between processing, mining, and ore quality targets.



(a) 95% confidence intervals on fraction of time processors shut down for 30 replications with different input parameters.

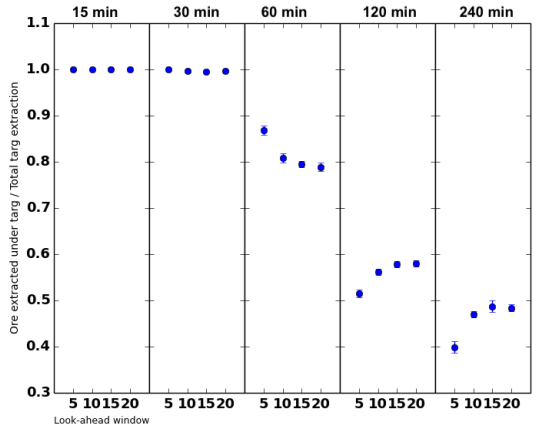


(b) 95% confidence intervals on fraction of time processors slowed down for 30 replications with different input parameters.

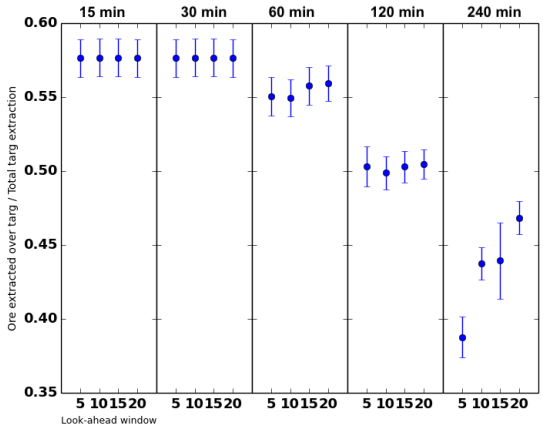


(c) 95% confidence intervals on total ore processed at one processing site for 30 replications with different input parameters. Target is 200,000 tons.

Figure 3.4: Parameter choices for MIP-based target-matching dispatching policy on processing metrics

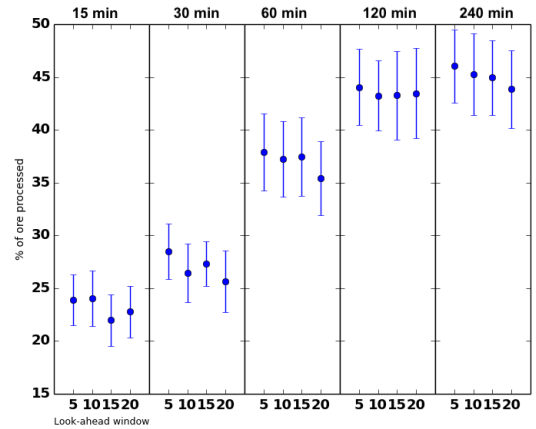


(a) 95% confidence intervals on maximum mining target shortfall across all mining sites (relative to target extraction) for 30 replications with different input parameters.

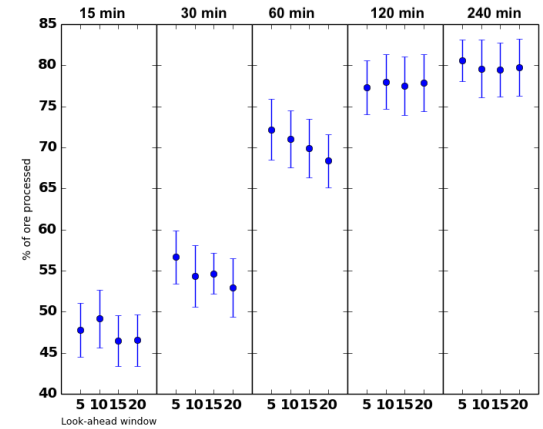


(b) 95% confidence intervals on maximum excess of mining targets across all mining sites (relative to target extraction) for 30 replications with different input parameters.

Figure 3.5: Parameter choices for MIP-based target-matching dispatching policy on mining metrics



(a) 95% confidence intervals on percent of ore processed within $\pm$ 0.5% of ore quality target for 30 replications with different input parameters.



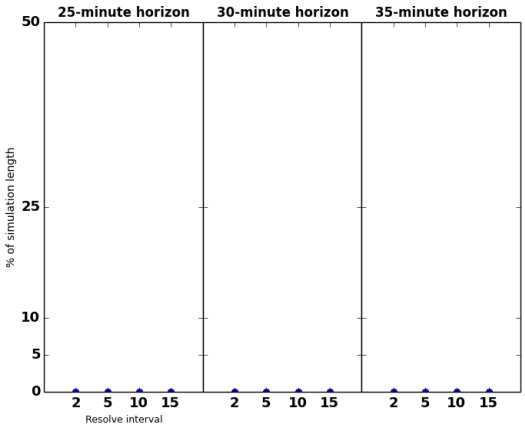
(b) 95% confidence intervals on percent of ore processed within $\pm$ 1.0% of ore quality target for 30 replications with different input parameters.

Figure 3.6: Parameter choices for MIP-based target-matching dispatching policy on ore quality metrics

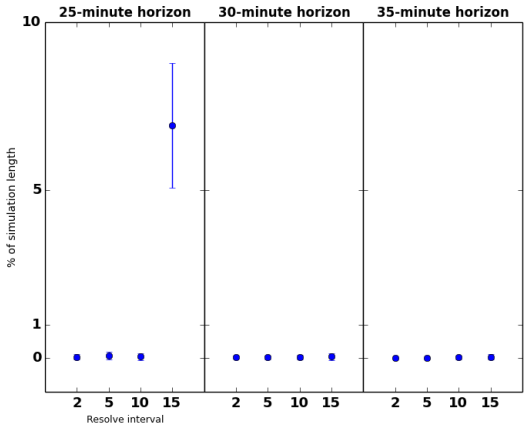
Finally, we display the results for the discrete-time MIP-based policy. Figures (3.7a)-(3.7b) show the percent of time processors are shut and slowed down, respectively. Figure (3.7c) displays the processing volume for each choice of parameter. Figures (3.8a)-(3.8b) display the amount under-mined and over-mined for each choice of parameter, respectively. Finally, the percent of the processed ore within half a percentage point of the target ore quality for each choice of parameter is shown in Figure (3.9a). The percent of ore within one percentage point of the target ore quality is shown in Figure (3.9b).

We see from Figures (3.7a)-(3.7c) that there is no strong correlation between planning horizon length, re-solve frequency, and processing target metrics. Planning horizons of 30 and 35 minutes are able to meet processing targets for any re-solve frequency. A planning horizon of 25 minutes meets processing targets for all but the 15-minute re-solve frequency. In Figure (3.8a), there is no clear correlation between horizon, re-solve frequency, and maximum violation of the mining targets. In Figure (3.8b), we observe that there is a slight increase in the maximum amount over-mined for higher re-solve frequencies. From the ore quality results in Figures (3.9a)-(3.9b), we note that more frequent re-solves tend to yield higher ore quality in the 25- and 35-minute planning horizons, but lower ore quality in the 30-minute planning horizon. We conclude that a planning horizon of 25 minutes and a re-solve frequency of 5 minutes results in a reasonable trade-off between the processing, mining, and ore quality objectives.

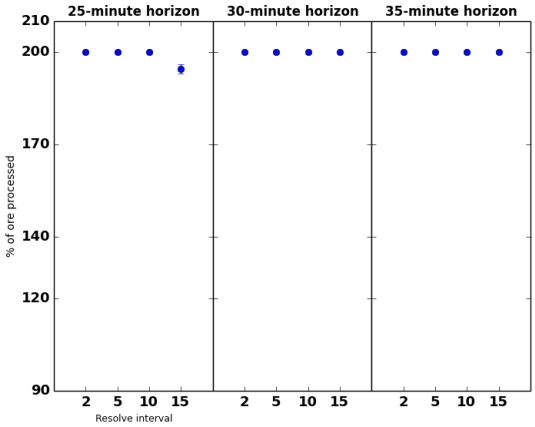
For all experiments discussed hereafter, we use a re-solve frequency of 60 minutes for the target-matching policies and a 20-minute look-ahead window for the MIP-based target-matching policy. We use a 25-minute planning horizon and a 5-minute re-solve frequency for the discrete-time MIP policy.



(a) 95% confidence intervals on fraction of time processors shut down for 30 replications with different input parameters.

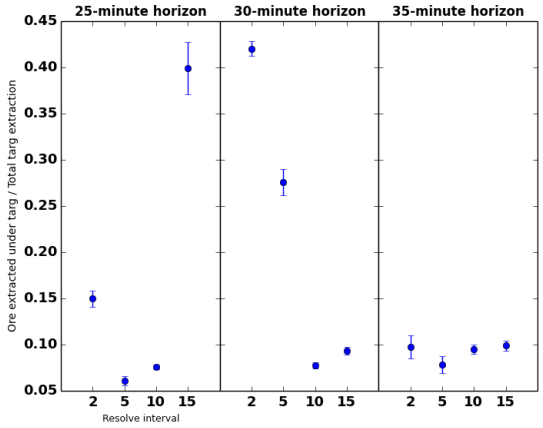


(b) 95% confidence intervals on fraction of time processors slowed down for 30 replications with different input parameters.

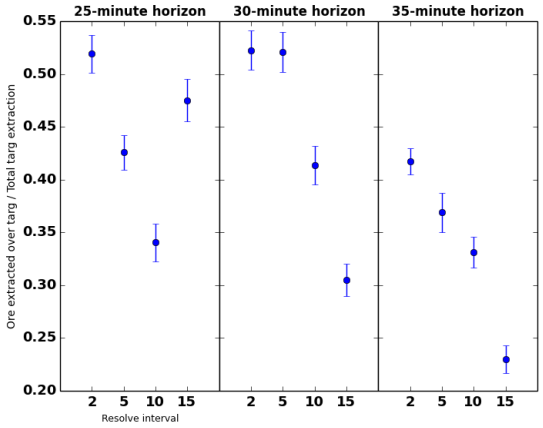


(c) 95% confidence intervals on total ore processed at one processing site for 30 replications with different input parameters. Target is 200,000 tons.

Figure 3.7: Parameter choices for discrete-time MIP dispatching policy on processing metrics

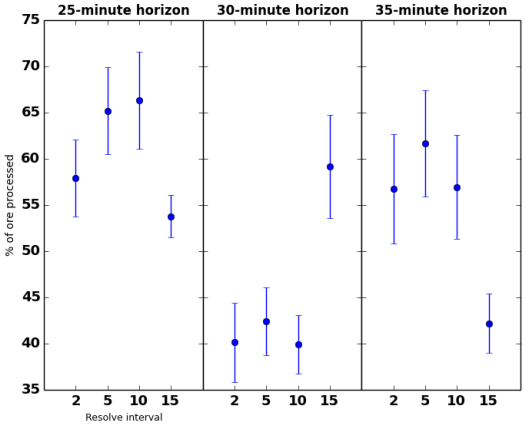


(a) 95% confidence intervals on maximum mining target shortfall across all mining sites (relative to target extraction) for 30 replications with different input parameters.

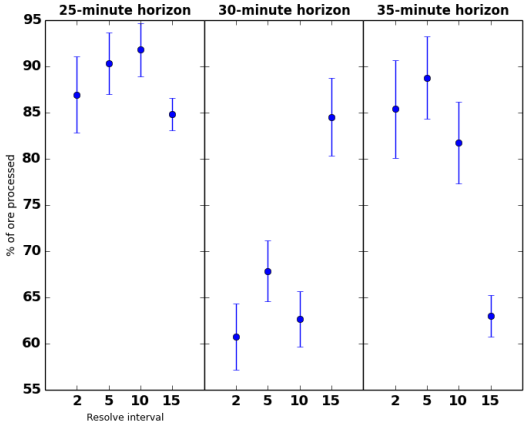


(b) 95% confidence intervals on maximum excess of mining targets across all mining sites (relative to target extraction) for 30 replications with different input parameters.

Figure 3.8: Parameter choices for discrete-time MIP dispatching policy on mining metrics



(a) 95% confidence intervals on percent of ore processed within ± 0.5% of ore quality target for 30 replications with different input parameters.



(b) 95% confidence intervals on percent of ore processed within ± 1.0% of ore quality target for 30 replications with different input parameters.

Figure 3.9: Parameter choices for discrete-time MIP dispatching policy on ore quality metrics

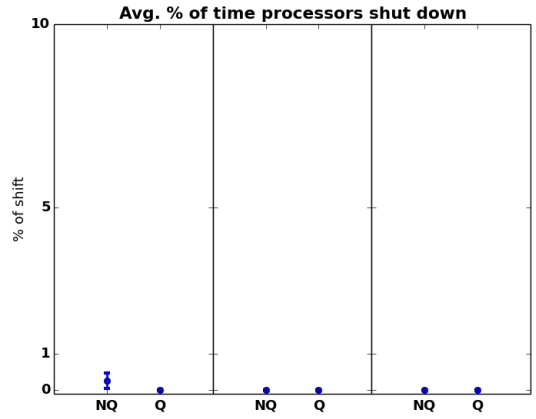
Evaluating Model Components

We next evaluate the different modeling choices we made in both the nonlinear flow-rate model and the discrete-time MIP dispatching model. In particular, we test the effect of the queueing constraints in the nonlinear flow-rate model (constraints (3.16)-(3.19)) and the moving window truck capacity and ore quality constraints in the MIP dispatching model described in Chapter 2. We evaluate each policy on an instance with $|G| = 5$, $|B| = 4$, $|P| = 2$, $|S| = 2$, $|D| = 2$, $|R| = 1$, and varying numbers of trucks. The simulation is run over the first 10 hours of a 12-hour shift, excluding the end-of-shift effects. The target processing volume over the 10-hour simulation is fixed at 200,000 tons. All other parameters used to build an instance of the open-pit mine and the distributions used to generate random variables can be found in Appendix B.1. We evaluate each policy on the same metrics described at the beginning of this section.

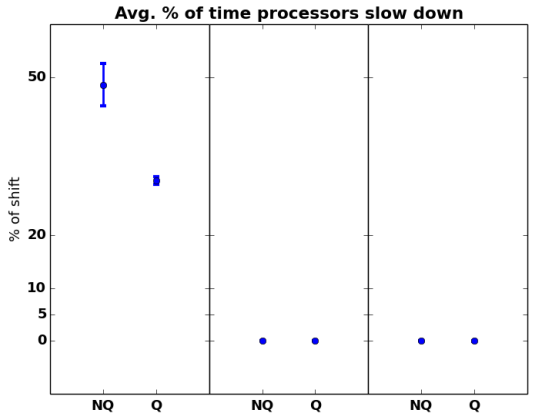
Evaluating effectiveness of constraints in the target-matching flow-rate dispatching policies

We begin by evaluating how the use of the queueing constraints in the nonlinear flow rate model affects the key metrics. We run 30 replications of the open-pit mine simulation both with and without the queueing constraints to examine the policies' performance. We repeat the experiment for a fixed instance with 15, 25, and 35 trucks.

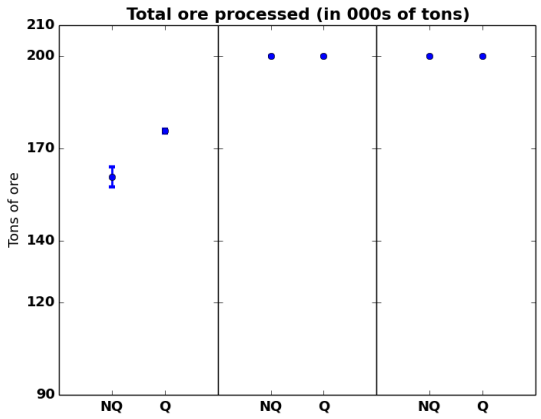
Queuing constraints in the greedy target-matching policy To evaluate the effectiveness of queueing constraints in the nonlinear flow rate model for the greedy target-matching policy, we run the simulation first without any consideration of queueing (NQ). We then run the simulation by using rotated second-order cone constraints (3.16)-(3.19)) to approximate the effects of queueing (Q). Figures (3.10a)-(3.12b) display the results of this study.



(a) 95% confidence intervals on fraction of time processors shut down for 30 replications of greedy target-matching policy with queueing constraints (Q) and without the constraints (NQ).

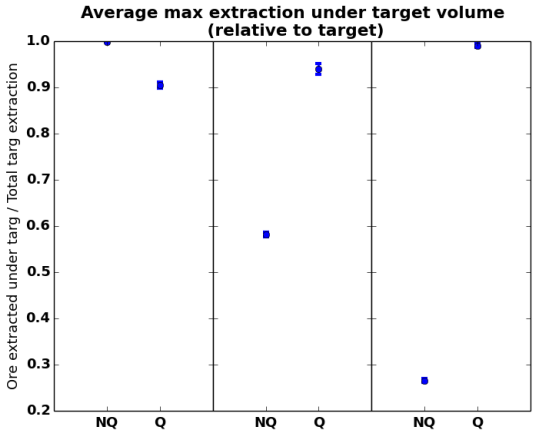


(b) 95% confidence intervals on fraction of time processors slowed down for 30 replications of greedy target-matching policy with queueing constraints (Q) and without the constraints (NQ).

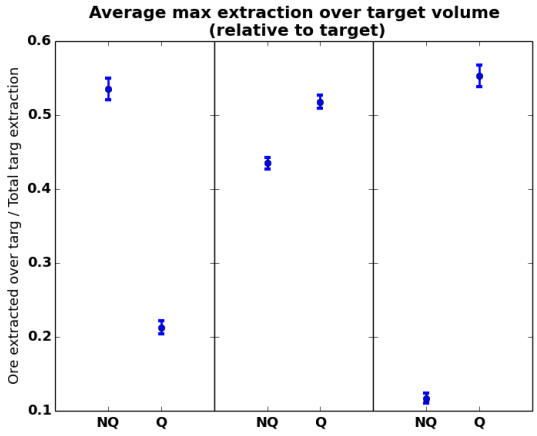


(c) 95% confidence intervals on total ore processed for 30 replications of greedy target-matching policy with queueing constraints (Q) and without the constraints (NQ).

Figure 3.10: Queuing constraints in nonlinear flow-rate model with greedy target-matching policy on processing metrics

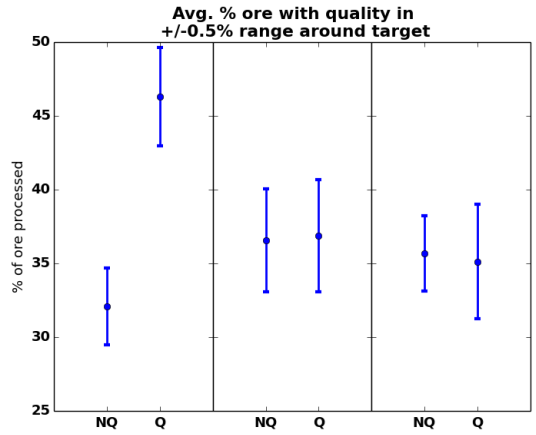


(a) 95% confidence intervals on maximum mining target shortfall across all mining sites (relative to target extraction) of greedy target-matching policy with queueing constraints (Q) and without the constraints (NQ).

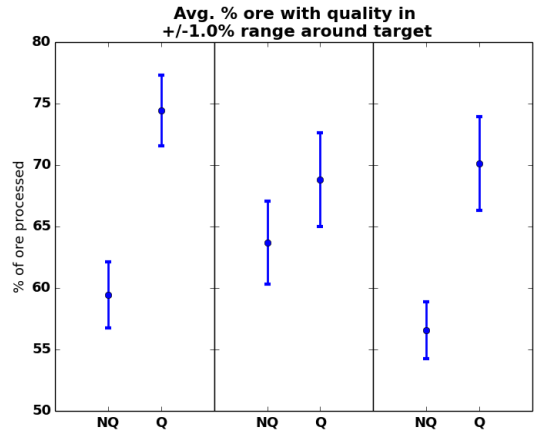


(b) 95% confidence intervals on maximum excess of mining targets across all mining sites (relative to target extraction) of greedy target-matching policy with queueing constraints (Q) and without the constraints (NQ).

Figure 3.11: Queuing constraints in nonlinear flow-rate model with greedy target-matching policy on mining metrics



(a) 95% confidence intervals on percent of ore processed within ± 0.5% of ore quality target for 30 replications of greedy target-matching policy with queueing constraints (Q) and without the constraints (NQ).

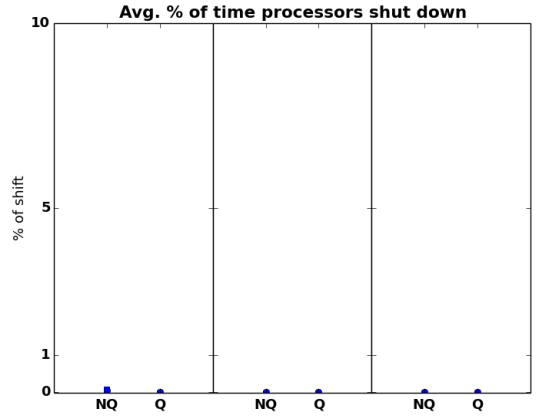


(b) 95% confidence intervals on percent of ore processed within ± 1.0% of ore quality target for 30 replications of greedy target-matching policy with queueing constraints (Q) and without the constraints (NQ).

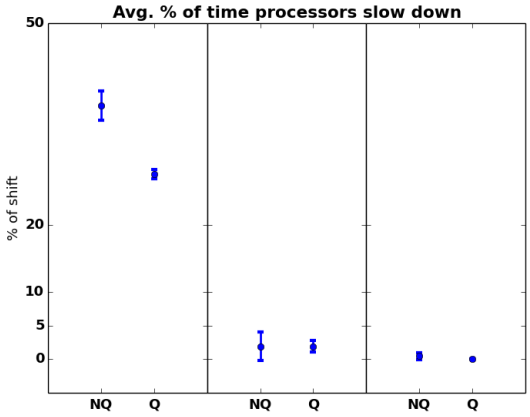
Figure 3.12: Queuing constraints in nonlinear flow-rate model with greedy target-matching policy on ore quality metrics

We observe that the queueing constraints make no difference in our ability to meet processing targets for 25 and 35 trucks. When only 15 trucks are available, however, use of queueing constraints significantly increases the processing volume. We also observe higher ore quality and less deviation from mining targets in the 15-truck instance with queueing constraints. As the number of trucks increases to 25 (and 35), removing the queueing constraints appears to increase our ability to meet mining targets. With 25 and 35 trucks, the ore quality metrics improve slightly with the queueing constraints. We conclude that the queueing constraints are critical to meeting the ore processing objective when few trucks are available, but use of the constraints has less impact when sufficiently many trucks are available and may make it more difficult to meet mining targets.

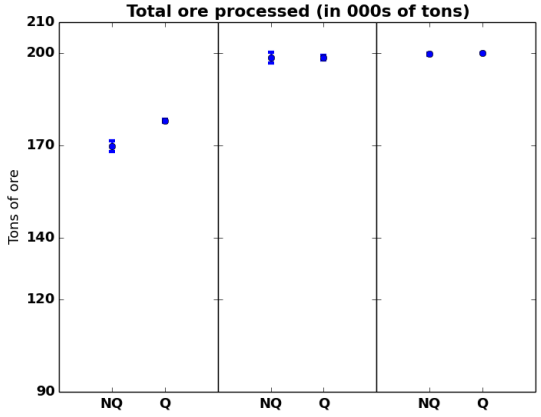
Queuing constraints in the MIP-based target matching policy We repeat the queueing constraint study with the MIP-based flow rate target matching policy in place of the greedy target-matching policy. We run the simulation first without any consideration of queueing (NQ). We then run the simulation by using rotated second-order cone constraints to approximate the effects of queueing (Q). Figures (3.13a)-(3.15b) display the results of this study.



(a) 95% confidence intervals on fraction of time processors shut down for 30 replications of MIP-based target-matching policy with queueing constraints (Q) and without the constraints (NQ).

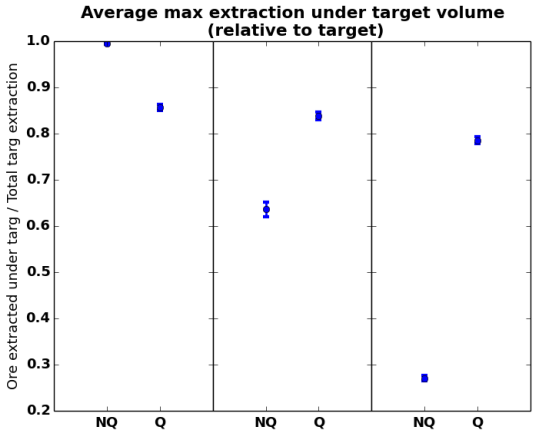


(b) 95% confidence intervals on fraction of time processors slowed down for 30 replications of MIP-based target-matching policy with queueing constraints (Q) and without the constraints (NQ).

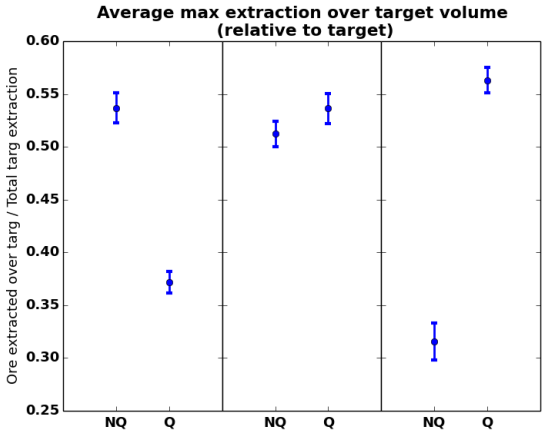


(c) 95% confidence intervals on total ore processed for 30 replications of MIP-based target-matching policy with queueing constraints (Q) and without the constraints (NQ).

Figure 3.13: Queuing constraints in nonlinear flow-rate model with MIP-based target-matching policy on processing metrics

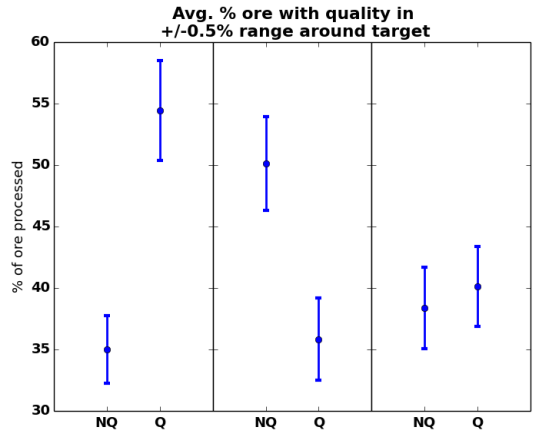


(a) 95% confidence intervals on maximum mining target shortfall across all mining sites (relative to target extraction) of MIP-based target-matching policy with queueing constraints (Q) and without the constraints (NQ).

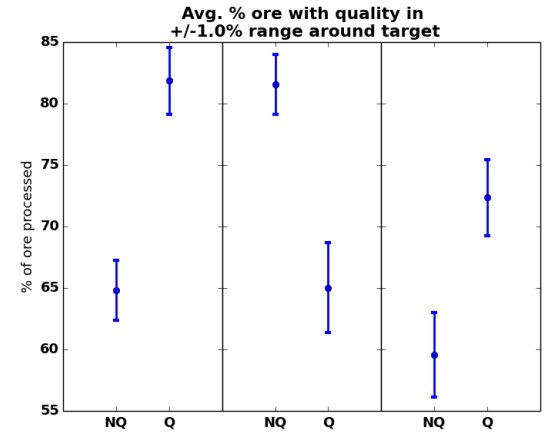


(b) 95% confidence intervals on maximum excess of mining targets across all mining sites (relative to target extraction) of MIP-based target-matching policy with queueing constraints (Q) and without the constraints (NQ).

Figure 3.14: Queuing constraints in nonlinear flow-rate model with MIP-based target-matching policy on mining metrics



(a) 95% confidence intervals on percent of ore processed within $\pm$ 0.5% of ore quality target for 30 replications of MIP-based target-matching policy with queueing constraints (Q) and without the constraints (NQ).



(b) 95% confidence intervals on percent of ore processed within $\pm$ 1.0% of ore quality target for 30 replications of MIP-based target-matching policy with queueing constraints (Q) and without the constraints (NQ).

Figure 3.15: Queuing constraints in nonlinear flow-rate model with MIP-based target-matching policy on ore quality metrics

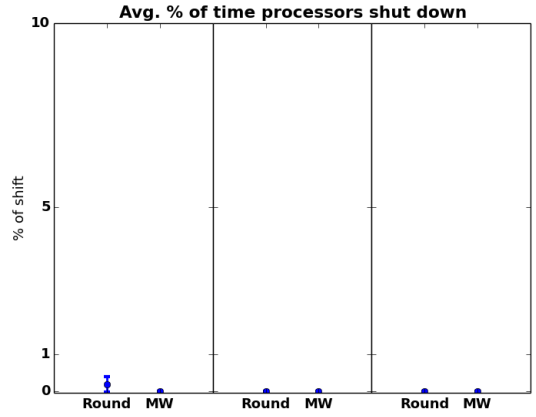
We observe that the queueing constraints make no significant difference in our ability to meet processing targets for 25 and 35 trucks. When we have 15 trucks, however, the use of queueing constraints significantly increases the processing volume. We also observe higher ore quality and less deviation from mining targets in the 15-truck instance with queueing constraints. As the number of trucks increases to 25 and 35, removing the queueing constraints appears to improve our ability to meet mining targets. With 25 trucks, the ore quality metric is significantly higher for the variation without queueing constraints, but the the ore quality metrics improve with queueing constraints for the 35-truck instance. We conclude that the queueing constraints are critical to meeting the ore processing objective when few trucks are available and can also improve the ore quality metrics. However, use of queueing constraints appears to make it more difficult to meet the mining targets when more trucks are available. This indicates that our approximation of queuing may be overly conservative, constraining the system more than necessary when many trucks are available.

Evaluating effectiveness of constraints in the discrete-time MIP-based dispatching policy

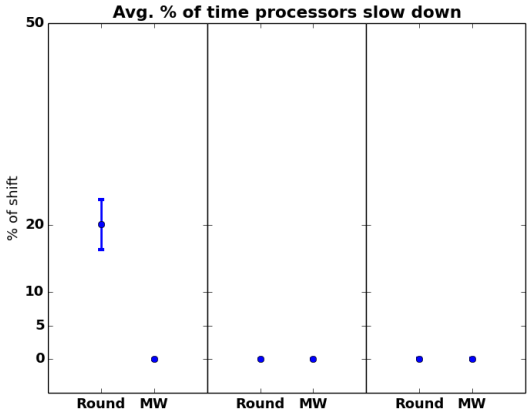
Next, we examine the effect of the use of moving window capacity and ore quality constraints in the discrete-time MIP dispatching model. We run 30 replications of the open-pit mine simulation both with and without the capacity constraints and with and without the quality constraints to examine the policy's performance. We repeat the experiment for a fixed instance with 15, 25, and 35 trucks.

Capacity constraints in the MIP dispatching model To evaluate the effectiveness of the moving window capacity constraints, we run the simulation first by simply rounding up the truck capacity in each time period at each location to the nearest integer (Round). We then run the simulation with moving window constraints that attempt to limit the

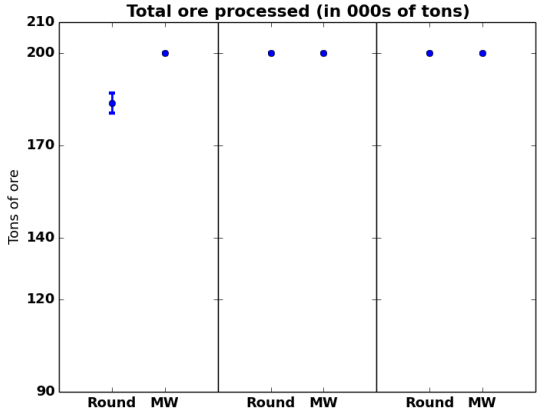
"round-up error" (MW). These constraints are described in the MIP dispatching model details in Chapter 2. Figures (3.16a)-(3.18b) display the results of this study.



(a) 95% confidence intervals on fraction of time processors shut down for 30 replications of discrete-time MIP policy with moving window capacity constraints (MW) and without the constraints (Round).

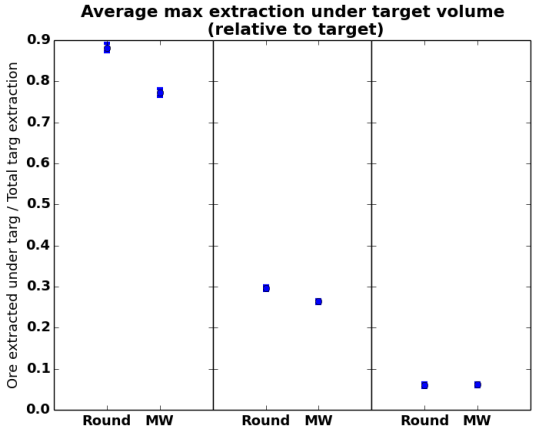


(b) 95% confidence intervals on fraction of time processors slowed down for 30 replications of discrete-time MIP policy with moving window capacity constraints (MW) and without the constraints (Round).

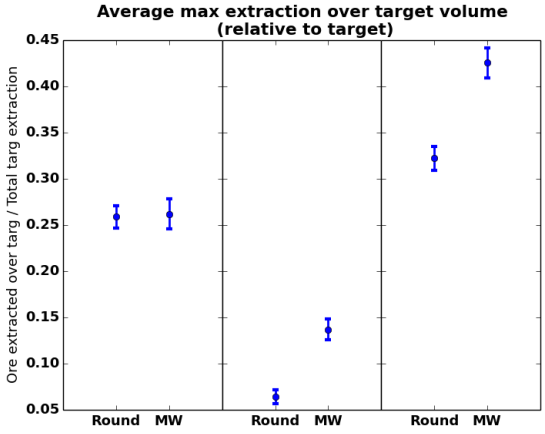


(c) 95% confidence intervals on total ore processed for 30 replications of discrete-time MIP policy with moving window capacity constraints (MW) and without the constraints (Round).

Figure 3.16: Capacity constraints in discrete-time MIP dispatching policy on processing metrics

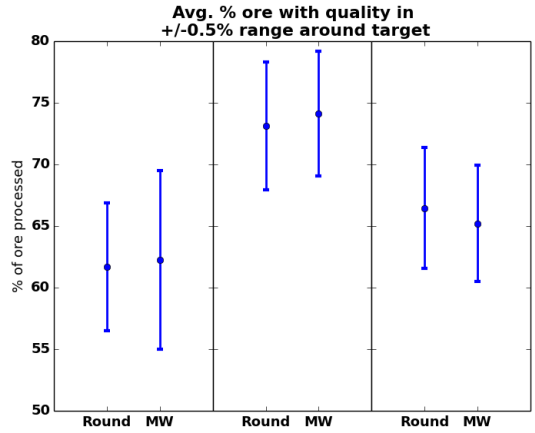


(a) 95% confidence intervals on maximum mining target shortfall across all mining sites (relative to target extraction) of discrete-time MIP policy with moving window capacity constraints (MW) and without the constraints (Round).

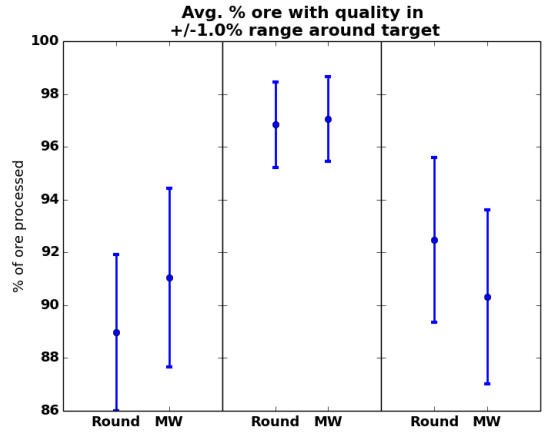


(b) 95% confidence intervals on maximum excess of mining targets across all mining sites (relative to target extraction) of discrete-time MIP policy with moving window capacity constraints (MW) and without the constraints (Round).

Figure 3.17: Capacity constraints in discrete-time MIP dispatching policy on mining metrics



(a) 95% confidence intervals on percent of ore processed within $\pm$ 0.5% of ore quality target for 30 replications of discrete-time MIP policy with moving window capacity constraints (MW) and without the constraints (Round).

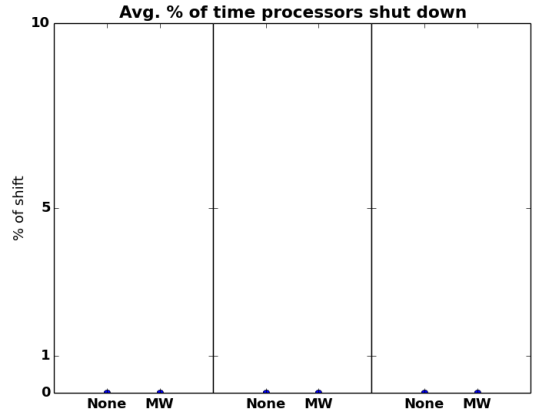


(b) 95% confidence intervals on percent of ore processed within $\pm$ 1.0% of ore quality target for 30 replications of discrete-time MIP policy with moving window capacity constraints (MW) and without the constraints (Round).

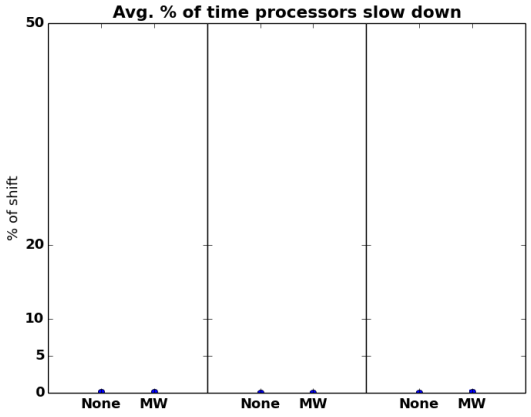
Figure 3.18: Capacity constraints in discrete-time MIP dispatching policy on ore quality metrics

We observe that the capacity constraints make no difference in our ability to meet processing targets for 25 and 35 trucks. When only 15 trucks are available, however, use of capacity constraints increases the processing volume. The two variations are nearly indistinguishable on both the mining and ore quality metrics, although the MW variation does appear to over-mine slightly more than simply rounding up. We conclude that the moving window capacity constraints are critical to meeting the ore processing objective when few trucks are available, but use of the constraints has less impact when sufficiently many trucks are available.

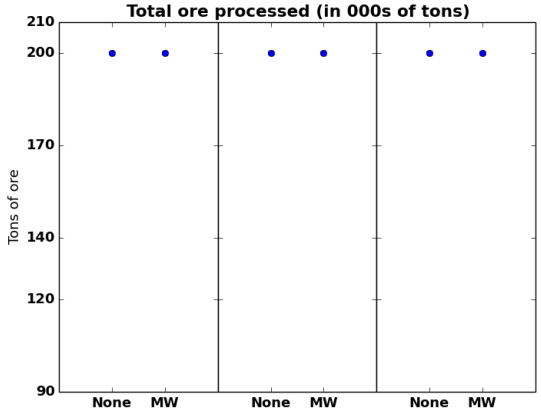
Ore quality constraints in the MIP dispatching model To evaluate the effectiveness of the moving window ore quality constraints, we run the simulation first by stopping after minimizing mining target deviation rather than continuing to minimize ore quality target violation (None). We then run the simulation by using moving window constraints to approximate ore quality violation (MW). These constraints are described in the MIP dispatching model details in Chapter 2. Figures (3.19a)-(3.21b) display the results of this study.



(a) 95% confidence intervals on fraction of time processors shut down for 30 replications of discrete-time MIP policy with moving window ore quality constraints (MW) and without the constraints (None).

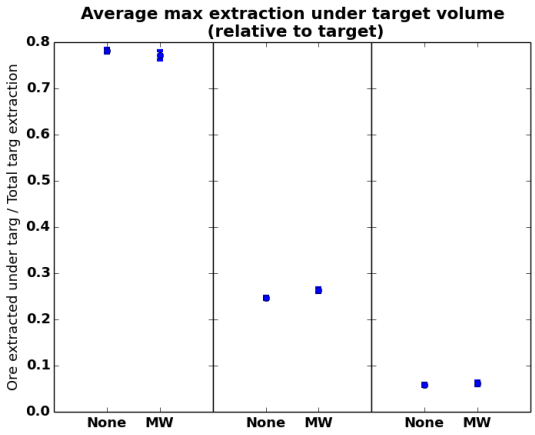


(b) 95% confidence intervals on fraction of time processors slowed down for 30 replications of discrete-time MIP policy with moving window ore quality constraints (MW) and without the constraints (None).

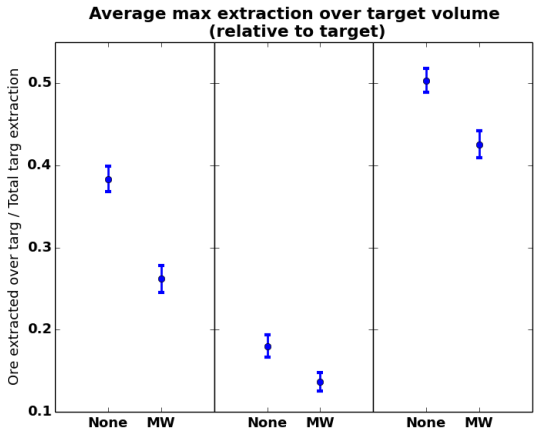


(c) 95% confidence intervals on total ore processed for 30 replications of discrete-time MIP policy with moving window ore quality constraints (MW) and without the constraints (None).

Figure 3.19: Ore quality constraints in discrete-time MIP dispatching policy on processing metrics

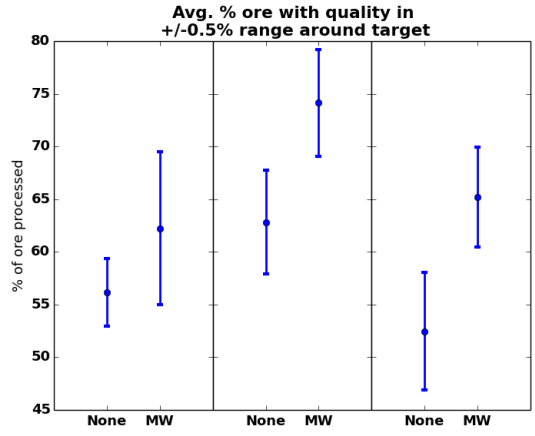


(a) 95% confidence intervals on maximum mining target shortfall across all mining sites (relative to target extraction) of discrete-time MIP policy with moving window ore quality constraints (MW) and without the constraints (None).

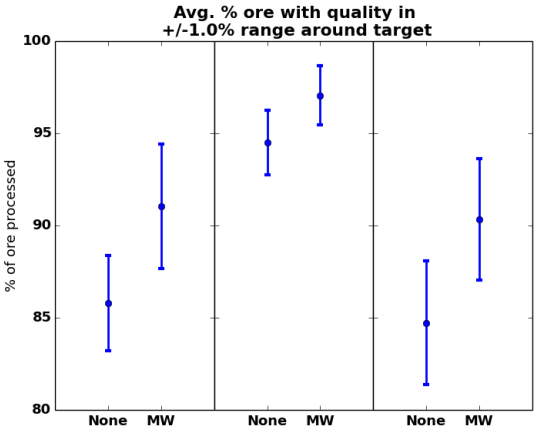


(b) 95% confidence intervals on maximum excess of mining targets across all mining sites (relative to target extraction) of discrete-time MIP policy with moving window ore quality constraints (MW) and without the constraints (None).

Figure 3.20: Ore quality constraints in discrete-time MIP dispatching policy on mining metrics



(a) 95% confidence intervals on percent of ore processed within ± 0.5% of ore quality target for 30 replications of discrete-time MIP policy with moving window ore quality constraints (MW) and without the constraints (None).



(b) 95% confidence intervals on percent of ore processed within ± 1.0% of ore quality target for 30 replications of discrete-time MIP policy with moving window ore quality constraints (MW) and without the constraints (None).

Figure 3.21: Ore quality constraints in discrete-time MIP dispatching policy on ore quality metrics

The use of the moving window ore quality constraints appears to have no impact on meeting processing targets, regardless of the number of trucks. Interestingly, although we would expect the MW variation to perform slightly worse on the mining metrics, the two variations are nearly indistinguishable in terms of under-mining. In fact, use of the MW constraints in the simulation actually seems to reduce the maximum amount over-mined. As expected, the ore quality values significantly improve when using the moving window ore quality constraints regardless of the number of trucks. We conclude that the moving window ore quality constraints are critical to better meeting the ore quality targets and, surprisingly, might even improve the mining metrics.

Comparing Tuned Policies

We conclude by comparing the performance of the tuned greedy target-matching policy (G), MIP-based target-matching policy (M), and discrete-time MIP policy (D) on the same metrics as described at the beginning of this section. We perform a study of the policies on mines with different characteristics, varying instance parameters that influence the performance of the policies. Specifically, we vary the number of trucks in the mine, the variance of the means of the ore quality at mining sites, the variance of the truck travel times, and the distance of the mines to the processing sites. We first construct a ”base” open-pit mine instance and change individual input parameters. The ”base” open-pit mine has $|G| = 5$, $|B| = 4$, $|P| = 2$, $|S| = 2$, $|D| = 2$, $|R| = 1$, and 35 trucks of a single size. The simulation is run over the first 10 hours of a 12-hour shift, excluding the end-of-shift effects. The target processing volume over the 10-hour simulation is fixed at 200,000 tons. All other parameters used to build an instance of the open-pit mine and the distributions used to generate random variables can be found in Appendix B.1.

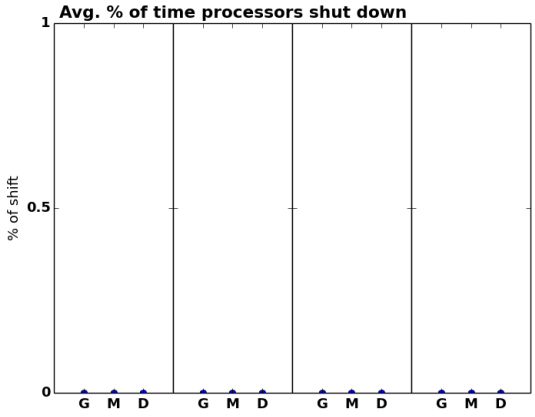
Varying numbers of trucks

We test each dispatching policy on instances of an open-pit mine with 15, 25, 35, and 45 trucks. Figures (3.22a)-(3.24b) display the output for the policy comparison on these instances.

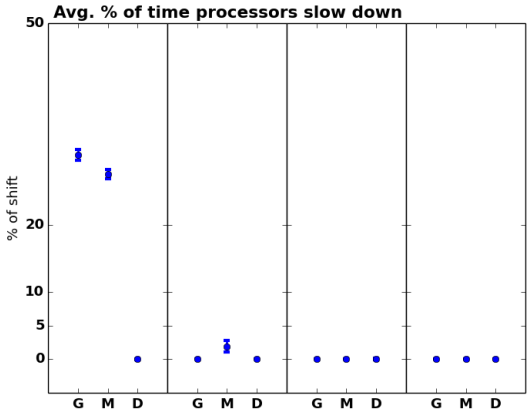
We first observe from Figures (3.22a) - (3.22c) that all policies avoid shutting down the processors, even with few trucks available. However, both the greedy and MIP-based target-matching policies must slow the processors in the 15-truck instance. The MIP-based target-matching policy slows the processors about 1% of the time in the 25-truck instance.

On the mining metrics (Figures (3.23a) - (3.23b)), the discrete-time MIP policy undermines significantly less than the MIP-based target-matching policy, which in turn undermines significantly less than the greedy target matching policy. In fact, the greedy target-matching policy ignores at least one mining site completely until there are at least 45 trucks available. The policies are difficult to distinguish on the over-mining metrics, although it appears that the MIP-based target-matching policy consistently over-mines the most. The discrete-time MIP policy is closest to meeting the mining targets in the 25- and 35-truck instances, and the greedy target-matching policy is closest in the 15 and 45 truck instances.

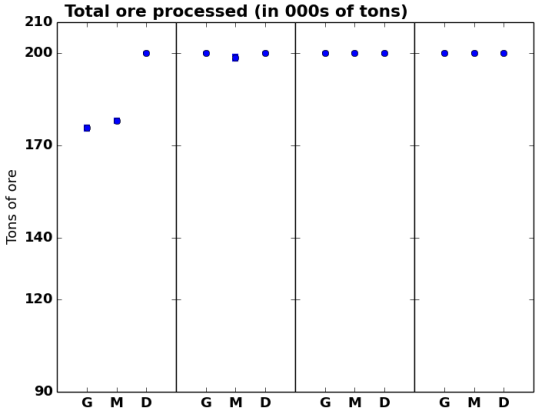
Finally, on the ore quality metrics (Figures (3.24a) - (3.24b)), we observe that the discrete-time MIP policy has the highest ore quality of the three policies for all instances. The greedy and MIP-based target matching policies are comparable in all but the 45-truck instance, in which the greedy policy significantly outperforms the MIP-based target-matching policy. This is likely due to the either having a too-short look-ahead window in the matching MIP to account for enough trucks to successfully match the target rates, or not solving the matching MIP to optimality within the 90-second time limit, resulting in sub-optimal matchings. We also note that in the 45-truck instance, the greedy policy and discrete-time MIP policy have similar ore quality.



(a) 95% confidence intervals on fraction of time processors shut down for 30 replications comparing three different dispatching policies for increasing numbers of trucks.

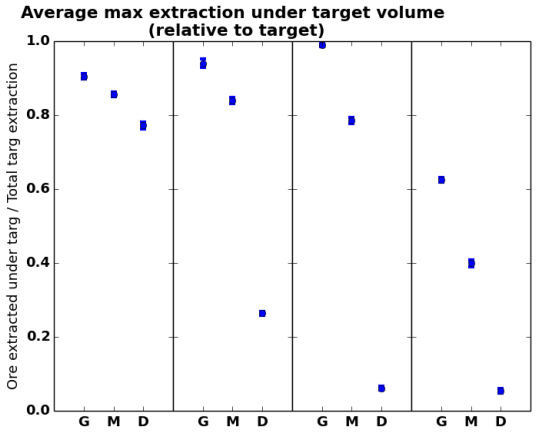


(b) 95% confidence intervals on fraction of time processors slowed down for 30 replications comparing three different dispatching policies for increasing numbers of trucks.

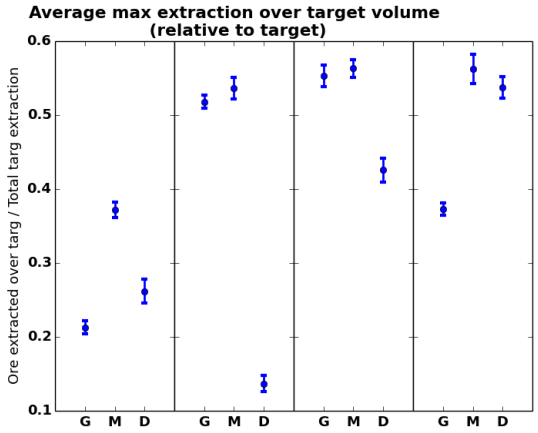


(c) 95% confidence intervals on total ore processed for 30 replications comparing three different dispatching policies for increasing numbers of trucks.

Figure 3.22: Comparing G, M, and D policies on processing metrics with 15, 25, 35, and 45 trucks.

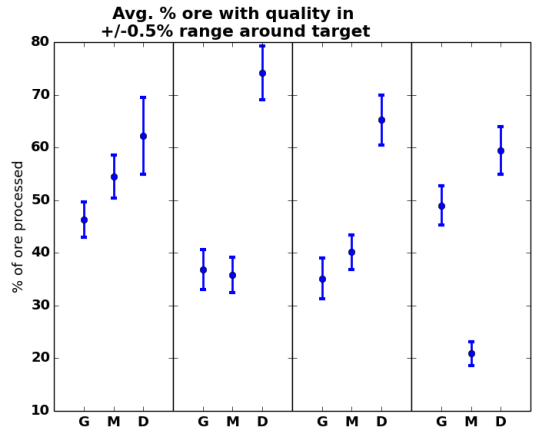


(a) 95% confidence intervals on maximum mining target shortfall across all mining sites (relative to target extraction) for 30 replications comparing three different dispatching policies for increasing numbers of trucks.

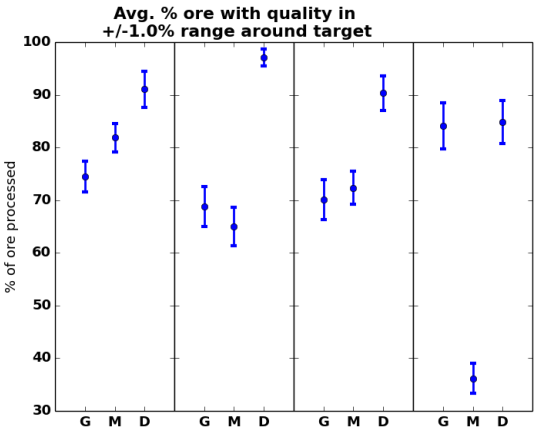


(b) 95% confidence intervals on maximum excess of mining targets across all mining sites (relative to target extraction) for 30 replications comparing three different dispatching policies for increasing numbers of trucks.

Figure 3.23: Comparing G, M, and D policies on mining metrics with 15, 25, 35, and 45 trucks.



(a) 95% confidence intervals on percent of ore processed within ± 0.5% of ore quality target for 30 replications comparing three different dispatching policies for increasing numbers of trucks.



(b) 95% confidence intervals on percent of ore processed within ± 1.0% of ore quality target for 30 replications comparing three different dispatching policies for increasing numbers of trucks.

Figure 3.24: Comparing G, M, and D policies on ore quality metrics with 15, 25, 35, and 45 trucks.

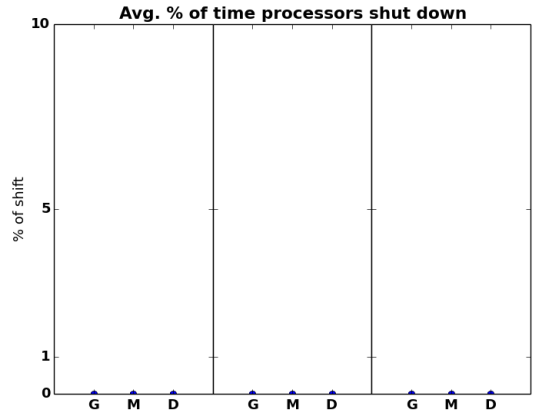
Varying the ore quality at mining sites

We also test each dispatching policy on instances of an open-pit mine with increasing variance in ore quality between mining sites. In the base instance, all mining sites have ore quality close to the target ore quality. In the second instance, we keep the mean quality across all mining sites close to the target quality, but allow the quality at individual mining sites to deviate from the mean by as much as 0.4 times the target quality. In the third instance, we again keep the mean quality close to the target quality but allow the quality at individual mining sites to deviate from the mean by as much as 0.7 times the target quality. Figures (3.25a)-(3.27b) display the output for the policy comparison on these instances.

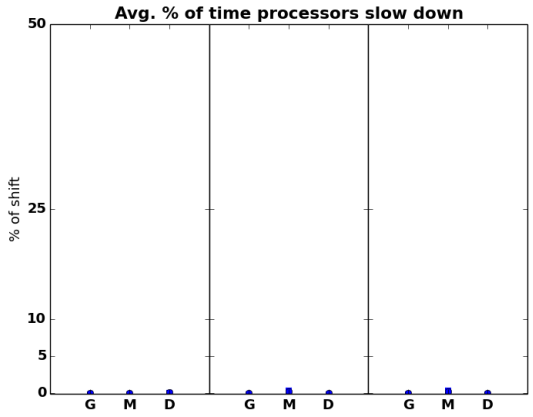
We first observe from Figures (3.25a) - (3.25c) that all policies avoid shutting and slowing down the processors, regardless of the variance of ore quality.

On the mining metrics (Figures (3.23a) - (3.23b)), the discrete-time MIP policy under-mines significantly less than the MIP-based target matching policy, which in turn under-mines significantly less than the greedy target matching policy for all instances. The discrete-time MIP policy also over-mines significantly less than the other two policies on all instances. There appears to be a reduction in the maximum amount over-mined for all policies when the ore quality variance increases from the base instance.

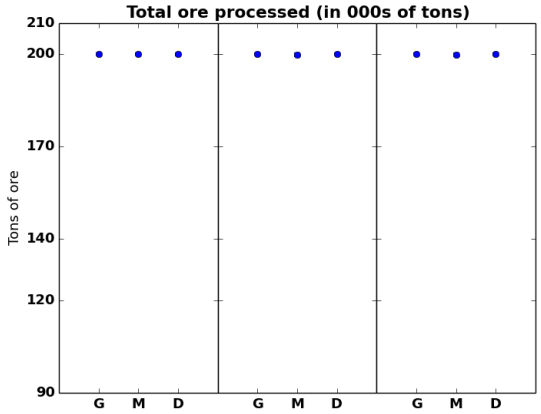
Finally, on the ore quality metrics (Figures (3.24a) - (3.24b)), we observe that the discrete-time MIP policy has the highest ore quality of the three policies for all instances. The MIP-based target matching policy has slightly higher ore quality than the greedy target-matching policy on all instances. As we would expect, we also observe a decrease in the ore quality as the variance increases, but the decrease is less significant for the discrete-time MIP policy than for the other two policies.



(a) 95% confidence intervals on fraction of time processors shut down for 30 replications comparing three different dispatching policies for increasing variance in ore quality across mining sites.

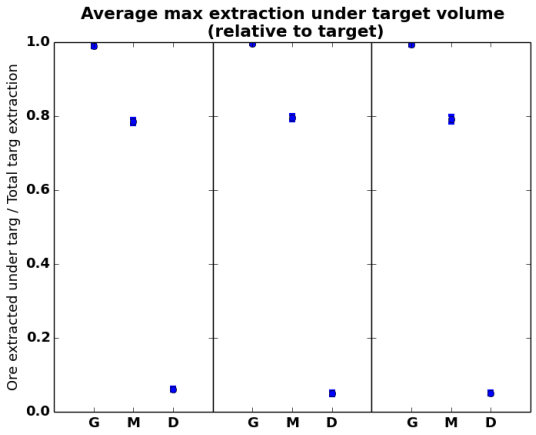


(b) 95% confidence intervals on fraction of time processors slowed down for 30 replications comparing three different dispatching policies for increasing variance in ore quality across mining sites.

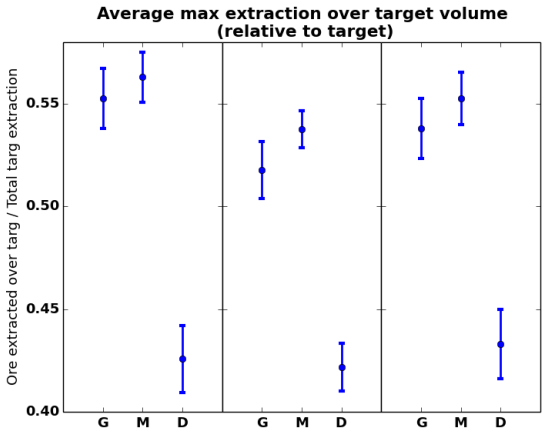


(c) 95% confidence intervals on total ore processed for 30 replications comparing three different dispatching policies for increasing variance in ore quality across mining sites.

Figure 3.25: Comparing G, M, and D policies on processing metrics with low, medium, and high variance in ore quality.

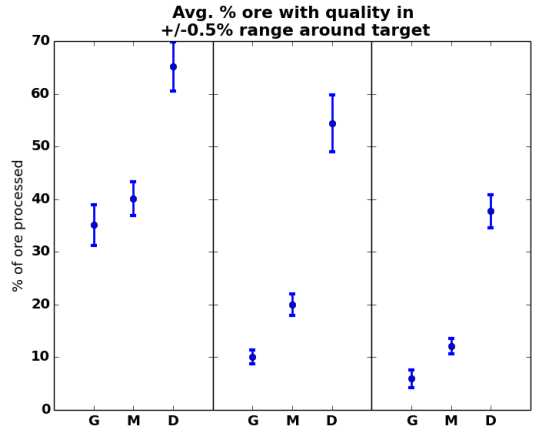


(a) 95% confidence intervals on maximum mining target shortfall across all mining sites (relative to target extraction) for 30 replications comparing three different dispatching policies for increasing variance in ore quality across mining sites.

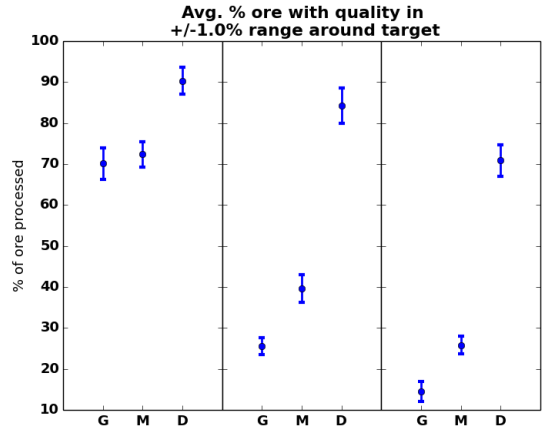


(b) 95% confidence intervals on maximum excess of mining targets across all mining sites (relative to target extraction) for 30 replications comparing three different dispatching policies for increasing variance in ore quality across mining sites.

Figure 3.26: Comparing G, M, and D policies on mining metrics with low, medium, and high variance in ore quality.



(a) 95% confidence intervals on percent of ore processed within ± 0.5% of ore quality target for 30 replications comparing three different dispatching policies for increasing variance in ore quality across mining sites.



(b) 95% confidence intervals on percent of ore processed within ± 1.0% of ore quality target for 30 replications comparing three different dispatching policies for increasing variance in ore quality across mining sites.

Figure 3.27: Comparing G, M, and D policies on ore quality metrics with low, medium, and high variance in ore quality.

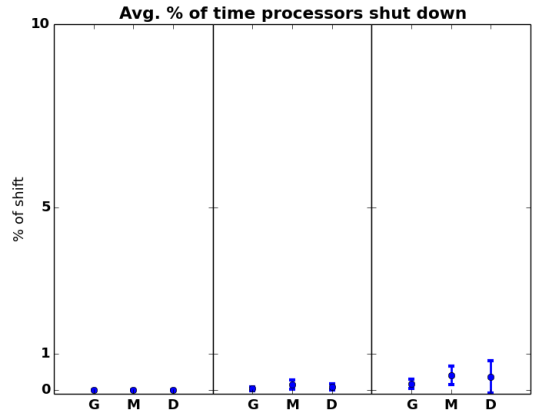
Varying the travel times

We next test each dispatching policy on instances of an open-pit mine with increasing variance in the travel times between pairs of locations. In the base instance, the travel times are randomly generated from Johnson SU distributions. In the second instance, we keep the mean travel time the same between each pair of sites, but with probability $p = 0.2$, we double the travel time. In the third instance, we again keep the mean travel time fixed between each pair of locations but with probability $p = 0.4$, we double the travel time. Figures (3.28a)-(3.30b) display the output for the policy comparison on these instances.

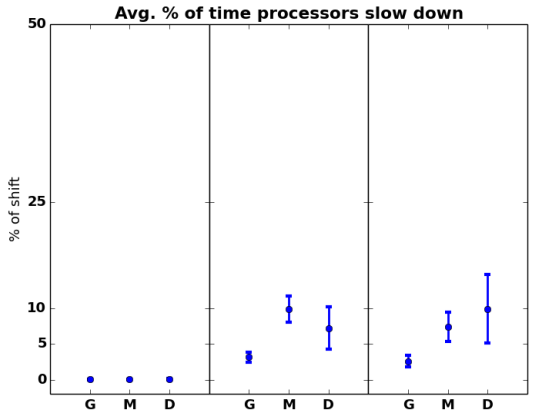
We first observe from Figures (3.28a) - (3.28c) that all three policies must slow and shut down the processors slightly more often when there is higher variance in travel times. However, the greedy target-matching policy performs the best out of the three policies. The MIP-based target-matching and discrete-time MIP policies are similar in all three instances.

On the mining metrics (Figures (3.23a) - (3.23b)), the discrete-time MIP policy undermines significantly less than the MIP-based target matching policy, which in turn undermines less than the greedy target matching policy for all instances. The discrete-time MIP policy also over-mines significantly less than the other two policies on all instances. There is a noticeable increase in the maximum shortfall for both the MIP-based target-matching and discrete-time MIP policies when travel time variance increases. All three policies over-mine less when travel time variance is high. We note that there is an initial change in mining metrics when the travel time variance is increased from the base instance, but further increasing the variance does not produce another similar change.

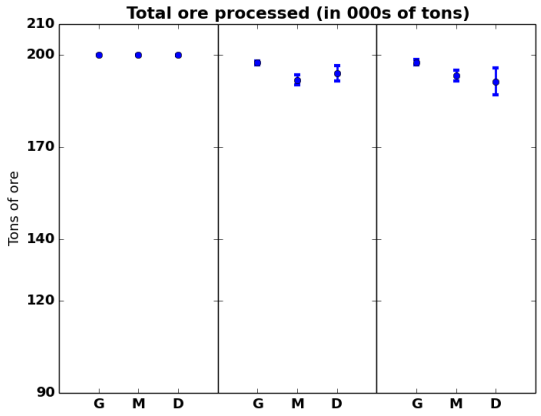
Finally, on the ore quality metrics (Figures (3.24a) - (3.24b)), we observe that the discrete-time MIP policy has the highest ore quality of the three policies for all instances. The MIP-based target matching policy has slightly higher ore quality than the greedy target-matching policy on all instances. There is a slight decrease in the ore quality as the variance increases for all three policies. Again, we observe that there is an initial decrease in ore quality when the travel time variance is increased from the base instance, but further



(a) 95% confidence intervals on fraction of time processors shut down for 30 replications comparing three different dispatching policies for increasing variance in travel times.



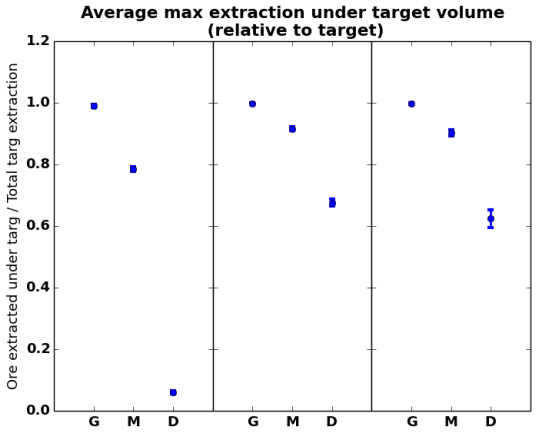
(b) 95% confidence intervals on fraction of time processors slowed down for 30 replications comparing three different dispatching policies for increasing variance in travel times.



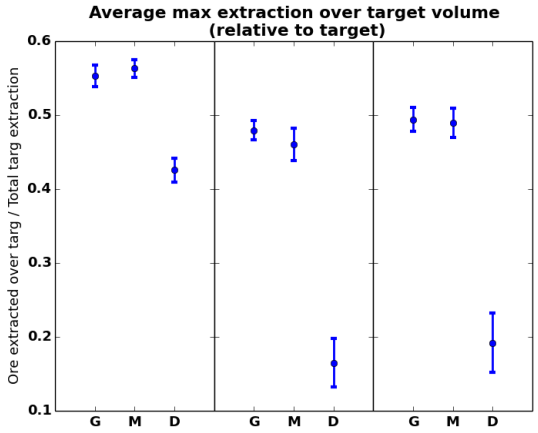
(c) 95% confidence intervals on total ore processed for 30 replications comparing three different dispatching policies for increasing variance in travel times.

Figure 3.28: Comparing G, M, and D policies on processing metrics with low, medium, and high variance in travel times.

increasing the variance does not produce another similar change.

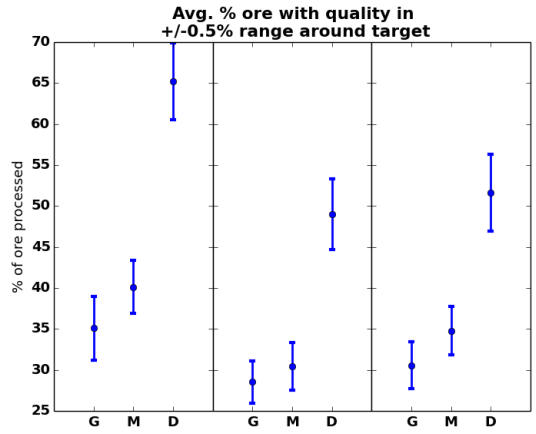


(a) 95% confidence intervals on maximum mining target shortfall across all mining sites (relative to target extraction) for 30 replications comparing three different dispatching policies for increasing variance in travel times.

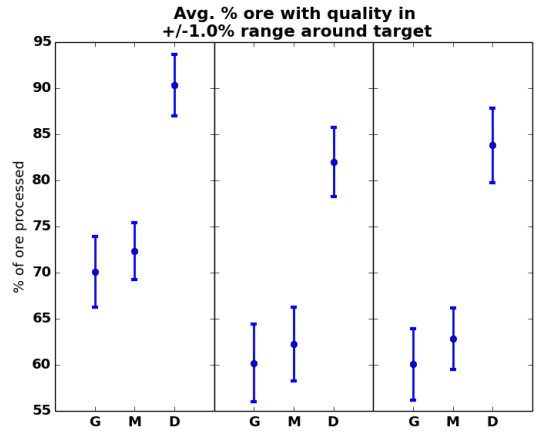


(b) 95% confidence intervals on maximum excess of mining targets across all mining sites (relative to target extraction) for 30 replications comparing three different dispatching policies for increasing variance in travel times.

Figure 3.29: Comparing G, M, and D policies on mining metrics with low, medium, and high variance in travel times.



(a) 95% confidence intervals on percent of ore processed within ± 0.5% of ore quality target for 30 replications comparing three different dispatching policies for increasing variance in travel times.



(b) 95% confidence intervals on percent of ore processed within ± 1.0% of ore quality target for 30 replications comparing three different dispatching policies for increasing variance in travel times.

Figure 3.30: Comparing G, M, and D policies on ore quality metrics with low, medium, and high variance in travel times.

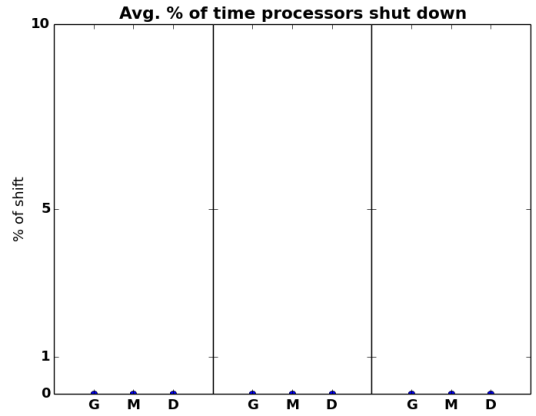
Varying the distance between mining sites and processing sites

Finally, we test each dispatching policy on instances of an open-pit mine with increasing variance in the distances between the mining sites and the processing sites. In the base instance, the mean travel times are determined from Johnson SU distributions. We use these values to calculate the average of the mean travel times between the mining and processing sites. In the second instance, we move two ore mining and two waste mining sites 1.5 times farther away from the processing sites. We move the rest of the mining sites close enough to the processors that the average travel time between all the mining sites and the processing sties is the same as in the base instance. In the third instance, we again keep the average travel time the same but move two ore mining and two waste mining sites 2.0 times farther away. Figures (3.31a)-(3.33b) display the output for the policy comparison on these instances.

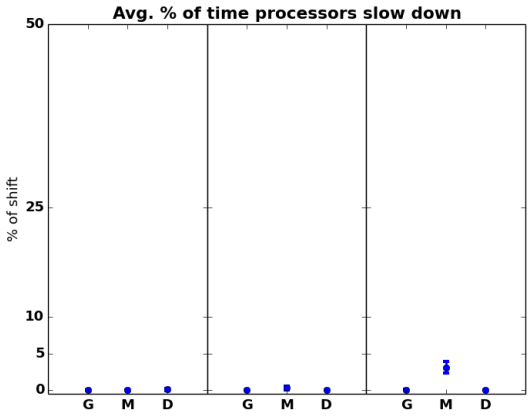
We first observe from Figures (3.28a) - (3.28c) that all three policies avoid shutting down the processors. However, the MIP-based target-matching policy slows the processors significantly more than the other two policies as the geographical variance increases. Both the greedy and discrete-time MIP policies meet processing targets in all three instances.

On the mining metrics (Figures (3.23a) - (3.23b)), the discrete-time MIP policy under-mines significantly less than the MIP-based target matching policy, which in turn under-mines less than the greedy target matching policy for all instances. The discrete-time MIP policy also over-mines significantly less than the other two policies on all instances. For the discrete-time MIP policy, there is a slight increase in shortfall as the variance increases, whereas the greedy target-matching policy improves as the variance increases. In addition, the discrete-time MIP policy over-mines slightly less as variance increases.

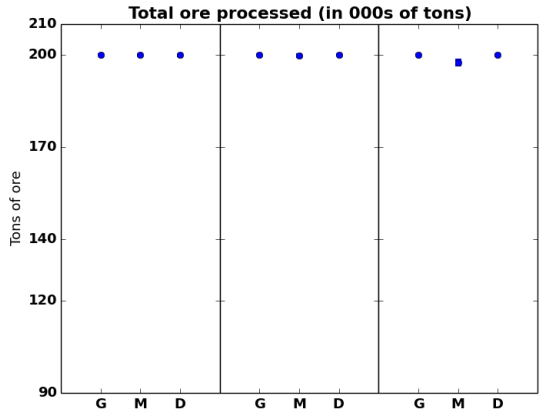
Finally, on the ore quality metrics (Figures (3.24a) - (3.24b)), we observe that the discrete-time MIP policy has the highest ore quality of the three policies for all instances. The MIP-based target matching policy is indistinguishable from the greedy target-matching policy on all instances. Overall, there is almost no difference in ore quality for any choice



(a) 95% confidence intervals on fraction of time processors shut down for 30 replications comparing three different dispatching policies for increasing variance in distances between mines and processing sites.



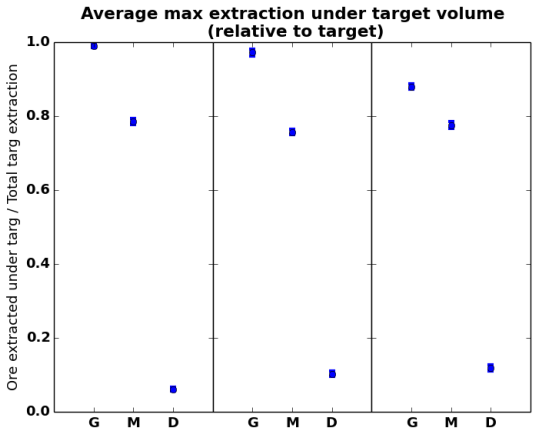
(b) 95% confidence intervals on fraction of time processors slowed down for 30 replications comparing three different dispatching policies for increasing variance in distances between mines and processing sites.



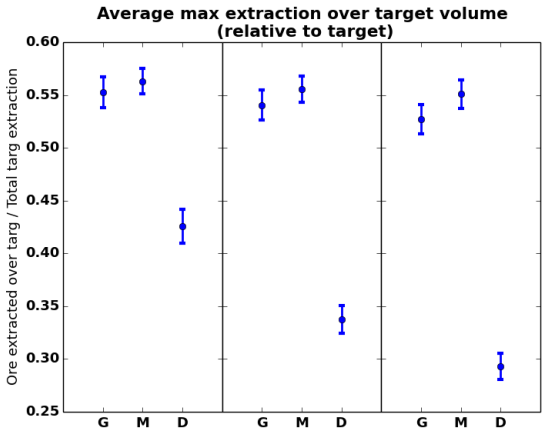
(c) 95% confidence intervals on total ore processed for 30 replications comparing three different dispatching policies for increasing variance in distances between mines and processing sites.

Figure 3.31: Comparing G, M, and D policies on processing metrics with low, medium, and high variance in distances between mines and processing sites.

of geographical variability.

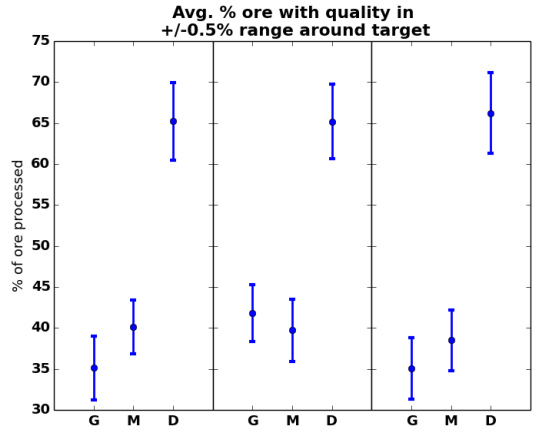


(a) 95% confidence intervals on maximum mining target shortfall across all mining sites (relative to target extraction) for 30 replications comparing three different dispatching policies for increasing variance in distances between mines and processing sites.

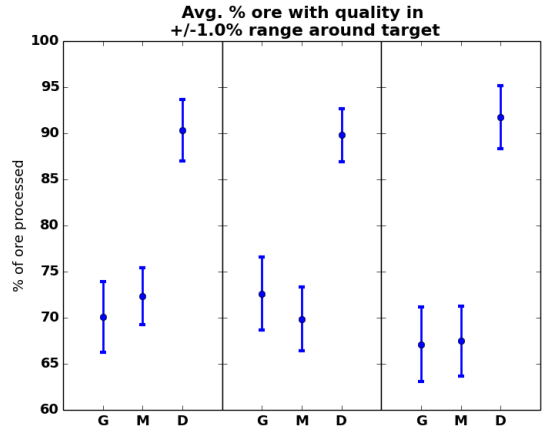


(b) 95% confidence intervals on maximum excess of mining targets across all mining sites (relative to target extraction) for 30 replications comparing three different dispatching policies for increasing variance in distances between mines and processing sites.

Figure 3.32: Comparing G, M, and D policies on mining metrics with low, medium, and high variance in distances between mines and processing sites.



(a) 95% confidence intervals on percent of ore processed within ± 0.5% of ore quality target for 30 replications comparing three different dispatching policies for increasing variance in distances between mines and processing sites.



(b) 95% confidence intervals on percent of ore processed within ± 1.0% of ore quality target for 30 replications comparing three different dispatching policies for increasing variance in distances between mines and processing sites.

Figure 3.33: Comparing G, M, and D policies on ore quality metrics with low, medium, and high variance in distances between mines and processing sites.

The results in this section allow us to draw some conclusions about the three dispatching policies. First, the discrete-time MIP dispatching policy consistently outperforms both the greedy and MIP-based target matching policies on almost every metric in every instance. The greedy and MIP-based target-matching policies are difficult to differentiate, although the MIP-based policy tends to produce higher ore quality and lower maximum shortfall on mining targets. When the targets are easy to meet (e.g., when plenty of trucks are available), the discrete-time MIP policy does not outperform the other policies as much. Finally, we note that when the discrete-time MIP model can take advantage of the system information, it performs better. When the dispatching model does not have access to the information, such as the instances with high travel time variance, it does not perform as well. However, it still outperforms the other policies on two of the three key metrics. Thus, we conclude that the discrete-time MIP policy is a good dispatching policy to use for an open-pit mine with a wide range of characteristics.

3.5 Conclusion

We present three new truck dispatching policies for use in open pit mining. The first two policies are defined in two phases, the first being solving a nonlinear flow-rate model that accounts for queueing in a computationally tractable way and the second being a method of matching the immediate dispatch decision to the target flow rate. One option is a greedy target-matching policy, simply selecting the round-trip farthest from its target flow. Another option is to solve a small MIP that matches trucks becoming available in the near future to their possible round trips. The third policy is defined by the solution to a time-discretized MIP dispatching model that accounts for both the current state of the system and the effects of an immediate decision on the future state of the system. We demonstrate the relative effectiveness of each of the three policies by embedding them in a discrete-event simulation of an open pit mine. By altering characteristics of the mine, we

show that the discrete-time MIP-based policy consistently outperforms both the greedy and MIP-based target-matching policies. Although computationally much less complex, the two-phase polices are not as effective at meeting production targets in a dynamic mining environment.

Future work could investigate alterations to the nonlinear flow-rate model or target-matching policies to improve their performance. Another route for future research would be changing the objective priorities to see how the competing policies respond. Related to this, since the model of ore quality used in this work is a very rough approximation, future research could investigate how the solution quality changes as the change in ore quality over time is modeled to more closely match reality.

4 OPTIMIZING FLUX BOUND CHANGE DECISIONS IN METABOLIC ENGINEERING

4.1 Modifying Flux Bounds

Metabolic engineering is used to alter chemical processes in cellular organisms to optimize certain outputs produced by the cell. Typically, the goal is to cause the cell to produce a desired biochemical product, such as biofuels, through optimal design of the metabolic network (Zeikus (1980); Stephanopoulos et al. (1998)). The relationships between reactions can be represented by sets of steady-state stoichiometric or nonlinear kinetic equations.

There are two major challenges of metabolic network design. The first is that it is difficult to exactly implement metabolic network modifications. Due to the highly complex interconnected structure of the networks, missing a planned modification by even a small amount could significantly reduce the effectiveness of the change. Another difficulty is that the cell's response to alterations to the network is not known with complete certainty. Modifications that appear desirable might impact the cell in unintended ways, such as causing the cell's growth to be inhibited or even killing the cell. The challenge, therefore, is to select a subset of reactions to alter to encourage the cell to produce the desired output without negatively impacting the cell's functionality (Stephanopoulos et al. (1998)).

A common perspective of cellular dynamics is the S-System representation of the metabolic network, which uses non-linear kinetic approximations of fluxes based on power-law expressions (Ranganathan et al. (2010); Stephanopoulos et al. (1998)). Flux balance analysis (FBA) is another common perspective for network design and analysis. FBA is an optimization-based approach to modeling and optimizing the interactions between reactions through the use of stoichiometric equations and flux bounds (Ranganathan et al. (2010); Orth et al. (2010b)). Our work falls under the FBA perspective, so we focus our literature review on work related to FBA.

Existing Methods

Early work in solving metabolic engineering problems with optimization involves single-level models based on cellular kinetics. Specifically, several authors aim to model and design metabolic pathways using single-stage optimization models to maximize the concentration of a desired biochemical product (Voit (1992); Regan et al. (1993); Hatzimanikatis et al. (1996a,b); Conejeros and Vassiliadis (1998)). Petkov and Maranas (1997) model the metabolic engineering problem using the S-System representation, but consider the kinetic orders to be uncertain. They provide probability distributions on the kinetic orders and model the uncertainty with chance constraints.

Schuster and Heinrich (1991) and Torres (1994) consider alternative objectives to maximizing biochemical yield, such as maximization or minimization of intermediate metabolites and minimization of the transition time between reactions, respectively. Vera et al. (2003), Halsall-Whitney et al. (2003), and Villaverde et al. (2016) extend this idea to a multi-objective modeling approach. The authors use systematic approaches to find efficient frontiers for two different objectives: maximizing biochemical yield and minimizing intermediate concentrations or the total amount the fluxes are changed.

Clark and Westerberg (1983, 1990) and Clark (1990) present early work using bilevel programming (BLP) models in the context of chemical engineering. More recently, BLP models have been built using principles of flux balance analysis (FBA) rather than the S-System perspective. One FBA-based BLP model is the OptKnock model described by Burgard et al. (2003). The OptKnock model features binary variables that select reactions to remove from the network to maximize biochemical yield. The cellular behavior is modeled as assuming it has the objective to maximize biomass. Burgard and Maranas (2003) show that typically this is a good choice for modeling the cellular objective. Burgard et al. (2003) also show that if an alternative cellular objective is used, specifically minimization of metabolic adjustments (MOMA), the final metabolic outputs are very similar. They conclude that the model is robust with respect to the choice of cellular objective. The authors

solve the model directly with a commercial mixed-integer programming (MIP) solver. Pharkya et al. (2003) apply a modified version of OptKnock to amino acid overproduction to find alternate optimal solutions and also use a commercial MIP solver to solve their model.

OptStrain (Pharkya et al. (2004)) and SimOptStrain (Kim et al. (2011)) incorporate OptKnock into a more complex design strategy. First, reactions are chosen to be added to the network to maximize theoretical biochemical production. Then OptKnock is solved to select the genes to be deleted. OptStrain applies these techniques sequentially, whereas SimOptStrain applies them simultaneously. In addition to the SimOptStrain method, Kim et al. present an MIQCP method, BiMOMA, that augments OptKnock with the network design step when MOMA is used as the cellular objective. The authors use some pre-processing steps to reduce the size of the solution space and propose an iterative algorithm of increasing gene knockouts to solve their models. Another extension of OptKnock, RobustKnock (Tepper and Shlomi (2010)), maximizes the minimum biochemical production at the top level while maximizing biomass at the lower level. This approach avoids strategies that have extreme production rates when the desired product is strongly inversely correlated with the cellular objective. The authors solve the model directly with a commercial MIP solver. A related extension that also avoids extreme strategies is the addition of a penalty term to the level objective that limits the number of gene deletions selected (Feist et al. (2011)).

Pharkya and Maranas (2006) and Ranganathan et al. (2010) further extend OptKnock into OptReg and OptForce, respectively. These models are similar to OptKnock, but use binary variables to select a set of reactions to up- or down-regulate in addition to those that are removed. OptORF considers a different extension of OptKnock by adding gene-to-protein-to-reaction information to the constraints to model how the reaction deletions relate to gene modifications (Kim and Reed (2010)).

A recent BLP metabolic engineering model is the CosMos model proposed by Cotten

and Reed (2013). This model builds on the OptKnock, OptReg, and OptForce models by allowing continuous modifications of the upper and lower bounds on reactions, in addition to choosing a set of reactions to alter. The model is a BLP with the objective to maximize worst-case biochemical yield. In CosMos, the authors model the cellular behavior with the assumption that the cell is trying to minimize biochemical yield. By modeling cellular behavior in this way, the authors obtain a lower bound on the achievable yield. To solve the model, the authors use bound-strengthening techniques and solve the resulting problem directly with a commercial MIP solver. Cotten and Reed (2013) also explore the uncertainty associated with the bound changes. Specifically, the authors obtain a solution to CosMos and generate Uniform(0,1) random variables to represent the percent of the planned bound change that is actually achieved. The fractional bound changes are then passed to the lower-level problem to find the resulting flux values. The authors demonstrate that some strategies that produce the same deterministic solution have significantly different responses to the uncertainty; some solutions are not very sensitive and produce high biochemical yields in spite of the fractional bound change, whereas some perform very poorly when only a fraction of the bound change is achieved. Thus, accounting for the uncertainty in the achievable bound change could cause one strategy to be preferred over another strategy that appears identical in terms of a deterministic estimate of yield.

Contributions

We present several new BLP models based on the CosMos model that include some additional features. We are interested in finding solutions in which the lower-level problem will not become infeasible when there is uncertainty in the true bounds achieved on the flux values. Existing work focuses on finding such solutions by limiting the number of bounds altered and discovering alternative optimal solutions that are less sensitive to the uncertainty in the achievable bound changes. However, existing models such as CosMos

can still produce solutions that are sensitive to the implementation uncertainty, as the decision maker changes the flux bounds to increase yield as much as possible, potentially constraining the lower-level problem so much that small deviations lead to infeasibility. One way to increase the likelihood of a feasible lower-level problem is to ensure a high rate of cellular growth. Our hypothesis is that a solution that produces simultaneously high yield and growth is likely to be less susceptible to small perturbations in bound changes. Toward that end, we first present a variation of the CosMos model in which we maximize an approximation of the worst-case productivity, defined here as cellular growth (biomass) $\times$ biochemical yield. This variation produces solutions that are less sensitive than solutions to the base CosMos model, but that are still sensitive to implementation uncertainty.

The second model is a bi-objective extension to the CosMos model (BiCosMos) with two lower level problems. BiCosMos encourages solutions that are less sensitive to implementation uncertainty with two top-level objectives: maximizing the worst-case yield and maximizing the best-case biomass. The lower-level problems – a minimum yield problem and a maximum growth problem – provide two alternative flux solutions: one that gives a lower bound on the achievable yield and one that guarantees the existence of a solution with positive growth. As part of our computational results, we show that this model can be solved to obtain a Pareto frontier of solutions that trade-off between biochemical production and cellular growth. We thus attempt to avoid strategies if they produce high yield but are almost infeasible for the lower-level problem. An additional advantage of our bi-objective model is that it gives a set of alternative solutions with different characteristics, rather than a single solution.

The final models are stochastic extensions of maximum productivity and BiCosMos models. Although the success of flux bound changes is uncertain, the uncertainty is not commonly incorporated into FBA models. We introduce the two-stage stochastic maximum productivity and BiCosMos models to account for this uncertainty as part of the optimization model. The success of bound changes depends on how the enzymes

used affect the fluxes, so we sample the fractional success of each proposed bound change from a probability distribution to obtain a list of scenarios. The direct stochastic extension of BiCosMos tends to produce solutions that have high expected worst-case yield and expected best-case growth but in which, for each scenario, only yield or growth are positive and rarely are both simultaneously positive. We thus propose a variation in which we solve two auxiliary stochastic problems: one in which we maximize both approximate productivity (best-case growth $\times$ worst-case yield) and best-case growth, and the other in which we maximize both approximate productivity and worst-case yield. To the best of our knowledge, these are the first stochastic bilevel FBA models.

While the maximum productivity and BiCosMos models can be solved directly using commercial MIP solvers, the stochastic BiCosMos model is a very large MIP with large bounds on many variables and therefore intractable with commercial MIP solvers. Recent work in stochastic programming methods suggests use of a modified progressive hedging algorithm to solve large stochastic MIPs. However, our stochastic BiCosMos model has a lower bound constraint on either expected growth or expected yield, preventing us from fully decomposing the model by scenario. Thus, we conclude with a description of how progressive hedging can be combined with dynamic allocation of budgets across scenarios in order to fully decompose the model. We demonstrate that incorporating the uncertainty directly into the model results in less sensitive solutions and higher expected yield after the implementation uncertainty is observed.

In Section 4.2, we present the full CosMos model as described by Cotten and Reed (2013). In Section 4.3, we alter the objective of CosMos to maximize a worst-case approximation of productivity. In Section 4.4, we extend CosMos to BiCosMos by introducing the additional objective of maximizing best-case cellular growth. We include computational results showing the effect of each model on yield and productivity in the face of implementation uncertainty. The stochastic extensions of the maximum productivity and BiCosMos models are described in Section 4.5. In Section 4.6, we describe our implementation of

progressive hedging with dynamic budget allocation. Finally, in Section 4.7, we present a computational study and evaluation of the solutions obtained from each new model and results demonstrating the effectiveness of including uncertainty in the model directly. Possible future research directions are discussed in Section 4.8.

4.2 The CosMos Model

We begin by presenting the full formulation of the original CosMos model of continuous flux bound modifications. CosMos is a bilevel program, which gives it a specific structure. First, one decision maker (the "top-level" decision maker) selects a solution (in this case, which bounds to change and how much to change them). A second decision maker ("lower-level" decision maker) then has a set of feasible decisions that could depend on the solution to the top-level problem. The second decision maker chooses a solution in response to the first decision maker's solution (in this case, the resulting flux values).

Given a set of reactions J and a set of metabolites M, the decisions that must be made are determining which lower bounds to alter ($\delta_j^\alpha,\ \forall j \in J$), which upper bounds to alter ($\delta_j^\beta,\ \forall j \in J$), and the selected lower and upper bound-change values ($\alpha_j, \beta_j\ \ \forall j \in J$, respectively). The δ^α variables are binary indicator variables such that for a reaction $j \in J$, $\delta_j^\alpha = 1$ if we change lower bound j and 0 otherwise. In the same way, the δ^β variables are binary indicator variables for the upper bound changes. The lower level decision variables are the flux values ($v_j,\ \forall j \in J$) chosen by the cell in response to the bound changes.

Every reaction $j \in J$ has a given lower bound of LB_j and a given upper bound UB_j. In many cases one bound is 0 and the other is large enough (or small enough) that the flux is effectively unconstrained on one side. There is also a matrix S, containing the stoichiometric coefficients S_{mj} for each $m \in M$ and $j \in J$. The decision maker's objective is to maximize the worst-case biochemical yield, given by a flux v_y for a specific reaction $y \in J$, while limiting the total number of bounds changed.

Each potential induced lower and upper bound is constrained by given bounds LB_j and UB_j, respectively:

$$LB_j\delta_j^\alpha \leq \alpha_j \leq UB_j\delta_j^\alpha \ \forall j \in J \tag{4.1}$$

$$LB_j\delta_j^\beta \leq \beta_j \leq UB_j\delta_j^\beta \ \forall j \in J. \tag{4.2}$$

In addition, the bound selection variables must be binary:

$$\delta_j^\alpha, \delta_j^\beta \in \{0,1\} \qquad \forall j \in J. \tag{4.3}$$

A small difference from the original CosMos model is the addition of a constraint on the number of bounds that can be changed. In the original CosMos model, there is a penalty in the objective function that prevents the decision-maker from changing a large number of bounds. In this formulation, we choose an integer K as an upper bound on how many bounds the decision maker can change:

$$\sum_{j\in J}(\delta_j^\alpha + \delta_j^\beta) \leq K. \tag{4.4}$$

The final constraint on the problem is that the set of fluxes v is a solution to the following lower-level optimization problem, in which cellular behavior is modeled as minimizing the desired biochemical yield v_y:

$$\min_v \ v_y$$

$$\text{s.t.} \sum_{j\in J} S_{mj}v_j = 0 \qquad \forall m \in M \tag{4.5}$$

$$(1-\delta_j^\alpha)LB_j + \alpha_j \leq v_j \leq (1-\delta_j^\beta)UB_j + \beta_j \qquad \forall j \in J. \tag{4.6}$$

Equations (4.5) are the stoichiometric equations for the cell. Constraints (4.6) are the bounds

imposed on the fluxes by the top-level decision maker or the original upper and lower bounds if no change is made. We note that this model of flux bounds differs slightly from the original CosMos model. In the original CosMos model, the β variables correspond to the extent to which the upper bound is decreased, whereas in this formulation the β variables correspond to the value of the upper bound induced on each reaction, if any. Similarly, in this formulation, the α variables represent the induced lower bounds on each reaction, if any. In both cases, $\delta_j^\beta = 0 \implies \beta_j = 0$ for all $j \in J$. The same holds for the lower bounds, α. This modeling choice is equivalent to that of CosMos. Although the values α and β take in a solution to this formulation of CosMos will be different, the resulting flux values are unaffected. We define the lower-level feasible set as

$$X(\alpha, \beta, \delta^\alpha, \delta^\beta) := \{v \in \mathbb{R}^{|J|} : v \text{ satisfies } (4.5) - (4.6)\}.$$

Cotten and Reed replace the lower-level problem by the necessary and sufficient optimality conditions from linear programming duality theory. The dual of the lower-level problem is

$$\max_{u,z,d} \sum_{j \in J}((1-\delta_j^\alpha)LB_j + \alpha_j)u_j + \sum_{j \in J}((1-\delta_j^\beta)UB_j + \beta_j)z_j$$

$$\text{s.t. } u_j + z_j + \sum_{m \in M} S_{mj}d_m = I_{j=y} \qquad \forall j \in J \qquad (4.7)$$

$$u_j \geq 0 \qquad \forall j \in J \qquad (4.8)$$

$$z_j \leq 0 \qquad \forall j \in J. \qquad (4.9)$$

$I_{j=y}$ is an indicator function that is equal to 1 when $j = y$ and 0 otherwise. We define the lower-level dual feasible set as

$$\Pi := \{(u, z, d) : (u, z, d) \text{ satisfies } (4.7) - (4.9)\}.$$

A solution (v, u, z, d) is primal and dual optimal if and only if v is feasible to the primal problem, (u, z, d) is feasible to the dual problem, and

$$u_j(v_j - (1 - \delta_j^\alpha)LB_j + \alpha_j) = 0 \qquad \forall j \in J \tag{4.10}$$

$$z_j((1 - \delta_j^\beta)UB_j + \beta_j - v_j) = 0 \qquad \forall j \in J. \tag{4.11}$$

Thus, the authors reformulate CosMos as a single-level bilinear model:

$$\begin{aligned} &\max_{\substack{\alpha,\beta,\delta^\alpha,\\ \delta^\beta,v,u\\ z,d}} \ v_y \\ &\quad \text{s.t. } (\alpha, \beta, \delta^\alpha, \delta^\beta) \text{ satisfy } (4.1) - (4.4) \\ &\qquad v \in X(\alpha, \beta, \delta^\alpha, \delta^\beta);\ (u, z, d) \in \Pi \\ &\qquad (u, z, v, \alpha, \beta, \delta^\alpha, \delta^\beta) \text{ satisfy } (4.10) \text{ and } (4.11). \end{aligned}$$

This model can be reformulated as a mixed-integer linear program (MILP) by introducing binary variables that correspond to the term that equals 0. In particular, it is possible to introduce a variable δ_j^u for all $j \in J$ such that if $\delta_j^u = 1$, then $v_j - (1 - \delta_j^\alpha)LB_j + \alpha_j = 0$, but if $\delta_j^u = 0$, then $u_j = 0$. A similar reformulation can be done with the second set of bilinear terms with a binary variable δ_j^z for all $j \in J$.

To formulate the optimality conditions as a MILP, there should be bound on the variables u and z, as well as the terms $(v_j - (1 - \delta_j^\alpha)LB_j + \alpha_j)$ and $((1 - \delta_j^\beta)UB_j + \beta_j - v_j)$. Since the greatest value v_j can take is UB_j, the greatest value $(v_j - (1 - \delta_j^\alpha)LB_j + \alpha_j)$ can take is $UB_j - LB_j$. The value $((1 - \delta_j^\beta)UB_j + \beta_j - v_j)$ is similarly bounded above by $UB_j - LB_j$. Although there are no natural bounds on the u_j or z_j variables, the authors suggest use of large artificial bounds of $U_j^{\max}$ and $Z_j^{\max}$ for each $j \in J$, respectively. By introducing artificial bounds on these variables, it is possible that some feasible solutions are cut off. However, as long as the bounds are sufficiently large, the optimal solution should remain

feasible. The CosMos model can then be written as a single-level MILP:

$$
\begin{aligned}
\max_{\substack{\alpha,\beta,\delta^\alpha,\\ \delta^\beta,v,u\\ z,d}} \; & v_y \\
\text{s.t. } & (\alpha,\beta,\delta^\alpha,\delta^\beta) \text{ satisfy } (4.1)-(4.4) \\
& v \in X(\alpha,\beta,\delta^\alpha,\delta^\beta);\ (u,z,d) \in \Pi \\
& (v_j - (1-\delta_j^\alpha)LB_j + \alpha_j) \le (UB_j - LB_j)(1-\delta_j^u) && \forall j \in J \\
& ((1-\delta_j^\beta)UB_j + \beta_j - v_j) \le (UB_j - LB_j)(1-\delta_j^z) && \forall j \in J \\
& u_j \le U_j^{\max}\delta_j^u && \forall j \in J^- \cup J^R \\
& z_j \ge Z_j^{\max}\delta_j^z && \forall j \in J^+ \cup J^R \\
& \delta_j^u, \delta_j^z \in \{0,1\} && \forall j \in J.
\end{aligned}
$$

Cotten and Reed (2013) solve instances of CosMos with a commercial MIP solver.

4.3 MaxProd: A Maximum Productivity Extension of the CosMos Model

We next discuss a modification to the objective of the CosMos model. The motivation for this change in objective is the observation that the lower-level problem is nearly infeasible in a solution to CosMos, meaning a small perturbation in the bound changes could easily kill the cell, resulting in no yield. To prevent this behavior, we aim to avoid solutions in which the cell produces high yield but grows very little. One way to quantify this more desirable result is with cellular *productivity*. We approximate productivity as the product of yield and growth. By maximizing worst-case productivity instead of worst-case yield, we aim to encourage solutions with high yield while at the same time ensuring a higher likelihood of successful implementation. As with the CosMos model, a solution to this

model gives a guaranteed lower bound on the achievable productivity. We represent the growth as the flux of a given reaction $g \in J$.

The complete bilevel Max Productivity (MaxProd) model is as follows:

$$\begin{aligned} &\max_{\substack{\alpha,\beta,\delta^\alpha,\\ \delta^\beta,v,u\\ z,d}} \; v_y v_g && \text{(MaxProd)}\\ &\quad \text{s.t. } (\alpha,\beta,\delta^\alpha,\delta^\beta) \text{ satisfy } (4.1)-(4.4)\\ &\qquad v \in \arg\min\{v'_y v'_g : v' \in X(\alpha,\beta,\delta^\alpha,\delta^\beta)\} \quad . \end{aligned}$$

As written, MaxProd is a nonlinear model. However, the nonlinear objective term $v_y v_g$ can be moved to the constraints by adding a new variable t and adding the constraints

$$v_y v_g \geq t^2 \tag{4.12}$$

$$t \geq 0. \tag{4.13}$$

Then the objective can be changed to maximizing t. If $t \geq 0$, the constraint (4.12) is a rotated second-order cone constraint, which can be handled efficiently by modern commercial MIP solvers.

Unlike the top-level objective, the lower-level objective is a nonconvex function that cannot be reformulated with convex constraints. However, it is still possible to reformulate this model as a single-level mixed-integer second-order-cone program (MISOCP) by replacing the embedded minimum productivity problem with its optimality conditions. Because the objective function is nonconvex, the optimality conditions only guarantee a local optimal solution (rather than a global optimal solution). For the lower-level minimum productivity problem, the Karush-Kuhn-Tucker (KKT) conditions for local optimality are:

- primal feasibility:

$$v \in X(\alpha, \beta, \delta^\alpha, \delta^\beta);$$

- stationarity:

$$\begin{aligned} u_y + z_y + \sum_{m \in M} S_{my} d_m &= v_g \\ u_g + z_g + \sum_{m \in M} S_{mg} d_m &= v_y \\ u_j + z_j + \sum_{m \in M} S_{mj} d_m &= 0 \qquad\qquad \forall j \in J \setminus \{y, g\}; \end{aligned}$$

- dual feasibility:

$$\begin{aligned} u_j &\geq 0 \qquad\qquad \forall j \in J \\ z_j &\leq 0 \qquad\qquad \forall j \in J; \end{aligned}$$

- and complementary slackness:

$$\begin{aligned} u_j(v_j - (1 - \delta_j^\alpha) LB_j + \alpha_j) &= 0 \qquad\qquad \forall j \in J \\ z_j((1 - \delta_j^\beta) UB_j + \beta_j - v_j) &= 0 \qquad\qquad \forall j \in J. \end{aligned}$$

We note that all constraints are linear except for the two sets of bilinear complementary slackness constraints. However, since these are identical to the complementary slackness conditions for the CosMos model, we use the same reformulation technique to write

MaxProd as a single level MISOCP:

$$
\begin{aligned}
\max_{\substack{\alpha,\beta,\delta^\alpha,\\ \delta^\beta,v,u\\ z,d}} \; & t \\
\text{s.t. } & (\alpha,\beta,\delta^\alpha,\delta^\beta) \text{ satisfy } (4.1)-(4.4) \\
& v \in X(\alpha,\beta,\delta^\alpha,\delta^\beta) \\
& v_y v_g \geq t^2 \\
& v_g + u_y + z_y + \sum_{m\in M} S_{my} d_m = v_y \\
& v_y + u_g + z_g + \sum_{m\in M} S_{mg} d_m = v_g \\
& u_j + z_j + \sum_{m\in M} S_{mj} d_m = v_j && \forall j \in J \setminus \{y,g\} \\
& u_j \geq 0 \; \forall j \in J;\; z_j \leq 0 \; \forall j \in J \\
& (v_j - (1-\delta_j^\alpha) LB_j + \alpha_j) \leq (UB_j - LB_j)(1-\delta_j^u) && \forall j \in J \\
& ((1-\delta_j^\beta) UB_j + \beta_j - v_j) \leq (UB_j - LB_j)((1-\delta_j^z) && \forall j \in J \\
& u_j \leq U_j^{\max} \delta_j^u && \forall j \in J \\
& z_j \geq Z_j^{\max} \delta_j^z && \forall j \in J \\
& \delta_j^u, \delta_j^z \in \{0,1\} && \forall j \in J.
\end{aligned}
$$

In this way, we aim to find solutions that result in high worst-case productivity with a formulation that has similar computational complexity as that of the original CosMos model.

4.4 BiCosMos: A Bi-Objective Extension of the CosMos Model

We next propose a bi-objective extension of the CosMos model (BiCosMos) that incorporates a measure of cellular viability into the top-level decision problem. The motivation for such a model is an observation that the CosMos model produces solutions that can be sensitive to implementation uncertainty. At the end of this section, we demonstrate that the MaxProd model still produces solutions that are sensitive to the uncertainty and provides a conservative estimate of the possible yield. With the BiCosMos model, we attempt to combine the guaranteed lower bound on yield with a proxy for maximizing likelihood of success of the bound implementation (high cellular growth). Another benefit of a bi-objective model is that it gives decision makers some control over how much they emphasize high yield versus high growth.

To encourage less sensitive solutions and give the decision maker more control of the solution, we propose the inclusion of an additional top-level objective: maximizing the best-case growth. This requires the introduction of an additional embedded optimization problem, in which we model the cellular objective as maximizing growth. Our choice of the embedded maximum growth lower-level problem is motivated by the use of this objective in OptKnock (Burgard et al. (2003)) and many of its extensions. The new objective forces the top-level decision maker to change flux bounds in such a way that there exists at least one solution to the lower-level problem in which the cell has high biomass. Because the top-level decision maker is also maximizing the worst-case yield, the minimum yield lower-level problem will still provide a guaranteed lower bound on the achievable yield. The two resulting sets of flux values give two possible cellular responses. Because the true cellular response is not known with complete certainty, it is reasonable and, perhaps, beneficial to have a model that considers more than one possibility for how the cell might respond to bound changes. Thus, for every top-level strategy, BiCosMos produces two

possible cellular responses to the bound changes, one with a guaranteed lower bound on yield, and one that assures existence of a solution with good cellular growth.

To model BiCosMos, we use the same binary indicator variables δ^α (δ^β) as in CosMos to select a set of bounds to change, where for reaction $j \in J$, $\delta^\alpha_j = 1$ if we induce a lower bound on j and 0 otherwise. Similarly, $\delta^\beta_j = 1$ if we induce an upper bound on j and 0 otherwise. The new lower bound on reaction j is given by α_j and the new upper bound is given by β_j. The lower-level decision variables in the minimum yield problem are the flux values, (v_j, $\forall j \in J$) chosen by the cell in response to the top-level decisions. For the maximum growth problem, the decision variables are an alternative set of flux values (w_j, $\forall j \in J$) chosen by the cell in response to the top-level decisions. The new top-level objective is to maximize both the worst-case biochemical yield, v_y, and the best-case cellular growth, w_g.

We find solutions on the Pareto frontier of this bi-objective problem by maximizing one of the objectives and constraining the other to be at least ℓ:

$$\max v_y$$

$$w_{\mathrm{g}} \geq \ell.$$

The parameter ℓ can be varied between the minimum and maximum values w_g can take in a feasible solution to CosMos ($w \in X(\alpha, \beta, \delta^\alpha, \delta^\beta)$ for any $(\alpha, \beta, \delta^\alpha, \delta^\beta_j)$ satisfying (4.1)-(4.4)) to find points along the Pareto frontier of these objectives. Note that the role of the two objectives could be swapped, though we do not explore that here. We present Pareto optimal solutions for several instances of BiCosMos at the end of this section.

As in CosMos, v must be a solution to the minimum yield problem. Now we also require w to be a solution to the maximum growth problem for a given top-level solution (α, β):

$$\max_{w} w_g$$
$$\text{s.t. } w \in X(\alpha, \beta, \delta^\alpha, \delta^\beta). \tag{4.14}$$

We note that since the top level and lower level growth problem both have the objective to maximize cellular growth, we are able to replace the full maximum growth problem with just its constraints. Because of the agreement of the top and lower level objectives, the full bilevel formulation of BiCosMos can be written as:

$$\max_{\substack{\delta^\alpha, \delta^\beta \\ \alpha, \beta \\ v, w}} v_y \tag{BCM}$$
$$\text{s.t. } w_{\text{g}} \geq \ell$$
$$w \in X(\alpha, \beta, \delta^\alpha, \delta^\beta)$$
$$(\delta^\beta, \delta^\alpha, \alpha, \beta) \text{ satisfy } (4.1) - (4.4)$$
$$v \in \arg\min_{v} \{v_y : v \in X(\alpha, \beta, \delta^\alpha, \delta^\beta)\}.$$

As described in Section 4.2, we replace the minimum yield problem with the linearized

necessary and sufficient optimality conditions to create a single-level MILP formulation:

$$
\begin{aligned}
&\max_{\substack{\alpha,\beta,\delta^{\alpha},\\ \delta^{\beta},v,w\\ u,z,d}} \; v_y \\
&\quad \text{s.t.} w_{\mathrm{g}} \geq \ell \\
&\qquad (\delta^{\beta}, \delta^{\alpha}, \alpha, \beta) \;\; \text{satisfy } (4.1) - (4.4) \\
&\qquad v \in X(\alpha, \beta, \delta^{\alpha}, \delta^{\beta});\; w \in X(\alpha, \beta, \delta^{\alpha}, \delta^{\beta}) \\
&\qquad (u, z, d) \in \Pi \\
&\qquad (v_j - (1 - \delta_j^{\alpha}) LB_j + \alpha_j) \leq (UB_j - LB_j)(1 - \delta_j^{u}) && \forall j \in J \\
&\qquad ((1 - \delta_j^{\beta}) UB_j + \beta_j - v_j) \leq (UB_j - LB_j)(1 - \delta_j^{z}) && \forall j \in J \\
&\qquad u_j \leq U_j^{\max} \delta_j^{u} && \forall j \in J \\
&\qquad z_j \geq Z_j^{\max} \delta_j^{z} && \forall j \in J \\
&\qquad \delta_j^{u}, \delta_j^{z} \in \{0, 1\} && \forall j \in J.
\end{aligned}
$$

We then solve this model with multiple feasible values of ℓ to obtain different candidate solutions along the Pareto frontier.

Computational Results for Deterministic Extensions of CosMos

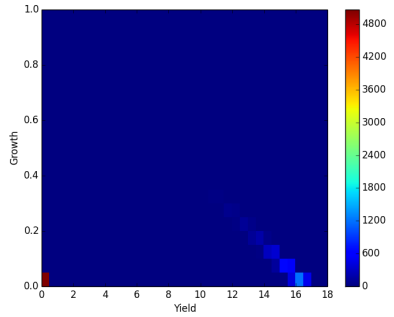
In our first computational study, we compare the effectiveness of our modeling choices in finding solutions that are robust to implementation uncertainty. We find solutions to instances of CosMos, MaxProd, and BiCosMos and evaluate metrics of interest to determine how sensitive each solution is.

We evaluate the effectiveness of CosMos, MaxProd, and BiCosMos by replicating the sampling procedure discussed in Cotten and Reed (2013) on solutions produced by each model. To generate solutions, we use data from *E. coli* cells. We present results for two reconstructions of the *E. coli* metabolic network: the core model with slightly fewer than

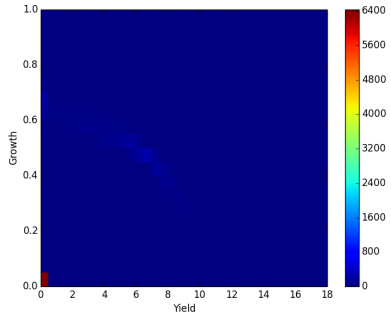
100 reactions (Orth et al. (2010a)) and the iJR904 reconstruction with approximately 1000 reactions (Reed et al. (2003)). We set the number of bound changes to be $k = 3$ for all models. After a solution to each model is obtained via the commercial MIP solver Gurobi, we generate 10,000 samples of the random variables representing the effectiveness of each bound change. This fraction could be > 1, representing a case where we overshoot the expected bound change. The details of the distribution and data used in these instances can be found in Appendix C. For each of the 10,000 scenarios, we solve the worst-case yield and best-case growth lower-level problems with the corresponding fractional bound change value and evaluate actual worst-case yield and actual best-case growth.

E. coli Core Model Reconstruction Results

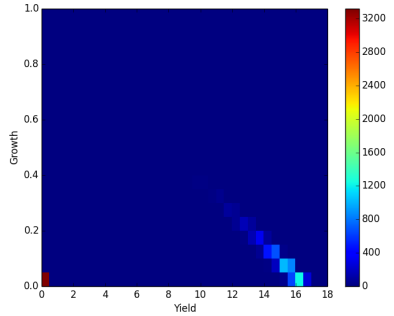
We begin with a discussion of the results obtained solving each model on the core model reconstruction of *E. coli*. In this reconstruction, the maximum possible yield is 16.819 and the maximum possible growth is 0.825. In Figure (4.1a) we plot a 2-dimensional histogram of yield versus growth for the sampled solution to the CosMos model. The histogram displays the relative frequency of samples that fall into discrete "boxes" of worst-case yield and best-case growth (e.g., yield between 0.5 and 1.0 and growth between 0.2 and 0.25). In these figures, a higher density of scenarios in a given box is represented by dark red and few or no scenarios in a given box is represented by dark blue. In Figure (4.1b), we repeat the procedure for MaxProd. Finally, in Figures (4.1c)-(4.1e) we evaluate the sampled yield and growth for three different solutions along the Pareto frontier of solutions to BiCosMos; in particular, we choose a solution with $w_g \geq 0.05$, a solution with $w_g \geq 0.35$, and a solution with $w_g \geq 0.75$.



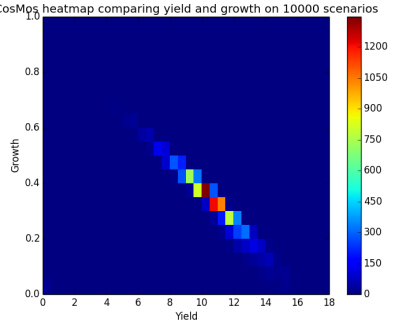
(a) Worst-case yield vs. best-case growth on a solution obtained from CosMos over 10000 scenarios.



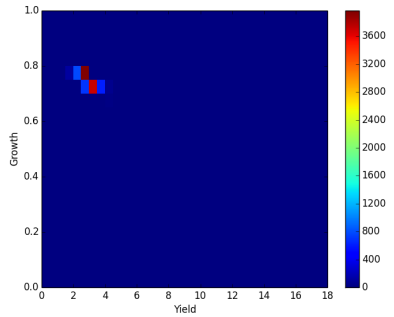
(b) Worst-case yield vs. best-case growth on a solution obtained from MaxProd over 10000 scenarios.



(c) Worst-case yield vs. best-case growth on a solution obtained from BiCosMos over 10000 scenarios. Lower bound on growth is 0.05.



(d) Worst-case yield vs. best-case growth on a solution obtained from BiCosMos over 10000 scenarios. Lower bound on growth is 0.35.



(e) Worst-case yield vs. best-case growth on a solution obtained from BiCosMos over 10000 scenarios. Lower bound on growth is 0.75.

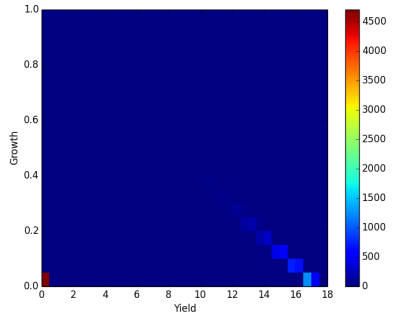
Figure 4.1: Worst-case yield versus best-case growth on 10,000 samples of bound implementation success on solutions to CosMos, MaxProd, and BiCosMos at three points along the Pareto frontier.

MaxProd produces a solution in which 8% of the 10,000 scenarios have yield at least 4.0 and growth at least 0.4. In the solution to CosMos, however, any scenario with positive yield

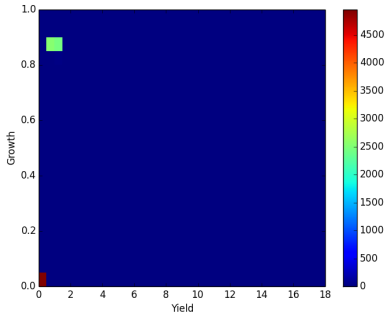
has growth less than 0.3. In both CosMos and MaxProd, there are a significant number of infeasible scenarios – almost 50% of the scenarios in CosMos and over 60% of the scenarios in MaxProd – where both yield and growth are 0. Overall, it appears that the deterministic MaxProd model makes little improvement over CosMos in terms of solution sensitivity. When the bound on w_g is small, BiCosMos produces solutions very similar to the solutions to CosMos and results in about 32% of the scenarios being infeasible. However, as we increase the bound to 0.35, we find no infeasible scenarios and a vast majority of scenarios with yield at least 9.0 and growth at least 0.3. When the bound is further increased to 0.75, again we find no infeasible solutions, but as we would expect the yield tends to be lower when the growth is higher. Only about 8% of the scenarios have yield at least 3.0, but all 10,000 scenarios have growth at least 0.6 and yield greater than 1.0.

E. coli iJR904 Reconstruction Results

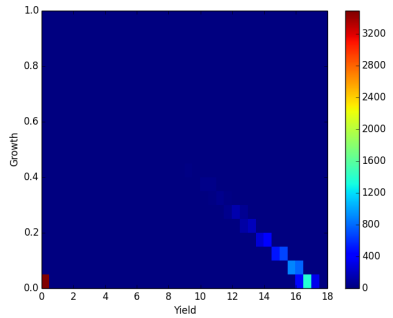
We next repeat the study with the iJR904 network reconstruction of *E. coli*. In this reconstruction, the maximum possible yield is 17.143 and the maximum possible growth is 0.921. In Figure (4.2a) we plot a 2-dimensional histogram of yield versus growth for the sampled solution to the CosMos model. In Figure (4.2b), we repeat the procedure for MaxProd. Finally, in Figures (4.2c)-(4.2e) we evaluate the sampled yield and growth for three different solutions along the Pareto frontier of solutions to BiCosMos. As with the core model reconstruction, we choose a solution with $w_g \geq 0.05$, a solution with $w_g \geq 0.35$, and a solution with $w_g \geq 0.75$.



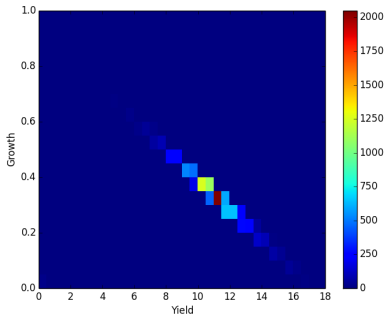
(a) Worst-case yield vs. best-case growth on a solution obtained from CosMos over 10000 scenarios.



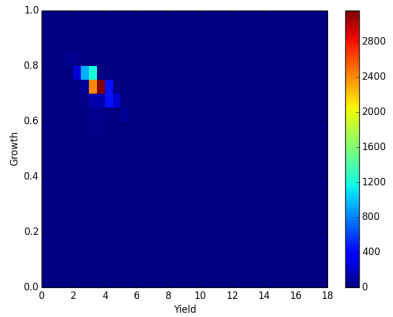
(b) Worst-case yield vs. best-case growth on a solution obtained from MaxProd over 10000 scenarios.



(c) Worst-case yield vs. best-case growth on a solution obtained from BiCosMos over 10000 scenarios. Lower bound on growth is 0.05.



(d) Worst-case yield vs. best-case growth on a solution obtained from BiCosMos over 10000 scenarios. Lower bound on growth is 0.35.



(e) Worst-case yield vs. best-case growth on a solution obtained from BiCosMos over 10000 scenarios. Lower bound on growth is 0.75.

Figure 4.2: Worst-case yield versus best-case growth on 10,000 samples of bound implementation success on solutions to CosMos, MaxProd, and BiCosMos at three points along the Pareto frontier.

We first note that the results demonstrate similar behavior to the results from the core model reconstruction. The one major difference is the MaxProd model, which now results

in approximately 50% of the scenarios having worst-case yield between 0.5 and 2.0 and best-case growth greater than 0.8. However, the other 50% of the scenarios are infeasible, similar to the results of CosMos. Once again, when the bound on w_g is small, BiCosMos produces solutions very similar to the solutions to CosMos and results in about 32% of the scenarios being infeasible, just as in the core model reconstruction results. The results for $w_g \geq 0.35$ and $w_g \geq 0.75$ in the iJR904 reconstruction are also very similar to those for the core model reconstruction.

Thus, from both network reconstructions of *E. coli*, we conclude that BiCosMos reduces the sensitivity of solutions to errors in implementation significantly over both CosMos and MaxProd. In addition, BiCosMos gives the decision maker more control over the solution by allowing them to choose how much to emphasize worst-case yield versus best-case growth. With certain choices of bounds on best-case growth, we observe significant increase in the number of scenarios with both positive growth and yield over both CosMos and MaxProd. When a solution to MaxProd is implemented, many scenarios are infeasible, similar to a solution to CosMos. However, when yield is positive, the solution to MaxProd also results in high best-case growth, increasing the likelihood that the bound changes will work as planned.

4.5 A Stochastic Programming Extension of CosMos

In this section, we discuss possibilities for incorporating implementation uncertainty directly into the optimization model. First, we examine a stochastic extension of MaxProd. Next, we argue that a direct stochastic extension of BiCosMos does not sufficiently capture the interaction between the two objectives on individual scenarios. We instead propose a variation of stochastic BiCosMos that involves solving two auxiliary problems that maximize expected approximate productivity to recover a subset of the Pareto optimal solutions that have more reasonable behavior on individual scenarios.

Uncertainty in CosMos

In the deterministic CosMos model and all of its extensions, we assume for every reaction $j \in J$ the values α_j and β_j are exactly implemented. Thus, we do not account for the uncertainty inherent in implementing the bound changes, which can result in solutions that are sensitive to that uncertainty. In this section, we explore extensions to the model in which we model implementation uncertainty using random variables. Specifically, rather than achieving a lower bound on reaction $j \in J$ of α_j, we achieve a fraction of that bound, $\xi_j \alpha_j$. Similarly, instead of an upper bound of β_j on reaction j, we have $\eta_j \beta_j$. We assume the uncertain coefficients ξ_j and η_j follow a known probability distribution. Note that it is possible to have $\xi_j > 1$ or $\eta_j > 1$. In the deterministic models, the decision maker would be indifferent between two strategies with the same yield even if one strategy has lower yield than the other when only a fractional bound change is achieved. With a stochastic model, we maximize the expected value of the objective and thus hope instead to find strategies that produce the maximum worst-case yield, worst-case productivity, or best-case growth *in expectation* with uncertain bound changes.

To build stochastic models for the extensions of CosMos, we use a technique called sample average approximation (SAA) to obtain a finite list of samples of ξ and η. It is well known that, in expectation, the optimal objective value of the SAA problem provides an upper bound on the objective value of the true stochastic program (Mak et al. (1999); Shapiro et al. (2009)). Furthermore, as the number of samples increases, the bound given by the SAA solution improves (Shapiro et al. (2009)). In addition, the solution obtained from the SAA problem, when evaluated under the true probability distribution, gives a lower bound on the optimal vale of the true stochastic programming problem. More details on SAA and modeling general stochastic programs are given by Mak et al. (1999) and Shapiro et al. (2009) and are summarized in Section 1.3. Given a probability distribution on the coefficients ξ_j and η_j, we sample a finite set of scenarios (possible realizations) $\{\xi^k, \eta^k\} \forall k \in K$. We assume each scenario occurs with equal probability $1/|K|$. For each

realization k and given top-level decisions $(\alpha, \beta, \delta^\alpha, \delta^\beta)$, the lower-level minimum yield problem is

$$\min_{v} v_y$$

$$\text{s.t.} \sum_{j \in J} S_{mj} v_j = 0 \qquad \forall m \in M \tag{4.15}$$

$$(1 - \delta_j^\alpha) LB_j + \xi_j^k \alpha_j \leq v_j \leq (1 - \delta_j^\beta) UB_j + \eta_j^k \beta_j \qquad \forall j \ \in J. \tag{4.16}$$

For each scenario $k \in K$, we define

$$X_k(\alpha, \beta, \delta^\alpha, \delta^\beta) := \{v : v \text{ satisfies } (4.15) - (4.16)\}.$$

A Stochastic Extension of MaxProd

We first propose a stochastic extension of MaxProd. By maximizing the expected value of worst-case productivity, we hope to find solutions that incorporate implementation uncertainty while still producing nonzero yield and growth simultaneously. It is possible to build a deterministic equivalent of a stochastic model with a finite scenario list called the *extensive form*. The extensive form model is based on the fact that some decisions must be made before uncertainty is realized (first-stage decisions), and some can be made after the uncertainty is realized (second-stage decisions). By creating a copy of each second-stage variable corresponding to a scenario $k \in K$, the bilevel formulation of the stochastic MaxProd model can be written in extensive form as:

$$\max_{\delta^\alpha, \delta^\beta, \alpha, \beta, v^s} \frac{1}{|K|} \sum_{k \in K} v_y^k v_g^k$$

$$\text{s.t. } (\delta^\beta, \delta^\alpha, \alpha, \beta) \text{ satisfy } (4.1) - (4.4)$$

$$v^k \in \arg\min\{v : v \in X_k(\alpha, \beta, \delta^\alpha, \delta^\beta)\} \ \forall k \in K.$$

This model can be reformulated as a single-level MISOCP using the same techniques as in previous sections. The advantage of the stochastic model is that the decision maker can find solutions that account for some aspects of the implementation uncertainty. However, as the number of scenarios $|K|$ grows large, the increasing number of integer variables makes the stochastic MaxProd model intractable to solve directly with a commercial MIP solver. To solve large instances of stochastic MaxProd, in Section 4.6 we describe a decomposition algorithm derived from the Frank-Wolfe-based progressive hedging algorithm given by Boland et al. (2018).

A Stochastic Extension of BiCosMos

We next discuss a stochastic extension of the BiCosMos model that maximizes the expected value of the worst-case biochemical yield and best-case cellular growth. Although stochastic MaxProd is able to account for some aspects of the implementation uncertainty, it is a pessimistic model of what will occur within the cell. By considering a stochastic extension of BiCosMos, we aim to take a less conservative view of the cellular response while still incorporating implementation uncertainty. In addition, a major benefit of a bi-objective model is that it gives a portfolio of possible solutions rather than a single solution.

For the stochastic extension of BiCosMos, we use the multi-objective method described by Gutjahr and Pichler (2016) to reduce the ill-defined stochastic bi-objective model to a deterministic equivalent bi-objective model by maximizing the expected best-case growth and worst-case yield. The full bilevel formulation of the stochastic BiCosMos model in which we maximize the expected value of the worst-case yield and give a lower bound L

on the expected best-case growth is as follows:

$$\begin{aligned}
\max_{\delta^\alpha,\delta^\beta,\alpha,\beta,v,w} \ & \frac{1}{|K|}\sum_{k\in K} v_y^k \\
\text{s.t. } & \frac{1}{|K|}\sum_{k\in K} w_g^k \geq L \\
& (\delta^\beta, \delta^\alpha, \alpha, \beta) \text{ satisfy } (4.1)-(4.4) \\
& w^k \in X_k(\alpha,\beta,\delta^\alpha,\delta^\beta) && \forall k \in K \\
& v^k \in \arg\min_v \{v_y : v \in X_k(\alpha,\beta,\delta^\alpha,\delta^\beta)\} && \forall k \in K.
\end{aligned}$$

Alternatives to modeling the objective could also be considered, such as maximizing the worst scenario's worst-case yield or minimizing the number of scenarios with worst-case yield below a given threshold.

In our test instances, we observe that when modeled in this way, the stochastic extension of BiCosMos produces some undesirable results. Specifically, we observe that while the expected worst-case yield and expected best-case growth are both nonzero, individual scenarios tend to have undesirable behavior similar to the behavior of solutions to the CosMos model. In particular, some scenarios will have high yield and very little or no growth, while the rest of scenarios have high enough growth to satisfy the expected growth constraint. See Figure (4.3) for a demonstration of this behavior on a small (10 scenario) test instance solved directly with a commercial MIP solver. We are therefore motivated to alter the model to discourage these outcomes, instead producing solutions with scenarios having both positive yield and growth.

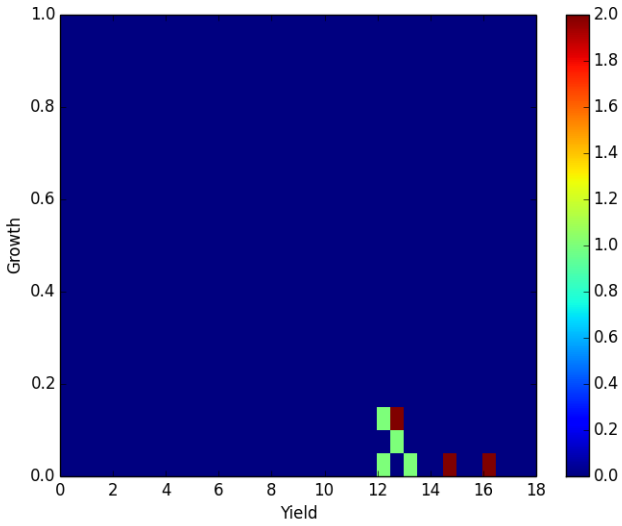


Figure 4.3: Demonstration of small stochastic BiCosMos solution that has scenarios with > 0 yield but 0 growth.

In this section, we propose a modification to the stochastic BiCosMos model in which we solve two auxiliary bi-objective problems to recover a subset of the Pareto frontier of the stochastic BiCosMos problem. We propose using a productivity-based objective in which we maximize expected *approximate* productivity, which we define as the product of worst-case yield and best-case growth. Note this is not the same as the productivity measure in MaxProd, which is the product of the value of v_y and the value of v_g in the worst-case productivity solution. We propose maximizing expected approximate productivity and expected yield simultaneously, and separately maximizing expected approximate productivity and expected growth simultaneously. In particular, we consider the following bi-objective problems:

$$\max_{\delta^\alpha,\delta^\beta,\alpha,\beta,v,w} \{\frac{1}{|K|}\sum_{k\in K} v_y^k w_g^k, \frac{1}{|K|}\sum_{k\in K} w_g^k\} \tag{PG}$$

$$\text{s.t. } (\delta^\beta, \delta^\alpha, \alpha, \beta) \text{ satisfy } (4.1)-(4.4)$$

$$w^k \in X_k(\alpha,\beta,\delta^\alpha,\delta^\beta) \qquad \forall k \in K$$

$$v^k \in \arg\min_v \{v_y : v \in X_k(\alpha,\beta,\delta^\alpha,\delta^\beta)\} \qquad \forall k \in K;$$

$$\max_{\delta^\alpha,\delta^\beta,\alpha,\beta,v,w} \{\frac{1}{|K|}\sum_{k\in K} v_y^k w_g^k, \frac{1}{|K|}\sum_{k\in K} v_y^k\} \tag{PY}$$

$$\text{s.t. } (\delta^\beta, \delta^\alpha, \alpha, \beta) \text{ satisfy } (4.1)-(4.4)$$

$$w^k \in X_k(\alpha,\beta,\delta^\alpha,\delta^\beta) \qquad \forall k \in K$$

$$v^k \in \arg\min_v \{v_y : v \in X_k(\alpha,\beta,\delta^\alpha,\delta^\beta)\} \qquad \forall k \in K;$$

Our consideration of these two stochastic bi-objective models is motivated by the following result for the deterministic problem:

Claim 4.1. *Consider the following deterministic variations of the BiCosMos model:*

$$\max_{\delta^\alpha,\delta^\beta,\alpha,\beta,v,w} \{v_y w_g, w_g\} \tag{D-PG}$$

$$\textit{s.t. } (\delta^\beta, \delta^\alpha, \alpha, \beta) \textit{ satisfy } (4.1)-(4.4)$$

$$w \in X(\alpha,\beta,\delta^\alpha,\delta^\beta)$$

$$v \in \arg\min_v \{v_y : v \in X(\alpha,\beta,\delta^\alpha,\delta^\beta)\};$$

$$\max_{\delta^\alpha,\delta^\beta,\alpha,\beta,v,w} \{v_y w_g, v_y\} \tag{D-PY}$$

$$\textit{s.t. } (\delta^\beta, \delta^\alpha, \alpha, \beta) \textit{ satisfy } (4.1) - (4.4)$$

$$w \in X(\alpha, \beta, \delta^\alpha, \delta^\beta)$$

$$v \in \arg\min_v \{v_y : v \in X(\alpha, \beta, \delta^\alpha, \delta^\beta)\}.$$

Any Pareto optimal solution of D-PG is a Pareto optimal solution of BiCosMos. Similarly, any Pareto optimal solution of D-PY is a Pareto optimal solution of BiCosMos.

Proof. Suppose the solution $(\delta^\alpha, \delta^\beta, \alpha, \beta, v, w)^*$ is Pareto optimal for (D-PG). Then for any point (v_y, w_g) that is feasible to (D-PG) where $w_g > w_g^*$, we have

$$v_y w_g < v_y^* w_g^* \implies v_y < v_y^*.$$

Furthermore, for any feasible point (v_y, w_g) such that $v_y > v_y^*$, either (a) $v_y w_g > v_y^* w_g^*$ or (b) $v_y w_g \leq v_y^* w_g^*$. If (a), by Pareto optimality we have

$$v_y w_g > v_y^* w_g^* \implies w_g < w_g^*.$$

If (b):

$$v_y w_g \leq v_y^* w_g^* \implies w_g < w_g^*.$$

Thus, because $w_g > w_g^* \implies v_y < v_y^*$ and $v_y < v_y^* \implies w_g > w_g^*$ for any point that is feasible to (D-PG) – wich is also feasible to BiCosMos – the point $(\delta^\alpha, \delta^\beta, \alpha, \beta, v, w)^*)$ must be Pareto optimal to BiCosMos.

The proof is nearly identical for (D-PY), so we exclude it for brevity. Thus, by solving (D-PY) and (D-PG) we recover a subset of the points that are Pareto optimal to BiCosMos.

□

Although this result does not have an obvious extension to the stochastic models (PG) and (PY), in practice we observe that solutions to these two auxiliary problems produce points along (or very close to) the Pareto frontier of the direct stochastic extension of BiCosMos. Thus, we argue that solving (PG) and (PY) produces points with different trade-offs between yield and growth while causing many scenarios to have nonzero yield and nonzero growth. Similar to stochastic MaxProd, we use a progressive hedging-based heuristic algorithm to produce high-quality feasible solutions to both (PG) and (PY) as described in Section 4.6.

4.6 A Lagrangian Decomposition Heuristic with Dynamic Budget Allocation

We propose a Lagrangian decomposition heuristic with dynamic weight allocation for solving instances of the stochastic BiCosMos model. Scenario-based stochastic models are often solved with algorithms that rely on decomposing the model into scenario subproblems. However, the stochastic extension of BiCosMos has a constraint that enforces a lower bound on the expected value of one of the objectives, which has no natural decomposition across scenarios. Thus, we propose an additional heuristic that allocates the lower bound – or "budget" – to each scenario subproblem in a strategic way. In our heuristic, we alternate between optimizing the Lagrangian dual multipliers for a fixed allocation and re-allocating the budget to improve the solution for fixed dual multipliers.

For ease of notation, we present a general-form stochastic mixed-integer program (SMIP), where x are the first-stage variables and $(y_k)_{k\in K}$ are the second-stage (recourse) variables for a set of scenarios K. Let $c \in \mathbb{R}^n$ be a given cost vector and X be a mixed-integer linear set. We write (SMIP) as

$$\max_x \{c^T x + \mathcal{Q}(x) \ : \ x \in X\}. \tag{SMIP}$$

The function $\mathcal{Q}$ is a real-valued expected recourse function that depends on a random variable ξ. We assume ξ is taken from a discrete distribution with possible outcomes $\xi_1, \cdots, \xi_{|K|}$, each of which has a known probability p_k for $k \in K$. We can reformulate (SMIP) as a deterministic equivalent as follows:

$$\max_{x,y} \Big\{ c^T x + \sum_{k \in K} p_k q_k^T y_k \ : (x, y_k) \in P_k \ \forall k \in K \Big\}.$$

We define P_k to be the set of constraints on both the first-stage variables $x \in X$ and the second-stage variables $y_k \in Y_k$, as well as any constraints linking the variables. Because of the special structure of the deterministic equivalent of (SMIP), there are a number of algorithms that rely on scenario decomposition. To decompose (SMIP) by scenario, we introduce a copy of of the first-stage variables x_k for every $k \in K$. We then add nonanticipativity constraints $x_k = \bar{x}$ for all $k \in K$, which enforces the requirement all first-stage variables are equal across scenarios. A number of decomposition-based algorithms relax the nonanticipativity condition through the use of a *nonanticipative Lagrangian dual function:*

$$\Phi(\lambda) := \max_{x,\bar{x},y} \Big\{ \sum_{k \in K} [p_k (c^T x_k + q_k^T y_k - \lambda_k^T (x_k - \bar{x}))] : \\ (x_k, y_k) \in P_k \forall k \in K, \ \bar{x} \in \mathbb{R}^n \Big\}.$$

In this function, $\lambda = (\lambda_1, \cdots, \lambda_{|K|})$ is the vector of multipliers associated with each nonanticipativity constraint. In order for $\Phi(\lambda)$ to be dual feasible, it must be bounded above. Thus, we require that $\sum_{k \in K} p_k \lambda_k = 0$. This assumption removes the $\bar{x}$ term from the objective, creating a separable function $\Phi(\lambda) = \sum_{k \in K} p_k \phi_k(\lambda_k)$, where for $k \in K$

$$\phi_k(\lambda_k) := \max_{x,y} \{ (c + \lambda_k)^T x + q_k^T y_k \ : \ (x, y_k) \in P_k \}. \tag{4.17}$$

It is well known that for any choice of λ, the value of $\Phi(\lambda)$ is an upper bound on the

optimal value of (SMIP). To find the best upper bound, we solve the Lagrangian dual problem of minimizing $\Phi(\lambda)$ over feasible choices of λ:

$$\min_{\lambda}\{\Phi(\lambda) : \sum_{k\in K} p_k\lambda_k = 0\}. \tag{LD}$$

A number of methods for solving stochastic programs, including progressive hedging algorithms, rely on solving (LD) to find upper bounds on the objective value of the original stochastic problem. Upper bounds on instances of MaxProd can be obtained with these methods. The solutions to the subproblems in (4.17) can be used within heuristics to find good feasible solutions to MaxProd.

Decomposing Stochastic BiCosMos Through Dynamic Budget Allocation

In order to use a decomposition algorithm to solve instances of the stochastic BiCosMos model, a method for allocating the lower bound on expected worst-case yield (L^y) or expected best-case growth (L^g) is needed. In general, suppose (SMIP) has an additional constraint

$$\sum_{k\in k} d_k^T y_k \geq L$$

for a fixed value L and some vector d_k that could depend on the scenario. In (PY), the constraint of this form is

$$\frac{1}{|K|}\sum_{k\in K} v_y^k \geq L^y, \tag{4.18}$$

and in (PG) the constraint is

$$\frac{1}{|K|}\sum_{k\in K} w_g^k \geq L^g. \tag{4.19}$$

To perform a scenario decomposition, we must assign a value ℓ_k to each scenario in such a way that $\sum_{k\in K} \ell_k = L$. We propose a heuristic for finding a budget allocation that improves on a given feasible solution.

The budget allocation heuristic takes as an input a set of Lagrangian dual multipliers obtained from solving (SMIP) with a decomposition algorithm, $\hat{\lambda}$. The BUDGETSET subroutine is motivated by an observation that for scenario $k \in K$, the value of the Lagrangian relaxation function for fixed λ_k decreases as ℓ_k increases. Thus, the Lagrangian relaxation function relies not only on the values of λ_k, but also on the values of ℓ_k for $k \in K$. For each $k \in K$, we define

$$\phi_k(\lambda_k, \ell_k) := \max_{x,y}\{(c + \lambda_k)^T x + q_k^T y_k \ : \ (x, y) \in P_k, \ d_k^T y_k \geq \ell_k\}. \tag{4.20}$$

For fixed $\hat{\lambda}_k$, there should exist some allocation of the ℓ_k to each scenario of maximal value of the function $\sum_{k\in K} \phi_k(\hat{\lambda}_k, \ell_k)$ such that $\sum_{k\in K} \ell_k \geq L$. Thus, it is left to determine (a) how to approximate each scenario's Lagrangian relaxation function as a function of ℓ_k and (b) how to choose an optimal value of ℓ_k for each scenario based on that approximation. The BUDGETSET subroutine relies on a simple piecewise linear approximation of $\phi_k(\lambda_k, \ell_k)$.

We observe that in our test instances, for fixed λ_k, the value of $\phi_k(\lambda_k, \ell_k)$ is roughly constant until a "breakpoint" value $\ell_k = b_k$. The function then decreases at an approximately linear rate until $\phi_k(\lambda_k, h^{\max}) = 0$, where $h^{\max}$ is the maximum value ℓ_k can take before the problem becomes infeasible. Thus, we approximate $\phi_k(\lambda_k, \ell_k)$ as:

$$\bar{\phi}_k(\ell_k) = \begin{cases} \phi_k(\lambda_k, 0) & \text{if } 0 \leq \ell_k \leq b_k \\ -\frac{\phi_k(\lambda_k,0)}{h^{\max}-b_k}\ell_k + \frac{\phi_k(\lambda_k,0)h^{\max}}{h^{\max}-b_k} & \text{if } b_k < \ell_k \leq h^{\max}. \end{cases}$$

The breakpoint b_k can be found using a standard binary search algorithm (BinSearch). In our implementation of BinSearch, we compute $\phi_k(\lambda_k, 0)$ and, if $\phi_k(\lambda_k, 0) > 0$, we search for a point ℓ_k^* where $\phi(\lambda_k, \ell_k^*) \approx \phi(\lambda_k, 0)$ but $\phi(\lambda_k, \ell_k^* + \epsilon)/\phi(\lambda_k, \ell_k^*) < \gamma$ for some given tolerance

γ. Once $\bar{\phi}_k(\ell_k)$ is constructed for each $k \in K$, we find the optimal choice of ℓ by solving

$$\ell \in \arg\max_{w} \{\sum_{k \in K} \bar{\phi}(w_k) : \sum_{k \in K} w_k \geq L\} \tag{W-LP}$$

(W-LP) takes as an input the set of breakpoints $b_k \; \forall k \in K$ and the piecewise approximation $\bar{\phi}_k(\ell_k) \; \forall k \in K$. The BUDGETSET subroutine is given in pseudocode in Algorithm 1. We define the feasible region of the kth scenario subproblem as $P_k(\ell_k) := P_k \cap \{(x, y) : d_k^T y \geq \ell_k\}$.

Algorithm 1 BUDGETSET

function BUDGETSET($\lambda, h^{\max}$)
  for $k \in K$ **do**
    $\phi_k(\lambda_k, 0) \leftarrow \max\{(c + \lambda_k)^T x + q_k^T y \; : \; (x, y) \in P_k(0)\}$
    $b_k \leftarrow$ BinSearch($h^{\max}, \phi_k(\lambda_k, 0), \gamma$)
  Let ℓ be a solution to (W-LP), using $\bar{\phi}_k$ as defined by b_k and $\phi_k(\lambda_k, 0)$ for $k \in K$
return $(\ell_k)_{k \in K}$.

Modified Progressive Hedging Heuristic

Given a budget allocation, we can fully decompose (SMIP) with a budget constraint by scenarios. It remains to discuss an algorithm to find an optimal solution to (LD) given a fixed budget allocataion. To solve (LD), we apply the Frank-Wolfe-based progressive hedging algorithm (PH+FW) described by Boland et al. (2018). Our implementation of PH+FW employs a single iteration of the augmented quadratic subproblem routine at every PH iteration. We alternate between running PH+FW and BUDGETSET to allocate the lower bounds on the second objective to each scenario. In our implementation, we run BUDGETSET every time the PH+FW algorithm converges. Thus, we alternate between optimizing with respect to a fixed budget allocation and re-allocating the budget given a set of Lagrangian multipliers.

In each iteration of PH+FW, we solve an augmented Lagrangian dual subproblem for each scenario. This augmented subproblem has a quadratic objective function for some

choice of a penalty parameter $\rho > 0$:

$$L_k^\rho(x, y, \bar{x}, \lambda_k) := c^T x + q_k^T y + \lambda_k^T (x - \bar{x}) + \frac{\rho}{2}\|x - \bar{x}\|_2^2. \tag{4.21}$$

The PH+FW+BUDGETSET heuristic is described in pseudocode in Algorithm 2. The parameters $\epsilon_1 > 0$ and $\epsilon_2 > 0$ are convergence tolerances. The first (ϵ_1) determines how frequently the BUDGETSET subroutine is run and the second (ϵ_2) indicates that the PH+FW algorithm has stopped making significant progress in updating the Lagrangian multipliers. These tolerances could be equal. The parameter $i_{\max}$ defines a maximum number of iterations of the PH+FW+BUDGETSET loop. The set V_k^i contains feasible solutions to the kth scenario subproblem.

Algorithm 2 PH+FW+BUDGETSET Heuristic applied to (SMIP)

function PH+FW+BUDGETSET($(V_k^0)_{k\in K}, \lambda^0, \rho, \epsilon_1, \epsilon_2, i_{\max}, (\ell_k^0)_{k\in K}$)
  for $k \in K$ **do**
    $(\hat{x}_k^0, \hat{y}_k^0) \in \arg\max_{x,y}\{(c + \lambda_k^0)^T x + q_k^T y \ : \ (x, y) \in P_k(\ell_k^0)\}$
    $V_k^1 \leftarrow V_k^0 \cup \{\hat{x}_k^0, \hat{y}_k^0\}$
  $\phi^0 \leftarrow \frac{1}{|K|}\sum_{k\in K}[(c + \lambda_k^0)^T x_k^0 + q_k^T y_k^T]$
  $\bar{x}^0 \leftarrow \frac{1}{|K|}\sum_{k\in K} x_k^0$
  $\lambda_k^1 \leftarrow \lambda_k^0 - \rho(x_k^0 - \bar{x}^0)$ for all $k \in K$
  for $i = 1, \cdots, i_{\max}$ **do**
    for $k \in K$ **do**
      $(\phi_k^i, \hat{x}_k^i, \hat{y}_k^i) \leftarrow \max_{x,y}\{(c + \lambda_k^i)^T x + q_k^T y \ : \ (x, y) \in P_k(\ell_k)\}$
      $V_k^i \leftarrow V_k^{i-1} \cup \{\hat{x}_k^i, \hat{y}_k^i\}$
      $(x_k^i, y_k^i) \in \arg\min_{x,y}\{L_k^\rho(x, y, \bar{x}^{i-1}, \lambda_k^i) \ : (x, y) \in \text{conv}(V_k^i)\}$
      $\lambda_k^i \leftarrow \lambda_k^{i-1} - \rho(x_k^i - \bar{x}^{i-1})$
    if $\frac{1}{|K|}\sum_{k\in K}\|x_k^i - \bar{x}^{i-1}\|_2^2 < \epsilon_1$ **then**
      $(\hat{\ell}_k)_{k\in K} \leftarrow$ BUDGETSET($\lambda^i, h^{\max}$)
      if $\|\ell - \hat{\ell}\|_2^2 < \epsilon_2$ **then return** $((x_k^i, y_k^i)_{k\in K}, \hat{x}^i, \lambda^i, \phi^i)$
      for $k \in K$ **do**
        Delete any point with $d_k^T y_k < \ell_k$ from V_k^i
      $\ell \leftarrow \hat{\ell}$
  return $((x_k^{i_{\max}}, y_k^{i_{\max}})_{k\in K}, \hat{x}^{i_{\max}}, \lambda^{i_{\max}}, \phi^{i_{\max}})$

Exact Algorithm for a Convex Relaxation

Although in our implementation we use the heuristic described in the previous section to find solutions to the stochastic BiCosMos model, there is an alternative exact algorithm for convex stochastic problems with a budget constraint. Namely, we aim to solve the following problem, where the budget on individual scenarios is a decision variable:

$$\begin{aligned}
\max_{x,\bar{x},y,\ell} \quad & \sum_{k\in K} p_k(c^T x_k + q_k^T y_k) \\
\text{s.t. } & (x_k, y_k) \in P_k \\
& \sum_{k\in K} \ell^k \geq L \\
& d_k^T y_k - \ell_k \geq 0 && \forall k \in K \\
& x_k = \bar{x} && \forall k \in K.
\end{aligned}$$

If any first- or second-stage variables are integer, the set P_k is nonconvex. In that case, existing methods are insufficient to find both the optimal solution (x^*, y^*) and the optimal budget allocation ℓ^*. However, if we instead consider the relaxed problem

$$\begin{aligned}
\max_{x,\bar{x},y,\ell} \quad & \sum_{k\in K} p_k(c^T x_k + q_k^T y_k) \\
\text{s.t. } & (x_k, y_k) \in \text{conv}(P_k) \\
& \sum_{k\in K} \ell^k \geq L \\
& d_k^T y_k - \ell_k \geq 0 && \forall k \in K \\
& x_k = \bar{x} && \forall k \in K.
\end{aligned}$$

we can find the optimal solution to the relaxed problem along with the optimal budget allocation. Note that this relaxed problem, apart from the constraint $\sum_{k\in K} \ell^k \geq L$, is the problem we solve with the modified progressive hedging algorithm described in the previ-

ous section. The heuristic described in the previous section relies on dual decomposition methods, but it is possible to incorporate primal decomposition methods as well. Specifically, a Benders decomposition algorithm could be employed where the master problem is as follows:

$$\begin{aligned} &\max_{\ell} G(\ell) \\ &\text{s.t.} \sum_{k \in K} \ell^k \geq L. \end{aligned}$$

Then, with $(\ell^k)_{k \in K}$ fixed, the subproblem is

$$\begin{aligned} G(\ell) = \max_{x, \bar{x}, y} & \sum_{k \in K} p_k (c^T x_k + q_k^T y_k) \\ \text{s.t. } & (x_k, y_k) \in \text{conv}(P_k) \\ & d_k^T y_k \geq \ell^k && \forall k \in K \\ & x_k = \bar{x} && \forall k \in K. \end{aligned}$$

The subproblem can be solved with the progressive hedging algorithm described in the previous section. Benders cuts can be derived from subgradients of the subproblem. The convexity of this problem guarantees an optimal solution will be found with a finite number of Benders cuts. Alternatively, a cross decomposition algorithm that alternates between finding valid Benders cuts for the complicating budget constraint with fixed Lagrangian dual multipliers and a Lagrangian dual decomposition-based algorithm with a fixed budget allocation will converge to the solution to the relaxed problem (Van Roy (1983)).

4.7 Stochastic Model Computational Results

In our second computational study, we compare the effectiveness of our stochastic modeling choices in finding solutions that are robust to implementation uncertainty. We find solutions

to instances of the stochastic extensions of MaxProd and BiCosMos and evaluate metrics of interest to determine how sensitive each solution is. We also compare the solutions of the stochastic models to the solutions to the deterministic CosMos and BiCosMos models to determine relative effectiveness of the stochastic extensions.

We evaluate the effectiveness of the stochastic models by replicating the sampling procedure discussed in Cotten and Reed (2013) and Section 4.4 on solutions produced by each model. To generate solutions, we use data from *E. coli* cells. We present results for the same two reconstructions of the *E. coli* metabolic network as used in the deterministic study (Orth et al. (2010a); Reed et al. (2003)). We set the number of bound changes to be $k = 3$ for all models. To solve instances of stochastic MaxProd, we use the modified progressive hedging algorithm described by Boland et al. (2018). To solve instances of stochastic BiCosMos, we use the modified progressive hedging algorithm combined with the budget-setting subroutine described in Section 4.6. We solve each problem on instances with 50, 100, and 300 scenarios. Results for instances with 500 scenarios are similar to those obtained with 300 scenarios, but instances with more than 300 scenarios, particularly the iJR904 instances, approach computational intractability with the methods used. We cannot rule out the possibility that the quality of solutions to instances with more scenarios (e.g., 1000 scenarios) could improve significantly. For each study, we run 10 replications of the instances with different random samples of the bound change coefficients. The details of the distribution and data used in these instances can be found in Appendix C.

The models are implemented in Python 2.7 and solved with Gurobi 7.0.2. After initializing the solution set used in the modified progressive hedging algorithm with one solution for each scenario subproblem, we impose a 1-node limit on each scenario subproblem and do not update the solution set for any scenario for which a feasible solution is not found at the root node. To ensure Gurobi emphasizes finding a good feasible solution at the root node, we increase the Gurobi heuristics parameter to 0.6 (60% of solution time is dedicated to using heuristics to find feasible solutions). Within the progressive hedging algorithm, we

use a penalty parameter of $\rho = 2.0$ and set the maximum number of iterations to 100. For BiCosMos, the tolerance for running the BUDGETSET subroutine is 10e-5 and the budget allocation is initialized as $\ell_1 = \ell_2 = \cdots = \ell_{|K|} = L$. The PH+FW convergence tolerance is also 10e-5. This study is conducted on an Intel Xeon X5650 server with 12 cores at 2.66Ghz and 128GB of RAM using 24 threads.

Stochastic MaxProd Computational Results

We begin with a discussion of the results obtained solving stochastic MaxProd on the core model and iJR904 reconstructions of *E. coli*. In the core model reconstruction, the maximum possible yield is 16.819 and the maximum possible growth is 0.825. In the larger iJR904 reconstruction, the maximum possible yield is 17.143 and the maximum possible growth is 0.921. For each instance size (50, 100, and 300 scenarios of each reconstruction), we solve 10 replications of the stochastic MaxProd model. After running the modified progressive hedging algorithm, we obtain a first-stage solution for each scenario subproblem. As a heuristic to find a good feasible solution, we evaluate each first-stage solution on all second-stage problems to find the first-stage solution that produces the highest expected worst-case productivity. Similar to the deterministic instances, we evaluate the best solution on 10,000 randomly sampled instances of the minimum yield and maximum growth subproblems.

To evaluate the results, we define an "OK" scenario as one in which the best-case biomass is at least 0.2 and worst-case yield is at least 2.0 and a "Good" scenario as one in which the best-case growth is at least 0.4 and the worst-case yield is at least 4.0. Any scenario with either very low yield or very low growth would be considered "bad," and thus we are interested in solutions that produce few "bad" scenarios. In Table 4.1 we provide summary statistics for the 10 replications of the 50, 100, and 300 scenario instances of the *E. coli* core model reconstruction. We report the same statistics for the *E. coli* iJR904 reconstruction in Table 4.2. In particular, we record the average percentage of scenarios that are "OK" and the average percentage of scenarios that are "Good." We also report

the minimum and maximum percentage of "OK" and "Good" scenarios across the 10 replications and the sample standard deviation.

Stochastic MaxProd – *E. coli* Core Model								
	OK				Good			
# Scen	Avg. %	Min %	Max %	St. Dev.	Avg. %	Min %	Max %	St. Dev.
50	2.3	0.0	6.7	2.6	0.7	0.0	2.7	1.2
100	40.4	0.0	67.1	25.2	21.6	0.0	49.1	17.0
300	42.6	0.0	71.8	25.9	15.0	0.0	38.6	14.1

Table 4.1: Summary statistics for stochastic MaxProd on 10 replications of instances of the core model with 50, 100, and 300 scenarios

Stochastic MaxProd – *E. coli* iJR904								
	OK				Good			
# Scen	Avg. %	Min %	Max %	St. Dev.	St. Dev. %	Min %	Max %	St. Dev.
50	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
100	0.0	0.0	0.3	0.0	0.0	0.0	0.0	0.0
300	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 4.2: Summary statistics for stochastic MaxProd on 10 replications of instances of iJR904 with 50, 100, and 300 scenarios

From the core model results, we observe that the average percentage of "OK" and "Good" scenarios tends to increase from 50 to 100 scenarios, then stay roughly the same from 100 to 300 scenarios. The standard deviation of the 10 replications similarly increases from 50 to 100 scenarios, then stays around the same value from 100 to 300 scenarios. The larger instance (iJR904) finds no solutions that produce "OK" or "Good" scenarios and appears to be an overly conservative model of cellular behavior. For the core model, the results indicate that it is likely good to use as many scenarios as is computationally tractable, which for this study is 300.

Stochastic BiCosMos Computational Results

Next we examine the stochastic extension of the BiCosMos model solving the two auxiliary approximate productivity problems on the *E. coli* core model and iJR904 reconstructions. For each instance size (50, 100, and 300 scenarios of each reconstruction), we solve 10

replications of the stochastic BiCosMos model for different values of L^y and L^g. For a given lower bound on expected growth or yield, after running the modified progressive hedging algorithm, we obtain a first-stage solution for each scenario subproblem. As a heuristic to find a good feasible solution, we evaluate each first-stage solution on all second-stage problems to find the first-stage solution that produces the highest expected approximate productivity. Similar to the deterministic instances, we evaluate the best solution on 10,000 instances of the minimum yield and maximum growth subproblems.

Our first observation is that (PY) tends to be infeasible for values of L^y greater than the value of the worst-case yield obtained in a solution to (PG) with $L^g = 0$. In other words, no additional efficient solutions can be discovered by solving (PY) with different values of L^y. An interesting feature of th PH+FW algorithm is that, because it solves each scenario subproblem independently, it is possible to obtain a solution to a model that is infeasible in its extensive form. Any such first-stage solution that is found can be fixed and the second-stage problems solved. In our implementation of (PY), many of the second-stage problems end up being infeasible, so a solution found with PH+FW to (PY) has similar behavior to that of the deterministic CosMos model, where many scenarios are infeasible (0 yield, 0 growth), and the remaining scenarios have high yield but little or no growth. Thus, because we do not find any additional efficient points with (PY), we only perform our computational study on (PG) for different values of L^g.

We begin by solving instances of stochastic BiCosMos with data from the *E. coli* core model with different values of L^g. We aim to discover the trade-off between best-case growth and approximate productivity. Figure (4.4) shows these results, plotting best-case growth against approximate productivity for the best solution for a variety of values of L^g.

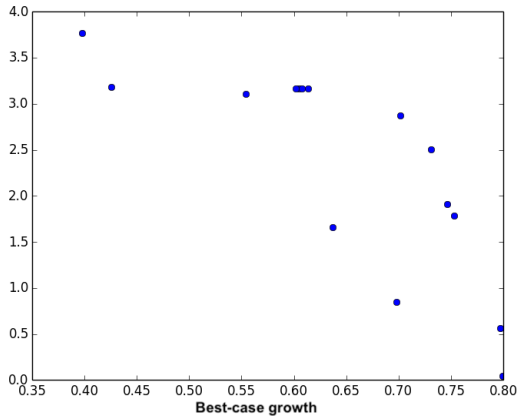


Figure 4.4: Best-case growth vs. approximate productivity in heuristic solutions to instances of stochastic BiCosMos with data from the *E. coli* core model for different values of L^g.

This plot demonstrates that it is possible to determine some aspects of the trade-off between growth and productivity by using the methods described in the preceding sections. However, there are also some sizable gaps between points, indicating that it may not be possible to find all possible trade-offs. We note that values obtained for instances of iJR904 display similar behavior and thus we do not present a separate plot.

To evaluate solutions along this curve, we once again define an "OK" scenario as one in which the best-case biomass is at least 0.2 and worst-case yield is at least 2.0 and a "Good" scenario as one in which the best-case growth is at least 0.4 and the worst-case yield is at least 4.0. In Table 4.3 we provide summary statistics for the 10 replications of the 50, 100, and 300 scenario instances of the *E. coli* core model reconstruction with $L^g = 0.424$. We repeat the study for a point closer to the right edge of the solution curve, specifically $L^g = 0.778$ and report the results in Table 4.4. We record the average percentage of scenarios that are "OK" and the average percentage of scenarios that are "Good." We also report the minimum and maximum percentage of "OK" and "Good" scenarios across the 10 replications and the sample standard deviation. We report the same statistics for the *E. coli* iJR904 reconstruction in Tables 4.5 and 4.6 for the same values of L^g.

Stochastic BiCosMos ($L^g = 0.424$) – *E. coli* Core Model								
	OK				Good			
# Scen	Avg. %	Min %	Max %	St. Dev.	Avg. %	Min %	Max %	St. Dev.
50	85.4	25.2	99.7	23.8	57.8	0.0	96.3	32.1
100	97.8	80.5	99.8	5.8	73.1	18.2	93.3	20.4
300	99.7	99.4	99.9	0.2	75.4	52.5	87.5	9.8

Table 4.3: Summary statistics for stochastic BiCosMos with $L^g = 0.424$ on 10 replications of instances of the core model with 50, 100, and 300 scenarios

Stochastic BiCosMos ($L^g = 0.778$) – *E. coli* Core Model								
	OK				Good			
# Scen	Avg. %	Min %	Max %	St. Dev.	Avg. %	Min %	Max %	St. Dev.
50	49.7	0.0	95.0	33.5	3.3	0.0	30.6	9.1
100	63.0	37.8	93.0	18.5	18.7	0.0	41.1	17.6
300	74.2	56.9	99.8	15.3	45.1	1.4	94.2	29.2

Table 4.4: Summary statistics for stochastic BiCosMos with $L^g = 0.778$ on 10 replications of instances of the core model with 50, 100, and 300 scenarios

Stochastic BiCosMos ($L^g = 0.424$) – *E. coli* iJR904								
	OK				Good			
# Scen	Avg. %	Min %	Max %	St. Dev.	Avg. %	Min %	Max %	St. Dev.
50	98.1	97.4	99.0	0.7	41.6	29.0	57.3	13.0
100	99.2	99.0	99.4	0.1	63.0	57.3	68.3	3.9
300	99.2	99.2	99.2	0.0	61.6	61.6	61.6	0.0

Table 4.5: Summary statistics for stochastic BiCosMos with $L^g = 0.424$ on 10 replications of instances of iJR904 with 50, 100, and 300 scenarios

Stochastic BiCosMos ($L^g = 0.778$) – *E. coli* iJR904								
	OK				Good			
# Scen	Avg. %	Min %	Max %	St. Dev.	Avg. %	Min %	Max %	St. Dev.
50	88.6	45.5	99.7	21.6	25.6	2.19	67.6	22.1
100	99.9	99.9	99.9	0.0	91.2	91.2	91.2	0.0
300	99.8	99.8	99.8	0.0	87.4	87.4	87.4	0.0

Table 4.6: Summary statistics for stochastic BiCosMos with $L^g = 0.778$ on 10 replications of instances of iJR904 with 50, 100, and 300 scenarios

In general, the core model results indicate that the average percent of "Good" and "OK" scenarios increases as the number of scenarios increases from 50 to 100, but then does not change as much from 100 to 300 scenarios. Furthermore, the standard deviation across the 10 instances generally decreases with increasing number of scenarios. The one exception to both of these trends this is in the core model when $L^g = 0.778$; the standard deviation of the percentage of "Good" scenarios increases with increasing number of scenarios and the average percentage of "Good" scenarios increases significantly from 100 to 300 scenarios. The minimum and maximum percentages also increase with increasing number of scenarios on that instance. The wide variance in some instances is likely due to the fact that the methods used are heuristic and do not give any guarantee on how close the solution found is to the optimal solution. There are a number of factors contributing to how well-solved an instance is if it surpasses the maximum number of iterations in the PH+FW+BUDGETSET heuristic, including whether the Lagrangian dual multipliers have fully converged and whether the budget allocation at the time of termination has been chosen well. Some larger instances (e.g., ones with hundreds of scenarios) were possibly terminated before a good solution could be found. For the core model reconstruction, it appears that it is preferable to use as many scenarios as is computationally tractable.

The results for the iJR904 network reconstruction again indicate that using more scenarios leads to higher solution quality. When $L^g = 0.778$, however, 100 scenarios is enough to result in roughly the same solution being found in all 10 instances. The values of the variables differ slightly between replications, but the differences are not great enough to produce different percentages of "OK" or "Good" scenarios. It is difficult to determine whether this is just an artifact of the heuristic methods used, or if using a very large number of scenarios would in fact produce the same solution. Regardless, when 100 scenarios are used and $L^g = 0.778$, over 90% of the 10,000 random samples are "Good" and nearly all random samples are "OK." Similarly, when $L^g = 0.424$ nearly all scenarios are "OK" and, on average, over 60% of them are "Good." Although there is still some variance across

the 10 instances, the standard deviation decreases by an order of magnitude between 50 and 100 scenarios when $L^g = 0.424$. When the number of scenarios is increased to 300, the results do not change significantly.

Finally, we compare the stochastic extensions of MaxProd and BiCosMos to their deterministic counterparts and to CosMos by computing the percent of "OK" and "Good" scenarios for the deterministic models and comparing the results to the stochastic model results. We do this for both the core model and iJR904 reconstructions and record the outcome in Tables 4.7 and 4.8. It is clear from the previous results that when solving the stochastic models, we should use as many scenarios as is computationally tractable for the highest solution quality. Thus, we compare the results for the 10 instances of 300 scenarios for both the stochastic MaxProd and stochastic BiCosMos models on both network reconstructions.

Model Comparison – *E. coli* Core Model								
	OK				Good			
Model	Avg. %	Min %	Max %	St. Dev.	Avg. %	Min %	Max %	St. Dev.
CM	4.2	-	-	-	0.2	-	-	-
MP	27.9	-	-	-	16.6	-	-	-
Stoch. MP	42.6	0.0	71.8	25.9	15.0	0.0	38.6	14.1
BCM (0.424)	99.6	-	-	-	69.8	-	-	-
Stoch. BCM (0.424)	99.7	99.4	99.9	0.2	75.4	52.5	87.5	9.8
BCM (0.778)	30.8	-	-	-	0.0	-	-	-
Stoch. BCM (0.778)	74.2	56.9	99.8	15.3	45.1	1.4	94.2	29.2

Table 4.7: Summary statistics for deterministic and stochastic model solutions with data from the *E. coli* core model evaluated on 10,000 randomly sampled scenarios. CM = CosMos, BCM = BiCosMos, MP = MaxProd.

To give a more complete visualization of these results, we also generate 2-dimensional histograms of worst-case yield versus best-case growth for the sampled solution to each model. As with the deterministic results in Section 4.4, the histograms display the relative frequency of samples that fall into discrete "boxes" of worst-case yield and best-case growth (e.g., yield between 0.5 and 1.0 and growth between 0.2 and 0.25). In these figures, a higher density of scenarios in a given box is represented by dark red and few or no scenarios in a

Model Comparison – *E. coli* iJR904								
	OK				Good			
Model	Avg. %	Min %	Max %	St. Dev.	Avg. %	Min %	Max %	St. Dev.
CM	8.0	-	-	-	0.0	-	-	-
MP	0.0	-	-	-	0.0	-	-	-
Stoch. MP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
BCM (0.424)	99.9	-	-	-	61.6	-	-	-
Stoch. BCM (0.424)	99.2	99.2	99.2	0.0	61.6	61.6	61.6	0.0
BCM (0.778)	99.6	-	-	-	3.1	-	-	-
Stoch. BCM (0.778)	99.8	99.8	99.8	0.0	87.4	87.4	87.4	0.0

Table 4.8: Summary statistics for deterministic and stochastic model solutions with data from the *E. coli* iJR904 model evaluated on 10,000 randomly sampled scenarios. CM = CosMos, BCM = BiCosMos, MP = MaxProd.

given box is represented by dark blue. Figure (4.5) displays the results for the deterministic models and two of the ten instances of each stochastic model with 300 scenarios, all with data from the *E. coli* core model reconstruction. We repeat the process in Figure (4.6) for solutions to all models with data from the *E. coli* iJR904 network reconstruction.

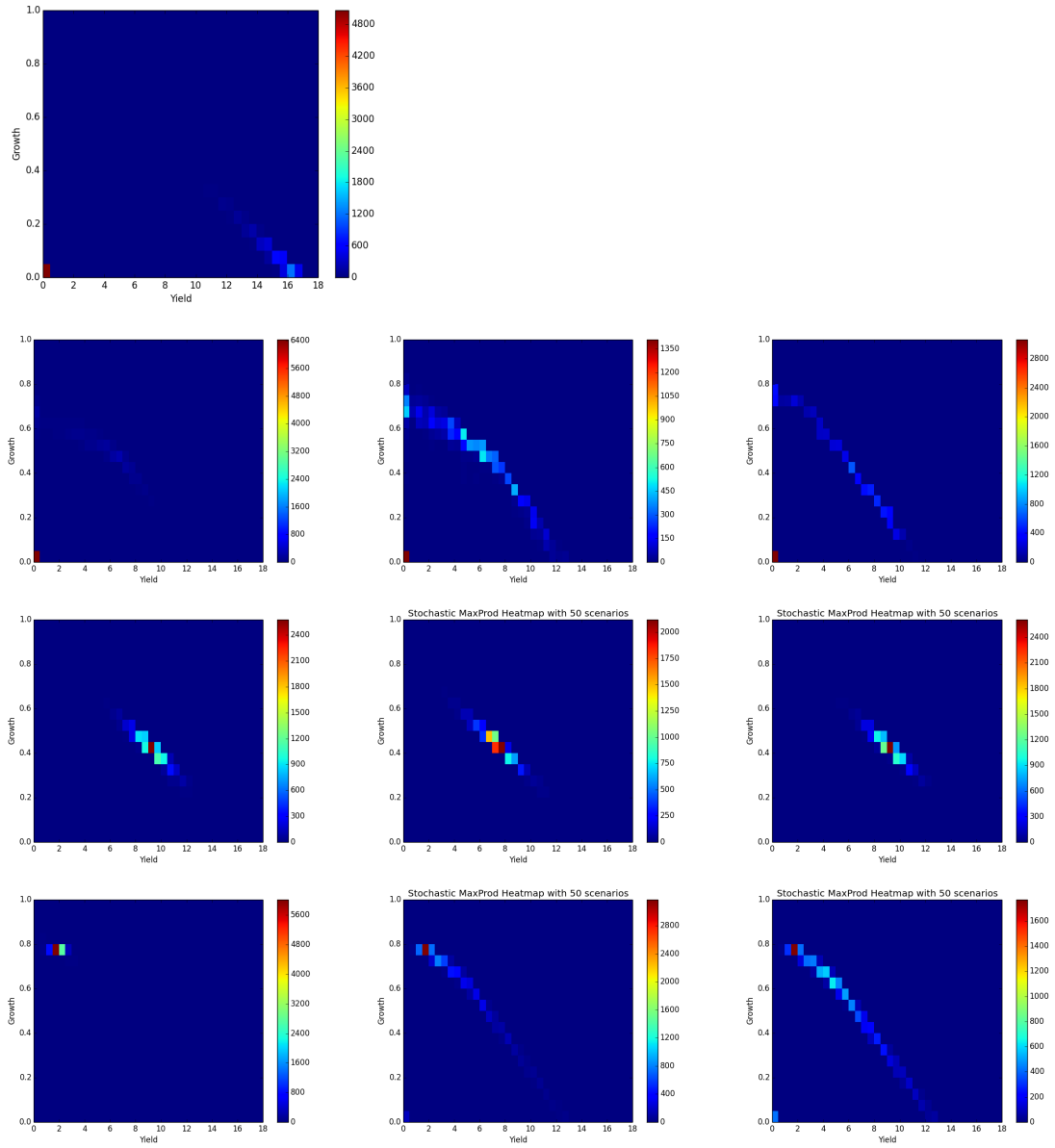


Figure 4.5: *E. coli* core model. Worst-case yield versus best-case growth on 10,000 samples of bound implementation success on solutions to the deterministic models beside a selected sample instance of their stochastic counterpart. Rows of figures are: CosMos, MaxProd vs. two instances of stochastic MaxProd, BiCosMos vs. two instances of stochastic BiCosMos with $L^g = 0.424$, and BiCosMos vs. two instances of stochastic BiCosMos with $L^g = 0.778$

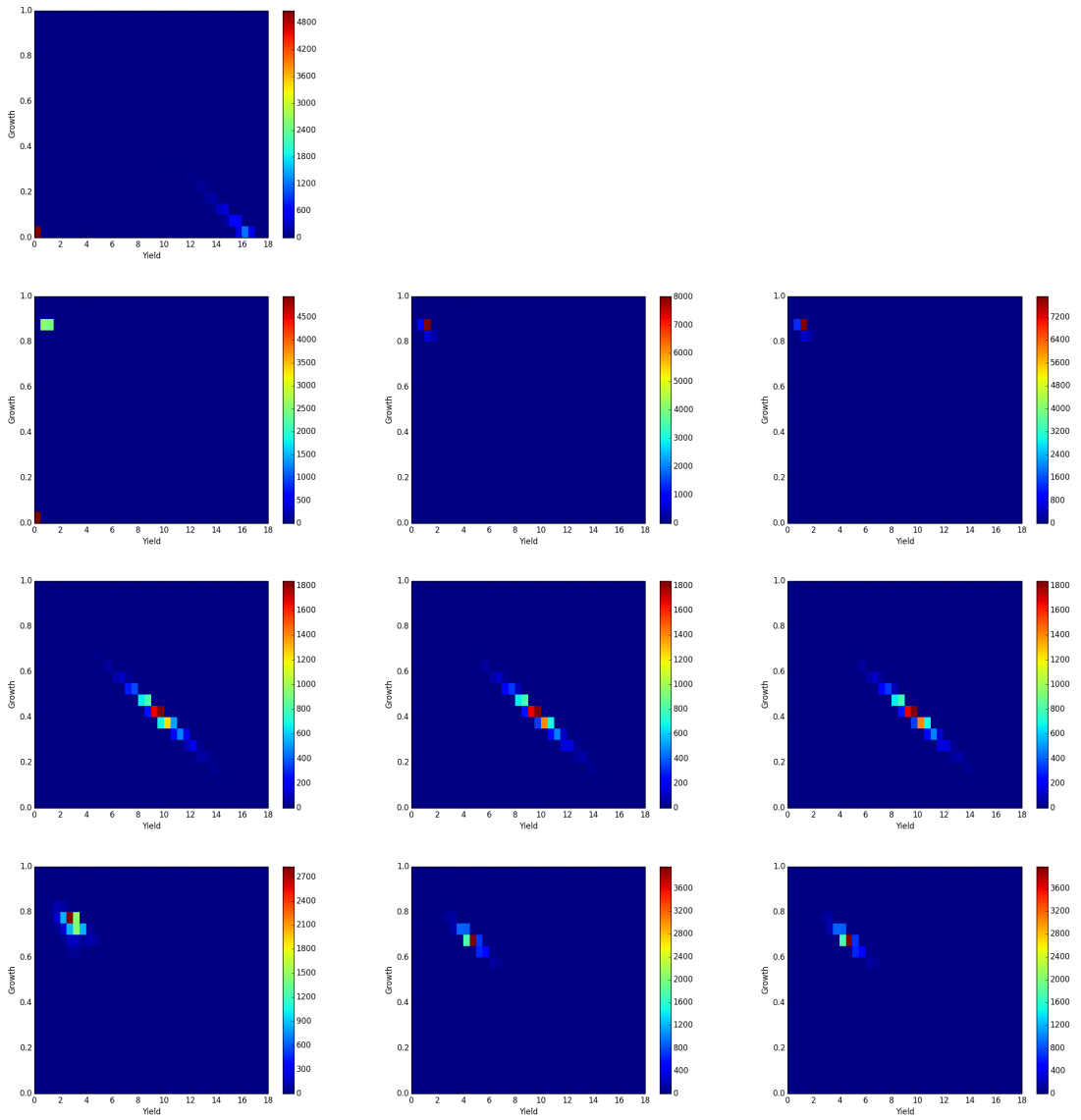


Figure 4.6: *E. coli* iJR904 model. Worst-case yield versus best-case growth on 10,000 samples of bound implementation success on solutions to the deterministic models beside a selected sample instance of their stochastic counterpart. Rows of figures are: CosMos, MaxProd vs. two instances of stochastic MaxProd, BiCosMos vs. two instances of stochastic BiCosMos with $L^g = 0.424$, and BiCosMos vs. two instances of stochastic BiCosMos with $L^g = 0.778$

The figures and tables relay similar information. Stochastic BiCosMos tends to be similar to deterministic BiCosMos for the core model reconstruction and outperforms deterministic BiCosMos for the iJR904 reconstruction. For both network reconstructions, stochastic BiCosMos nearly always outperforms CosMos by a significant margin. Similarly, stochastic MaxProd outperforms deterministic MaxProd, which in turn improves on CosMos. Stochastic BiCosMos significantly outperforms stochastic MaxProd when at least 300 scenarios are used in the stochastic model. Both models result in higher worst-case yield than their deterministic versions on average, though there is a chance of doing slightly worse than the deterministic version. We conclude that, if the implementation is performed many times, on average the decision maker will benefit from implementing a stochastic model with an appropriate choice of L^g. However, the deterministic version of BiCosMos has a high percentage of "Good" and "OK" scenarios and is much less computationally intensive. All models discussed in this work have the potential to significantly outperform CosMos when implementation uncertainty is present.

In summary, when compared with CosMos, all models discussed in this work produce solutions that result in significantly higher frequencies of outcomes where both yield and growth have a chance to be positive when uncertainty is present. The benefit of using a stochastic model versus a deterministic model is less obvious. It is clear that stochastic MaxProd is preferable to deterministic MaxProd with respect to concentration of scenarios with simultaneously high worst-case yield and best-case growth. The same conclusion, however, is not as obvious when comparing stochastic BiCosMos to deterministic BiCosMos. When using the iJR904 network reconstruction, higher values of L^g result in significantly higher average worst-case yield in stochastic BiCosMos than deterministic BiCosMos, but the core model reconstruction does not produce similar results. In fact, in the core model reconstruction, the deterministic and stochastic results are nearly indistinguishable, indicating that the value from using a stochastic model likely does not outweigh the added computational complexity.

Increasing Number of Bound Changes

We also conduct a computational study on several models with a maximum of 5 bound changes. We aim to discover if the solution is more sensitive to implementation uncertainty when more bounds are changed. We solve 10 instances of the *E. coli* core model and iJR904 reconstructions of stochastic BiCosMos with $L^g = 0.424$. We use 300 scenarios for the core model but only 100 scenarios for the iJR904 model, as the computational complexity increases significantly when the number of allowable bound changes increases. We compare the results for the deterministic CosMos and BiCosMos (with $\ell = 0.424$) models and two of the ten stochastic BiCosMos instances with $L^g = 0.424$. In Figure (4.7), we plot two-dimensional histograms of worst-case yield versus best-case growth on 10,000 randomly sampled scenarios for the three models with $k = 5$. The top four plots show the results for the core model reconstruction; the bottom four plots show the results for the iJR904 reconstruction.

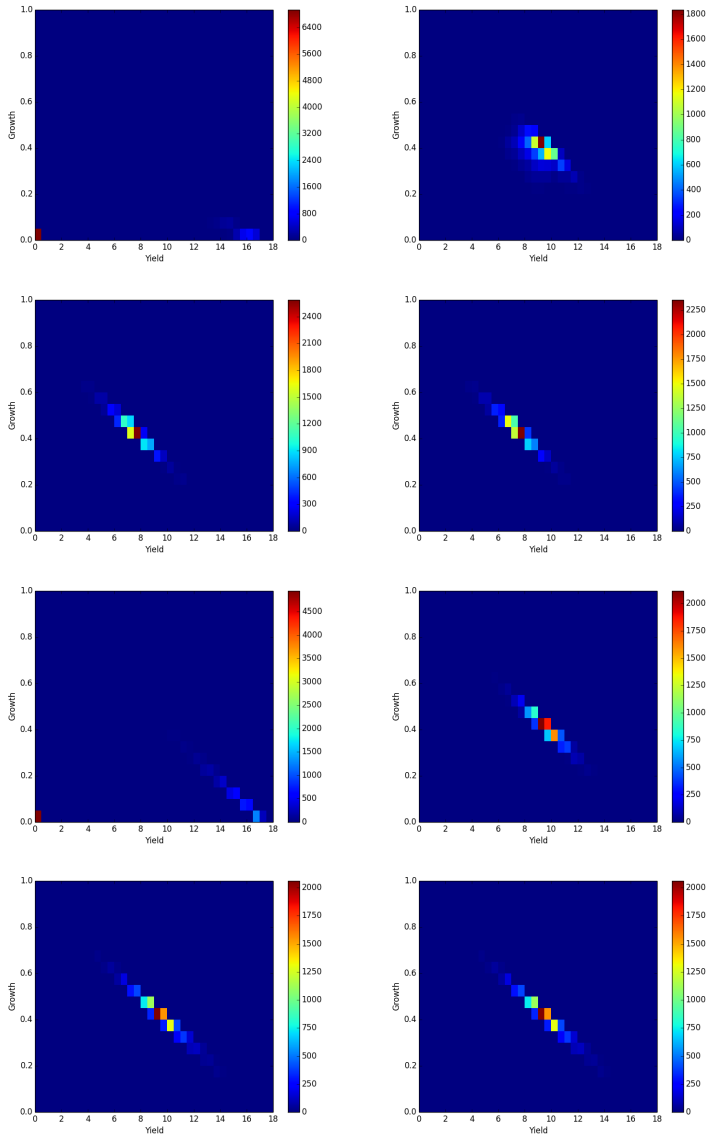


Figure 4.7: *E. coli* core mode (top four plots) and *E. coli* iJR904 (bottom four plots). Worst-case yield versus best-case growth on 10,000 samples of bound implementation success on solutions to the deterministic models and stochastic BiCosMos with $k = 5$. Order of figures is: CosMos, BiCosMos with $\ell = 0.424$, and two instances of stochastic BiCosMos with $L^g = 0.424$.

We first observe that CosMos is even more sensitive to bound-change uncertainty when more bounds are changed, which is to be expected. Similarly, for the core model, BiCosMos has a higher concentration of scenarios with growth less than 0.4 than when fewer bounds are changed. In fact, only 50% of the 10,000 scenarios have growth at least 0.4 and yield at least 4.0, indicating deterministic BiCosMos is slightly more sensitive to bound change

uncertainty with larger values of k. In contrast, across all replication of stochastic BiCosMos, the average percent of scenarios with growth at least 0.4 and yield at least 4.0 is about 83%; the minimum percentage is 76%, higher than the deterministic BiCosMos percentage.

In the iJR904 reconstruction, the results are similar but less pronounced. BiCosMos results in only 54% of the 10,000 scenarios with yield at least 4.0 and growth at least 0.4, whereas stochastic BiCosMos results in over 68% of the scenarios with at least that combination of yield and growth in all instances. Thus, we conclude that when uncertainty plays a larger role in the outcome (i.e., when more bounds are changed), stochastic BiCosMos should be used over deterministic BiCosMos. It is worth noting that deterministic BiCosMos is still a significant improvement over CosMos when $k = 5$.

4.8 Conclusion

We present several new extensions to the CosMos model of flux bound changes. The MaxProd model changes the objective function to be a model of cellular productivity, which encourages viable cells that produce high yield. The BiCosMos model optimizes worst-case yield and best-case growth in the top-level problem, encouraging solutions that are not as sensitive to implementation uncertainty as the solutions to CosMos and MaxProd.

We also propose stochastic extensions of both MaxProd and BiCosMos, explicitly incorporating an aspect of uncertainty within the models. We sample the fractional success of the planned bound changes to account for the uncertainty in the cell's response, thus finding solutions that account for some of the implementation uncertainty. The direct stochastic extension of BiCosMos produces solutions with undesirable behavior, so we present a slight modification in which we solve two auxiliary bi-objective problems that maximize approximate productivity and worst-case yield (or best-case growth). We show that in the deterministic setting, these auxiliary problems recover a subset of the Pareto

frontier of BiCosMos. In practice, these problems give good solutions in the stochastic setting as well.

We solve the deterministic bi-level MaxProd and BiCosMos models using MIP reformulations and commercial MIP solvers. However, the stochastic MaxProd and BiCosMos models are intractable with commercial MIP solvers. Thus, we propose use of a modified progressive hedging algorithm with a quadratic subproblem solved over the convex hull of a set of points that are feasible to the Lagrangian dual problem. Due to the budget constraint in BiCosMos that couples scenarios, we also present a budget allocation heuristic that allows BiCosMos to be fully decomposed by scenario. By alternating between the budget allocation heuristic and the modified progressive hedging algorithm, we find good feasible solutions to instances of the stochastic extension of BiCosMos. We conclude with computational results showing that solutions to the stochastic extensions of MaxProd and BiCosMos have the potential to be less sensitive to implementation uncertainty than their deterministic counterparts, particularly when more bounds are changed. However, the computational complexity of the stochastic models is great enough that in most cases, it is likely sufficient to solve the deterministic MaxProd or BiCosMos models. All models explored in this work are significantly less sensitive to bound change uncertainty than the deterministic CosMos model.

Future work could include an implementation of the solutions obtained from these models to investigate their usefulness in practice. It would also be valuable to investigate how other probability distributions of the bound change uncertainty affect the deterministic and stochastic solutions. Another route could be to explore ways to reduce the time required to run the heuristic algorithm, which would allow use of more scenarios in the stochastic models. Finally, it would be valuable to investigate other choices of stochastic objectives, such as minimizing risk.

A APPENDIX 1

A.1 Data used in experiments

Here we present the data used in the experiments described in Section 2.4. We use the same data for the experiments in both studies in Section 2.4, except for the maximum unloading and loading capacity values. Because the loading and unloading rates are nearly integer in Table A.1, the use of the additional capacity constraints does very little to alter the solution. Thus we use more fractional unloading and loading rates in Table A.2 to gain a better understanding of how the constraints affect the solution.

In these tables we give the parameter along with a brief description of how the parameter is used. We also give the range of values this parameter can take and the method we use to generate the parameter. If the parameter is generated by a "fixed" method, it is simply calculated as described in the "Range" column. "Uniform C" parameters are randomly generated from a continuous uniform distribution with the bounds given in the "Range" column. "Uniform I" parameters are randomly generated from an integer uniform distribution with the bounds given in the "Range" column.

[1]Mining locations in the mine are randomly generated on a circular map between 0.9 and 1.5 miles from the origin. Processing, stockpile, and dump sites are randomly generated on the map between 0.2 and 1.2 miles from the origin. The distance is calculated as the (fixed) Euclidean distance between two locations.
[1]

Symbol	Description	Range	Method
α_k	Size of truck type $k \in K$	[350, 420]	Uniform C
$\bar{p}_i$	Maximum number of trucks that can be processed per 30-sec time period at site $i \in N$	Mines: 0.99 trucks/minute; Processing and stockpiles: 1.0 trucks/minute; Dump: 2 trucks/minute	Fixed
Θ_{jt}^P	Amount that should be processed by processing site $j \in P$ during period t	[9000/30, 12000/30]	Uniform C
$weight_i$	Relative importance of mining site $i \in M$	[1, 3]	Uniform I
Θ_i^M	Amount that should be extracted from mine $i \in M$ during the planning horizon	num period$*weight_i$/sum($weight_i$ for $i \in M$)*[120, 160]	Uniform C
C_j	Buffer $j \in P$ maximum capacity (approximately an hour's worth of processing volume)	[9000, 12000]	Uniform C
I_{j0}	Amount in buffer $j \in P$ at time 0	[0, C_j]	Uniform C
$dist_{i,j}$	Distance between $i \in N$ and $j \in N$	See footnote[1]	Fixed
τ_{ij}	Number of periods needed to travel from i to j, $i, j \in N$	$\lceil dist_{i,j}/1.333 \rceil$	Fixed
Q_{jr}^P	Target grade-element ratio $r \in R$ of the ore at processing site $j \in P$	[0.05, 0.15]	Uniform C
Q_{ir}^M	Grade-element ratio $r \in R$ at mine or stockpile $i \in M \cup S$	[0.5 *Q_{jr}^P, 1.5*Q_{jr}^P]	Uniform C
Q_{jr0}^I	Grade-element ratio $r \in R$ of the ore in buffer $j \in P$ at time 0	[0.9*Q_{jr}^P, 1.1*Q_{jr}^P]	Uniform C
$\hat{x}_{ijkt}$	# of trucks of type k that left from $i \in N$ to $j \in N$ at time $t = -\tau_{ij}, ..., 0$	[0, randint(0,total trucks)]	Uniform I
$\hat{g}_{kj}$	# of trucks of type $k \in K$ available at $j \in N$ at $t = 0$	[0,randint(0, total trucks - total en route)]	Uniform I
$\hat{q^M}_{ik}$	# of trucks of type $k \in K$ waiting in a queue at mine $i \in M$ at time $t = 0$	[0,randint(0,total trucks - total en route - $\sum_{k \in K, j \in N} \hat{g}_{jk}$)]	Uniform I
$\hat{q^P}_{ijk}$	# of trucks of type $k \in K$ waiting in a queue at $j \in P \cup D$ holding load from mine $i \in M$ at time $t = 0$	[0,randint(0,total trucks - total en route - $\sum_{k \in K, j \in N} \hat{g}_{jk}$ - $\sum_{i \in M, k \in K} \hat{q^M}_{ik}$)]	Uniform I

Table A.1: All data used for the computational experiments described in Section 2.4

Symbol	Description	Range	Method
α_k	Size of truck type $k \in K$	[350, 420]	Uniform C
$\bar{p}_i$	Maximum number of trucks that can be processed per time period at site $i \in N$	Mines and processing: 0.5 trucks/minute; Dump: 2 trucks/minute; Stockpiles: 1 truck/minute	Fixed
Θ^P_{jt}	Amount that should be processed by processing site $j \in P$ during period t	[9000/30, 12000/30]	Uniform C
$weight_i$	Relative importance of mining site $i \in M$	[1, 3]	Uniform I
Θ^M_i	Amount that should be extracted from mine $i \in M$ during the planning horizon	num period$*weight_i$/sum($weight_i$ for $i \in M$)*[120, 160]	Uniform C
C_j	Buffer $j \in P$ maximum capacity (approximately an hour's worth of processing volume)	[9000, 12000]	Uniform C
I_{j0}	Amount in buffer $j \in P$ at time 0	[0, C_j]	Uniform C
$dist_{i,j}$	Distance between $i \in N$ and $j \in N$	See footnote[1]	Fixed
τ_{ij}	Number of periods needed to travel from i to j, $i, j \in N$	$\lceil dist_{i,j}/1.333 \rceil$	Fixed
Q^P_{jr}	Target grade-element ratio $r \in R$ of the ore at processing site $j \in P$	[0.05, 0.15]	Uniform C
Q^M_{ir}	Grade-element ratio $r \in R$ at mine or stockpile $i \in M \cup S$	[0.5 *Q^P_{jr}, 1.5*Q^P_{jr}]	Uniform C
Q^I_{jr0}	Grade-element ratio $r \in R$ of the ore in buffer $j \in P$ at time 0	[0.9*Q^P_{jr}, 1.1*Q^P_{jr}]	Uniform C
$\hat{x}_{ijkt}$	# of trucks of type k that left from $i \in N$ to $j \in N$ at time $t = -\tau_{ij}, ..., 0$	[0, randint(0,total trucks)]	Uniform I
$\hat{g}_{kj}$	# of trucks of type $k \in K$ available at $j \in N$ at $t = 0$	[0,randint(0, total trucks - total en route)]	Uniform I
$\hat{q^M}_{ik}$	# of trucks of type $k \in K$ waiting in a queue at mine $i \in M$ at time $t = 0$	[0,randint(0,total trucks - total en route - $\sum_{k \in K, j \in N} \hat{g}_{jk}$)]	Uniform I
$\hat{q^P}_{ijk}$	# of trucks of type $k \in K$ waiting in a queue at $j \in P \cup D$ holding load from mine $i \in M$ at time $t = 0$	[0,randint(0,total trucks - total en route - $\sum_{k \in K, j \in N} \hat{g}_{jk}$ - $\sum_{i \in M, k \in K} \hat{q^M}_{ik}$)]	Uniform I

Table A.2: All data used for the capacity constraint experiment described in Section 2.4

B APPENDIX 2

B.1 Extraction-based model of ore quality

Here we describe the model of true GER with respect to tons extracted from each mining site used in Section 3.4. We build an approximate function of GER versus tons extracted from each mine to roughly simulate the change in ore quality at a mining site as more ore is removed. Let T be the target tons extracted during the shift and M be the estimate of the average grade of ore extracted from the mine. There are many ways to model the change in ore quality with respect to tons extracted, but one potential model is a very simple piecewise linear function of tons extracted of the form

$$f(x) = \begin{cases} \frac{-2M}{T}x + 2M & \text{if } x \in [0, 0.5T] \\ \frac{2M}{T}x - M & \text{if } x \in (0.5T, T]. \end{cases}$$

Each time ore is extracted, we use this function to find the expected value of the GER, then generate a uniform random variate with bounds of +/- 0.1% times the expected value to represent the true GER of the ore taken from the mine. We choose this function for the key feature that the expected value of f is M.

B.2 Data used in experiments

Attachment *A2-SimData.xlsx* contains the data used for the computational experiments.

C APPENDIX 3

C.1 Distribution for Bound Change Uncertainty

Here we describe the probability distribution we use to generate the bound change uncertainty for the computational studies performed in Sections 4.4 and 4.7. The probability distribution used represents the fraction of the bound change that results from the attempted bound change and could be > 1. The true nature of the bound change success distribution is not well understood and relies in part on the magnitude of the bound change and which flux is selected to be changed. Because these factors are decision variables in our models, we build an approximation of the distribution that does not depend on knowledge of the bound changes. We assume that it is equally likely to overshoot a planned bound change and to undershoot a planned bound change. We further assume that the mean value is 1.0, meaning that over many implementations, the bound change will on average be implemented as planned.

For each flux $j \in J$, we define a Laplace distribution with a location parameter $\mu = 1$ and shape parameter $\lambda = 0.1$. The probability density function (pdf) of the Laplace distribution is as follows:

$$f(x) = \frac{1}{2\lambda} e^{\frac{-|x-\mu|}{\lambda}}.$$

The mean value of this function is μ. The Laplace distribution has non-zero probability for values of x from $-\infty$ to ∞. However, it is not reasonable for the bound change to have a negative or very large fraction of success. Thus, we must truncate the distribution both at 0 and a reasonable upper bound, such as 2.0. The truncated pdf requires definition of the

cumulative density function $F(x)$:

$$F(x) = \begin{cases} \frac{1}{2}e^{\frac{x-\mu}{\lambda}} & \text{if } x \leq \mu \\ 1 - \frac{1}{2}e^{\frac{-(x-\mu)}{\lambda}} & \text{if } x > \mu. \end{cases}$$

As with any truncated distribution, we defined the truncated pdf of the Laplace distribution as

$$f(x|0 \leq x \leq 2) = \frac{\frac{1}{2\lambda}e^{\frac{-|x-\mu|}{\lambda}}}{F(2) - F(0)} = \frac{\frac{1}{2\lambda}e^{\frac{-|x-\mu|}{\lambda}}}{(1 - 0.5e^{-10}) - 0.5e^{-10}}.$$

Thus, the pdf of the truncated Laplace can be written as

$$g(x) := \begin{cases} \frac{f(x)}{1-e^{-10}} & \text{if } 0 \leq x \leq 2 \\ 0 & \text{else.} \end{cases}$$

The mean value of $g(x)$ is also μ. In our computational experiments described in Sections 4.4 and 4.7, we sample the fractional bound change as $g(x)$.

C.2 Data used in computational studies

Attachments *A3-CoreModel.xlsx* and *A3-iJR904.xslx* contain the data used for the computational experiments.

REFERENCES

Ahangaran, D., A. Yasrebi, A. Wetherelt, and P. Foster. 2012. Real–time dispatching modelling for trucks with different capacities in open pit mines. *Archives of Mining Sciences* 57(1):39–52.

Arnold, MJ, and J Wm White. 1983. Computer-based truck dispatching. In *Proceeding of world mining congress, serbia*, 53–57.

Ataeepour, N, and EY Baafi. 1999. Arena simulation model for truck-shovel operation in despatching and non-despatching modes. *International Journal of Surface Mining, Reclamation and Environment* 13(3):125–129.

Bertsimas, Dimitris. 2014. Statistics and machine learning via a modern optimization lens. *INFORMS, The* 2015:1–42.

Bixby, Robert E. 2012. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica* 107–121.

Boland, Natashia, Jeffrey Christiansen, Brian Dandurand, Andrew Eberhard, Jeff Linderoth, James Luedtke, and Fabricio Oliveira. 2018. Combining Progressive Hedging with a Frank–Wolfe Method to Compute Lagrangian Dual Bounds in Stochastic Mixed-Integer Programming. *SIAM Journal on Optimization* 28(2):1312–1336.

Bonates, Eduardo Jorge Lira. 1992. The development of assignment procedures for semi-automated truckshovel systems.

Burgard, Anthony P, and Costas D Maranas. 2003. Optimization-Based Framework for Inferring and Testing Hypothesized Metabolic Objective Functions.

Burgard, Anthony P, Priti Pharkya, and Costas D Maranas. 2003. Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnology and bioengineering* 84(6):647–657.

Burt, Christina Naomi. 2008. An Optimisation Approach to Materials Handling in Surface Mines (August).

Chankong, Vira, and Yacov Y. Haimes. 1982. On the characterization of noninferior solutions of the vector optimization problem. *Automatica* 18(6):697–707.

Chatterjee, PK, and DJ Brake. 1981. Truck dispatching and simulation methods in open-pit operations. *CIM Bulletin* 74(835):102–107.

Choudhary, Ram Prasad. 2015. Optimization of Load – Haul – Dump Mining System By OEE and Match Factor for Surface Mining. *International Journal of Applied Engineering and Technology* 5(2):96–102.

Clark, P.A. 1990. Bilevel programming for steady-state chemical process design – ii. performance study for nondegenerate problems. *Computers and Chemical Engineering* 14(1):99–109.

Clark, P.A., and A.W. Westerberg. 1990. Bilevel programming for steady-state chemical process design – i. fundamentals and algorithms. *Computers and Chemical Engineering* 14(1):87–97.

Clark, Peter A., and Arthur W. Westerberg. 1983. Optimization for design problems having more than one objective. *Computers & Chemical Engineering* 7(4):259 – 278.

Colson, Benoît, Patrice Marcotte, and Gilles Savard. 2007. An overview of bilevel optimization. *Annals of operations research* 153(1):235–256.

Conejeros, Raul, and Vassilios S. Vassiliadis. 1998. Analysis and optimization of biochemical process reaction pathways. 2. optimal selection of reaction steps for modification. *Industrial & Engineering Chemistry Research* 37(12):4709–4714. `http://dx.doi.org/10.1021/ie980411c`.

Conforti, M., G. Cornuéjols, and G. Zambelli. 2014. *Integer programming*. Graduate Texts in Mathematics, Springer International Publishing.

Cotten, Cameron, and Jennifer L. Reed. 2013. Constraint-based strain design using continuous modifications (CosMos) of flux bounds finds new strategies for metabolic engineering. *Biotechnology Journal* 8(5):595–604.

Coverstone-Carroll, V., J.W. Hartmann, and W.J. Mason. 2000. Optimal multi-objective low-thrust spacecraft trajectories. *Computer Methods in Applied Mechanics and Engineering* 186(2):387–402.

Danna, Emilie, Edward Rothberg, and Claude Le Pape. 2005. Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming* 102(1):71–90.

Dempe, Stephan. 2002. *Foundations of bilevel programming*. Springer Science & Business Media.

Ehrgott, Matthias. 2006. *Multicriteria optimization*. Springer Science & Business Media.

Elbrond, J., and F. Soumis. 1987a. Towards integrated production planning and truck dispatching in open pit mines. *International Journal of Surface Mining, Reclamation and Environment* 1(1):1–6.

Elbrond, J, and F Soumis. 1987b. Towards integrated production planning and truck dispatching in open pit mines. *International Journal of Surface Mining, Reclamation and Environment* 1(1):1–6.

Ercelebi, S. G., and A. Bascetin. 2009. Optimization of shovel-truck system for surface mining. *Journal of the Southern African Institute of Mining and Metallurgy* 109(7):433–439.

Faraji, Razieh. 2013. A comparison between linear programming and simulation models for a dispatching system in open pit mines. Ph.D. thesis, University de Montreal.

Feist, Adam M, Daniel C Zielinski, Jeffrey D , Jan Schellenberger, J Markus, and Bernhard Ø Palsson. 2011. NIH Public Access 12(3):173–186.

Forsman, Bo, E Rönnkvist, and Nikos Vagenas. 1993. Truck dispatch computer simulation in aitik open pit mine. *International Journal of Surface Mining, Reclamation and Environment* 7(3):117–120.

Goodman, Gerrit VR, and Subhash C Sarin. 1988. Using mathematical programming to develop optimal overburden transport strategies in a surface coal mining operation. *International Journal of Surface Mining, Reclamation and Environment* 2(1):51–58.

Gurobi Optimization, LLC. 2018. Gurobi optimizer reference manual.

Gutjahr, Walter J, and Alois Pichler. 2016. Stochastic multi-objective optimization: a survey on non-scalarizing methods. *Annals of Operations Research* 236(2):475–499.

Halsall-Whitney, H., D. Taylor, and J. Thibault. 2003. Multicriteria optimization of gluconic acid production using net flow. *Bioprocess and Biosystems Engineering* 25(5):299–307.

Hatzimanikatis, V., C.A. Floudas, and J.E. Bailey. 1996a. Analysis and design of metabolic reaction networks via mixed-integer linear optimization. *AIChE Journal* 42(5):1277–1290.

Hatzimanikatis, Vassily, Christodoulos A. Floudas, and James E. Bailey. 1996b. Optimization of regulatory architectures in metabolic reaction networks. *Biotechnology and Bioengineering* 52(4):485–500.

Hauck, RF. 1973. A real-time dispatching algorithm for maximizing open-pit mine production under processing and blending requirements. In *Proceedings, seminar on scheduling in mining, smelting, and steelmaking*. Canadian Institute of Mining and Metallurgy.

———. 1979. Computer-controlled truck dispatching in open-pit mines. *Computer Methods for the* 80:735–742.

Jeroslow, Robert G. 1985. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming* 32(2):146–164.

Kall, Peter, Stein W Wallace, and Peter Kall. 1994. *Stochastic programming*. Springer.

Kappas, George, and Tunce M Yegulalp. 1991. An application of closed queueing networks theory in truck-shovel systems. *International Journal of Surface Mining, Reclamation and Environment* 5(1):45–51.

Kim, Joonhoon, and Jennifer L Reed. 2010. OptORF: Optimal metabolic and regulatory perturbations for metabolic engineering of microbial strains. *BMC systems biology* 4:53.

Kim, Joonhoon, Jennifer L. Reed, and Christos T. Maravelias. 2011. Large-Scale Bi-Level strain design approaches and Mixed-Integer programming solution techniques. *PLoS ONE* 6(9).

Labbé, Martine, Patrice Marcotte, and Gilles Savard. 1998. A bilevel model of taxation and its application to optimal highway pricing. *Management science* 44(12-part-1):1608–1622.

Labbé, Martine, and Alessia Violin. 2013. Bilevel programming and price setting problems. *4OR* 11(1):1–30.

Li, Z. 1990. A methodology for the optimum control of shovel and truck operations in open-pit mining. *Mining Science and Technology* 10(3):337–340.

Lizotte, Yves, and Eduardo Bonates. 1987. Truck and shovel dispatching rules assessment using simulation. *Mining Science and technology* 5(1):45–58.

Mak, Wai-Kei, David P Morton, and R Kevin Wood. 1999. Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations research letters* 24(1-2): 47–56.

McKenzie, Peter, Alexandra M Newman, and Luis Tenorio. 2008. Front range aggregates optimizes feeder movements at its quarry. *Interfaces* 38(6):436–447.

Mena, Rodrigo, Enrico Zio, Fredy Kristjanpoller, and Adolfo Arata. 2013. Availability-based simulation and optimization modeling framework for open-pit mine truck allocation under dynamic constraints. *International Journal of Mining Science and Technology* 23(1): 113–119.

Moreno, Eduardo, Mojtaba Rezakhah, Alexandra Newman, and Felipe Ferreira. 2017. Linear models for stockpiling in open-pit mine production scheduling problems. *European Journal of Operational Research* 260(1):212–221.

Munirathinam, Mohan, and Jon C Yingling. 1994. A review of computer-based truck dispatching strategies for surface mining operations. *International Journal of Surface Mining and Reclamation* 8(1):1–15.

Naoum, Shamil, and Ali Haidar. 2000. A hybrid knowledge base system and genetic algorithms for equipment selection. *Engineering Construction and Architectural Management* 7(1):3–14.

Oraee, Kazem, and Bahareh Asi. 2004. Fuzzy model for truck allocation in surface mines. In *Mine planning and equipment selection 2004, thirteenth international symposium on mine planning and equipment selection*, 585–593. Taylor & Fancis (Routledge, USA).

Orth, Jeffrey D, Ronan MT Fleming, and Bernhard O Palsson. 2010a. Reconstruction and use of microbial metabolic networks: the core escherichia coli metabolic model as an educational guide. *EcoSal plus*.

Orth, Jeffrey D, Ines Thiele, and Bernhard Ø Palsson. 2010b. What is flux balance analysis? *Nature biotechnology* 28(3):245–248.

Panagiotou, GN, and TN Michalakopoulos. 1995. Bedisp – a computer-based truck dispatching system for small-medium scale mining operations. *Mine planning and equipment selection* 481–486.

Petkov, Spas B, and Costas D Maranas. 1997. Quantitative Assessment of Uncertainty in the Optimization of Metabolic Pathways.

Pharkya, Priti, Anthony P Burgard, and Costas D Maranas. 2003. Exploring the Overproduction of Amino Acids Using the Bilevel Optimization Framework OptKnock.

———. 2004. OptStrain : A computational framework for redesign of microbial production systems (814):2367–2376.

Pharkya, Priti, and Costas D Maranas. 2006. An optimization framework for identifying reaction activation / inhibition or elimination candidates for overproduction in microbial systems 8:1–13.

Ranganathan, Sridhar, Patrick F Suthers, and Costas D Maranas. 2010. OptForce : An Optimization Procedure for Identifying All Genetic Manipulations Leading to Targeted Overproductions 6(4).

Reed, Jennifer L, Thuy D Vo, Christophe H Schilling, and Bernhard O Palsson. 2003. An expanded genome-scale model of escherichia coli k-12 (i jr904 gsm/gpr). *Genome biology* 4(9):R54.

Regan, L., I.D.L. Bogle, and P. Dunnill. 1993. Simulation and optimization of metabolic pathways. *Computers & Chemical Engineering* 17(5):627 – 637. European Symposium on Computer Aided Process Engineering 3-1.

Rubio, E. 2006. Mill feed optimization for multiple processing facilities using integer linear programming. *Proc. 15th Internat. Sympos. Mine Planning Equipment Selection (MPES)* 1206–1212.

Sadler, WM. 1988. Practical truck dispatch-a micro computer based approach. *Computer applications in the mineral industry* 495–500.

Schuster, S., and R. Heinrich. 1991. Minimization of intermediate concentrations as a suggested optimality principle for biochemical networks. *Journal of Mathematical Biology* 29(5):425–442.

Shapiro, Alexander, Darinka Dentcheva, and Andrzej Ruszczyński. 2009. *Lectures on stochastic programming: modeling and theory*. SIAM.

Smith, Martin L, and Stewart J Wicks. 2014. Medium-term Production Scheduling of the Lumwana Mining Complex using MIP with a Rolling Horizon. *Interfaces* 44(2):176–194.

Soumis, F, J Ethier, and J Elbrond. 1989. Truck dispatching in an open pit mine. *International Journal of Surface Mining, Reclamation and Environment* 3(2):115–119.

Stephanopoulos, G., A.A. Aristidou, and J. Nielsen. 1998. *Metabolic engineering: Principles and methodologies*. Elsevier Science.

Subtil, R F, D M Silva, and J C Alves. 2011. A Practical Approach to Truck Dispatch for Open Pit Mines. *APCOM Symposium* (September):765–777.

Ta, C. H., J. V. Kresta, J. F. Forbes, and H. J. Marquez. 2005. A stochastic optimization approach to mine truck allocation. *International Journal of Surface Mining, Reclamation and Environment* 19(3):162–175.

Temeng, Victor. 1998. A Nonpreemptive Goal Programming Approach to Truck Dispatching in Open Pit Mines. *Mineral resources engineering* 07(2):59–67.

Temeng, Victor A, Francis O Otuonye, and James O Frendewey Jr. 1997. Real-time truck dispatching using a transportation algorithm. *International Journal of Surface Mining, Reclamation and Environment* 11(4):203–207.

Tepper, Naama, and Tomer Shlomi. 2010. Predicting metabolic engineering knockout strategies for chemical production : accounting for competing pathways 26(4):536–543.

Topal, Erkan, and Salih Ramazan. 2012. Mining truck scheduling with stochastic maintenance cost. *Journal of Coal Science and Engineering* 18(3):313–319.

Torres, N V. 1994. Application of the transition time of metabolic system as a criterion for optimization of metabolic processes. *Biotechnology and bioengineering* 44(3).

Tu, JH, and VJ Hucka. 1985. Analysis of open-pit truck haulage system by use of a computer-model. *CiM Bulletin* 78(879):53–59.

Van Roy, Tony J. 1983. Cross decomposition for mixed integer programming. *Mathematical Programming* 25(1):46–63.

Vera, Julio, Pedro De Atauri, Marta Cascante, and V Torres. 2003. riteria Optimization of Biochemical Systems by Linear Programming : Application to Production of Ethanol by Saccharomyces cerevisiae.

Villaverde, Alejandro F, Sophia Bongard, Klaus Mauch, Eva Balsa-Canto, and Julio R Banga. 2016. Metabolic engineering with multi-objective optimization of kinetic models. *Journal of biotechnology* 222:1–8.

Voit, Eberhard O. 1992. Optimization in integrated biochemical systems. *Biotechnology and Bioengineering* 40(5):572–582.

Von Stackelberg, Heinrich. 1952. *The theory of the market economy*. Oxford University Press.

Weintraub, A, L Barros, A Magendzo, F Ibarra, C Ortiz, and G Rand. 1987. A truck dispatching system for a large open pit mine. *Operational Research* 87:650–662.

White, J Wm, MJ Arnold, and JG Clevenger. 1982. Automated open-pit truck dispatching at tyrone. *E&MJ-ENGINEERING AND MINING JOURNAL* 183(6):76–84.

White, J Wm, JP Olson, and SI Vohnout. 1993. On improving truck/shovel productivity in open pit mines. *CIM bulletin* 86:43–43.

White, James Wm. 1991. Computer Dispatch Aids Mine Productivity. *Australian Journal of Mining*.

Zeikus, J G. 1980. Chemical and fuel production by anaerobic bacteria. *Annual review of microbiology* 34.