Short-term scheduling of an open-pit mine with multiple objectives

Michelle Blom, Adrian R. Pearce and Peter J. Stuckey

Department of Computing and Information Systems, The University of Melbourne, Parkville, Australia

**ABSTRACT**

This article presents a novel algorithm for the generation of multiple short-term production schedules for an open-pit mine, in which several objectives, of varying priority, characterize the quality of each solution. A short-term schedule selects regions of a mine site, known as ‘blocks’, to be extracted in each week of a planning horizon (typically spanning 13 weeks). Existing tools for constructing these schedules use greedy heuristics, with little optimization. To construct a single schedule in which infrastructure is sufficiently utilized, with production grades consistently close to a desired target, a planner must often run these heuristics many times, adjusting parameters after each iteration. A planner’s intuition and experience can evaluate the relative quality and mineability of different schedules in a way that is difficult to automate. Of interest to a short-term planner is the generation of multiple schedules, extracting available ore and waste in varying sequences, which can then be manually compared. This article presents a tool in which multiple, diverse, short-term schedules are constructed, meeting a range of common objectives without the need for iterative parameter adjustment.

**ARTICLE HISTORY**

Received 23 June 2015
Accepted 22 July 2016

**KEYWORDS**

short-term open-pit mine production scheduling; multi-objective optimization; mixed-integer programming; decision support

1. Introduction

Open-pit mining is a commonly used method for the extraction of a range of metal ores, including iron, copper and gold. Horizontal layers of earth (known as *benches*) are extracted, from top to bottom, to form a stepped wall cavity (Hustrulid and Kuchta, 2006). The orebody of a mine is divided into a set of *blocks*. Each block is a region of earth with an estimated grade (percentage of metal in the block), quality (levels of contaminants) and tonnage. A short-term schedule selects a set of these blocks to be mined in each week of a (typically) 13-week horizon, and routes the mined material to a destination: a processing facility, a stockpile or a waste dump. Ore passing through each processing facility is blended (combined) to form a *product*. This article considers the short-term scheduling of a single open-pit mine, producing a single ore product with lower and upper bounds on its grade and quality. Equipment capacities must not be exceeded, all processing facilities must be utilized to their available capacity in each week, and mining precedences constrain the order in which blocks can be extracted.

This article presents a novel algorithm for the generation of multiple candidate solutions to this scheduling task, given a priority ordering over a set of supported objectives. Each candidate solution is a *schedule*—an allocation of material source (block or stockpile) to destination (stockpile or processing plant).
facility) movements to each period in the scheduling horizon. The quality of a schedule is characterized by its performance across a range of objectives. In the presence of hard constraints enforcing full use of available processing capacity, maintaining consistent production of ore whose grade lies within desired lower and/or upper bounds is usually the most important objective. Objectives may be defined to reduce spurious equipment moves, or maximize the alignment of each schedule to an over-arching longer-term plan. It may be important, for example, to avoid the mining of specific areas of the mine site (to be ‘saved’ for a later date), unless necessary to keep produced ore grades within bounds or to utilize the available processing facilities fully.

In prior work, Blom, Pearce and Stuckey (2016) present a rolling-horizon-based algorithm for the generation of a single short-term schedule at a single mine. This algorithm divides a horizon of $T$ periods into two aggregate periods of size 1 and $T-1$. The size of an aggregate refers to the number of periods it contains (or spans). A mixed integer program (MIP) models the short-term scheduling problem across this two-period horizon. Decision variables define which blocks are to be extracted in each period, and the destination of this material. Constraints place restrictions on the use of truck, dig unit and processing resources. Given a solution to this two-period MIP, the activities of the first period are fixed, and one rolls forward one period to consider the remaining horizon. A second two-period MIP, covering periods 2 and 3 to $T$, is solved, and the activities of period 2 fixed. This process continues until all $T$ periods are scheduled. This article presents a novel variation of this algorithm, in which multiple distinct schedules are generated, concurrently, each optimized with respect to an ordered sequence of objectives. This concurrent rolling-horizon-based (CRH) algorithm aggregates time periods into $N \geq 2$ groups, reflecting the concept of receding horizon control (Goodwin et al., 2006), solving a series of $N$-period MIP representations of the scheduling problem to generate each schedule. A split-and-branch approach (Section 5) is used to generate multiple schedules.

The CRH algorithm uses an optimize-and-prune approach to optimize each generated schedule with respect to multiple objectives. Given a sequence of objectives, ordered from highest to lowest priority, CRH considers each objective in turn when solving each $N$-period MIP. Each MIP is solved with respect to the first objective in this sequence. Constraints are then added to prune from the space of feasible solutions all those that are inferior to the obtained solution. The MIP is then re-solved with respect to the second objective in the sequence, and this process of pruning its feasible solution space repeated. The solution obtained to the MIP, after consideration of the final objective in the sequence, is used to fix the activities of the relevant period.

The CRH algorithm has been implemented in a short-term scheduling system, developed in conjunction with an industry partner. With this tool, a scheduler is able to create a variety of optimization scenarios, where each scenario is a sequence of available objectives to consider, listed from highest to lowest priority. The presented tool is able to generate multiple distinct short-term schedules for a mine site, each satisfying a set of relevant constraints (described in Section 3.3) within minutes. For each of these schedules, the order in which blocks are extracted, and the performance of each schedule with respect to the chosen objectives, is visualized in the form of tables, charts and maps.

The presented CRH scheduler is demonstrated in the short-term scheduling of an existing open-pit mine producing several million tons of ore per year. Two data sets have been provided by an industry partner for the purpose of demonstrating the tool’s capabilities. Each data set relates to a distinct point in time, describing the state of the mine and the blocks of ore and waste available for extraction. The time required by CRH to find sets of candidate schedules with respect to a variety of optimization scenarios, and the degree to which these schedules meet the specified objectives, is reported. This article describes a range of modelling and implementation choices, made in the design and construction of this tool, presenting practical advice for ensuring its efficiency.

The key contributions of this article are: an adaptation of the rolling-horizon-based approach of Blom, Pearce and Stuckey (2016) to generate concurrently multiple distinct short-term block extraction schedules via a novel split-and-branch algorithm; the application of an optimize-and-prune approach to optimize each schedule with respect to a sequence of priority-ordered objectives; and practical advice on how these approaches can be implemented efficiently. This advice, which outlines
key implementation choices designed to ensure efficient performance of the algorithms presented both in this article and in the work of Blom, Pearce and Stuckey (2016), is provided in an Online Supplement (http://dx.doi.org/10.1080/0305215X.2016.1218002).

The remainder of this article is structured as follows. Section 2 compares the presented work with existing literature on the scheduling of open-pit mines and multi-objective optimization. An MIP representation of the short-term scheduling problem is presented in Section 3. The CRH algorithm is described in two parts. First, an adaptation of the rolling-horizon-based algorithm of Blom, Pearce and Stuckey (2016) is presented, for which a series of MIPs (with time periods aggregated into $N \geq 2$ groups) are solved to generate a single schedule, optimized with respect to a sequence of objectives (Section 4). Using the components of this algorithm as building blocks, the complete CRH algorithm is presented in Section 5. The CRH algorithm is applied to an existing open-pit mine in Section 6.

2. Related work

Blom, Pearce and Stuckey (2016) present a decomposition-based algorithm for the short-term scheduling of multiple open-pit mines across multiple time periods. A key component of this algorithm is a method for generating multiple schedules for a number of individual mine sites. These schedules are formed by repeatedly solving an MIP with varying bounds on the grade of ore produced by the mine (i.e., blending constraints are modified between solves). The presented CRH algorithm generates multiple schedules with a split-and-branch approach (Section 5), where each schedule aims to produce ore within a fixed set of grade bounds. Both CRH and the algorithm of Blom, Pearce and Stuckey (2016) utilize a rolling-horizon-based approach, optimizing with respect to multiple objectives. Blom, Pearce and Stuckey (2016) consider only two objectives—the satisfaction of blending goals, and the maximization of mine productivity—while the tool presented in this article supports dozens of objectives. The modelling of mine operations in this article is more sophisticated than that of Blom, Pearce and Stuckey (2016). This article models the activities of multiple types of truck and dig unit. Blom, Pearce and Stuckey (2016) consider only a single trucking and extraction capacity (in tons). In practice, a fraction of the material sent to a plant, over the course of a week, is rehandled via a run-of-mine stockpile (ROM), using additional trucking resources. Dig unit productivity is reduced in regions of the mine that are difficult to excavate. Implicit rehandling and varying dig unit productivity are modelled in this article, but not by Blom, Pearce and Stuckey (2016).

The use of rolling-horizon-based methods for mine scheduling is not new. Goodwin et al. (2006) use receding horizon control to solve a long-term mine planning problem. Time is discretized into multiple periods of increasing size, with each consecutive period covering a larger timespan. The scheduling problem is solved with respect to the discretized horizon, and the activities of the first period fixed. The remaining periods are re-discretized, and the scheduling problem solved on the reduced horizon. This process continues until a schedule, whose time periods are uniformly discretized, is generated.

Cullenbine, Wood and Newman (2011) present a sliding time window heuristic (STWH) for solving a long-term mine planning problem, in which a sequence of integer programs (IPs) are generated and solved. In each IP, the full set of constraints is enforced in a window of $w$ periods, initially spanning periods 1 to $w$. A Lagrangian relaxation of the model is enforced outside of this window. The solution to this IP is used to fix the activities of period 1, after which the window slides forward by one period. The decision variables of the subsequent IP are fixed for period 1, subject to all problem constraints in the $w$ periods in the window, and a relaxation of the model thereafter. The STWH terminates once the last period in the horizon is scheduled. Lambert and Newman (2014) apply the STWH in the search for an initial feasible solution to an ultimate pit and block sequencing problem, while O’Sullivan and Newman (2015) apply the STWH to underground mine scheduling. The CRH algorithm enforces the full set of problem constraints across the whole horizon. Moreover, the focus is on the short-term scheduling of open-pit mines, whereas Cullenbine, Wood and Newman (2011) and Lambert and Newman (2014) focus on long-term planning.
Diverse approaches exist for solving mathematical programs in the presence of multiple objectives. A common goal in the solving of such programs is the generation of non-dominated solutions. Given a series of objectives, a solution is non-dominated if there does not exist an alternative that performs better relative to one objective without performing worse relative to another. Numerous reviews of such approaches exist, for example see Ehrgott and Gandibleux (2002), Ehrgott and Wiecek (2005), Chinchuluun and Pardalos (2007) and Alves and Clímaco (2007, 2009). In the present application, natural priority orderings exist over the objectives of interest. Given such an ordering over objectives, a solution is non-dominated if there does not exist an alternative that performs equally well in all objectives up to position $i$, but better against the objective in position $i+1$. The optimize-and-prune approach employed by CRH finds schedules that are, by this definition, non-dominated given a sequence of priority-ordered objectives. The existing literature on the scheduling of open-pit mines typically focuses on a narrow range of objectives: the maximization of net present value (relevant over long-term horizons); the minimization of costs; and the formation of correctly blended products. To the best of the present authors’ knowledge, no work exists in the mine scheduling literature in which multiple distinct schedules are generated with respect to a diverse range of customizable objectives.

Current tools used, in practice, for the short-term scheduling of open-pit mines typically apply greedy heuristics to generate a block extraction sequence. Similar to early work by Gershon (1987), these tools schedule each period in turn, with blocks selected for mining on the basis of their desirability according to a set of short- and long-term objectives. Achieving the correct blend of ore in the period being scheduled is a short-term objective. Gershon (1987) identifies a range of long-term objectives designed to increase the likelihood that the set of short-term objectives can be achieved in later periods. Encouraging the mining of material that will expose processable ore for later extraction is an example of such an objective. The planners that use these tools, however, find that many iterations of parameter changes, adjustment to objective weights, rule changes (restricting the movement of material along various source to destination pathways), and precedence additions (forcing certain blocks to be extracted before others), are required to create a satisfactory schedule. The approach presented in this article more reliably supports the satisfaction of objectives across a horizon. It does so by capturing future decisions and constraints, over a coarse discretization of the remaining horizon, when it schedules the activity of each period. Each decision that is made in period $t$ is subject to the performance of objectives on a coarse approximation of the remaining $T-t$ periods.

The scheduling of mines over longer-term horizons (typically spanning decades) has been well studied in the literature—see Osanloo, Gholamnejad and Karimi (2008), Newman et al. (2010), Askari-Nasab et al. (2011), Espinoza et al. (2012) and Lambert et al. (2014) for reviews. By contrast, short-term scheduling has received little attention. The key point of difference between the short-term problem presented in this article, relative to scheduling over longer horizons, is the level of detail at which mine operations are modelled. Consequently, there are significant differences in the decision variables and constraints that define the short- and long-term scheduling problems—in particular, dig unit and truck behaviour and mining precedences are modelled in greater detail in the short-term.

3. Modelling the mining environment

A mine is modelled in terms of a set of blocks, $B$, each with a defined geometry and composition. Each block contains one or more different types of material (e.g. high grade, low grade and waste). To access a block $b$, at least one set of blocks in its precedence list $A_b$ must be extracted entirely. For example, consider the arrangement of blocks in Figure 1. Block $b_0$ can be accessed by removing one of two sets of blocks, $A_{b_0} = \{\{b_1, b_2\}, \{b_3, b_4\}\}$. The modelling of precedences in terms of a disjunction of block groups differs from much of the existing literature on scheduling for open-pit mines, in which each $b \in B$ is assigned a single set of blocks, all of which must be extracted before $b$ can be accessed. Modelling mining precedences in terms of groups-of-sets is not new, however. O’Sullivan and Newman (2015) consider an underground mining scenario in which activities (such as extraction and backfilling) are typically dependent on a number of sequences of activities (where each sequence
can be progressed concurrently alongside the others) being performed beforehand. The advantage of disjunctive precedences is that the direction by which each block is approached is not fixed a priori, affording more flexibility in how the orebody is mined. Schedule generation is more time consuming, however, when disjunctive precedences are used, as shown in Section 6.

Material is excavated from each block by a dig unit, and then hauled, by truck, to a destination: a stockpile \( s \in S \) (including one or more waste dumps); or a processing plant \( p \in P \). Material on a stockpile can be extracted, by a dig unit, and hauled, by truck, to a plant. Ore produced by each plant \( p \in P \) is uniformly blended to form a product (uniform blending is assumed in the scheduling of open-pit mines). Multiple types of dig unit and truck are used at the mine. This article models the operation of plants in more detail than usually present in the literature. Each plant \( p \in P \) is assigned a rehandling and yield percentage, \( R_p \) and \( Y_p \). A fraction \( R_p \) of the total tons of material sent to plant \( p \), from blocks in \( B \), will be implicitly rehandled via a run-of-mine stockpile, consuming both truck and dig unit resources. This means that a fraction \( (R_p) \) of the material sent to plant \( p \) will be re-directed to a ROM, and then extracted from this ROM and sent to plant \( p \). ROMs are located close to each plant, but are not explicitly modelled (i.e. the movement of material to and from each ROM is not scheduled). Each plant \( p \in P \) yields a fraction \( Y_p \) of its input feed (\( p \) produces \( xY_p \) tons of ore given an input feed of \( x \) tons).

This article considers the short-term scheduling of an open-pit mine producing a single ore product whose composition is defined in terms of the percentage of a range of attributes \( q \in Q \) (metal grade and contaminants), in each period of a \( T \)-period horizon. A solution is a schedule stating the tons of material to be extracted from each source \( s \in S \cup B \), in each period, and the destination \( d \in S \cup P \) to which it will be sent.

In this section, an MIP model, \( MIP_{STP} \), of the short-term scheduling problem for a single open-pit mine, producing a single ore product, is presented. This MIP is repeatedly solved, as described in Section 4, over the course of a rolling-horizon-based algorithm. Instances of \( MIP_{STP} \) are solved over a horizon that reduces in size over the course of the algorithm (i.e. from \( 1, \ldots, T \) to \( 2, \ldots, T \), and so on until the horizon spans just one period \( T \)). To reduce the complexity of each solve, \( MIP_{STP} \) is defined on a discretization of the \( T \)-period horizon in which time periods are aggregated into groups of non-uniform and increasing size (with the first aggregate containing only one period). A procedure for performing this aggregation is provided in the Online Supplement (http://dx.doi.org/10.1080/0305215X.2016.1218002). For example, a horizon of thirteen periods can be discretized into three period-aggregates as follows:

- **Original horizon**: \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13\}
- **Discretized horizon**: \{\{1\}, \{2, 3, 4, 5\}, \{6, 7, 8, 9, 10, 11, 12, 13\}\}.
3.1. Notation

This section defines the indices and sets, functions, parameters, and decision variables that characterize the mining environment.

Indices and sets

\( t \in T \) Index over time period-aggregates in the discretized horizon \( T \).

\( b \in B \) Set of blocks in the block model of the mine.

\( l \in L \) Set of modelled material types (e.g. high grade, low grade, waste).

\( s \in S \) Set of stockpiles available at the mine.

\( p \in P \) Set of plants (processing options) available at the mine.

\( q \in Q \) Attributes defining the composition of ore and waste.

\( \tau \in R^k \) Set of truck types operating at the mine site.

\( \delta \in R^d \) Set of dig unit types operating at the mine site.

\( k \in A_b \) Each \( k \) is a group of blocks that, if removed, will provide access to block \( b \in B \). There can be multiple ways of accessing \( b \), by removing different groups of blocks (\( A_b \) defines the different ways in which \( b \) can be accessed).

\( j \in \hat{B} \) Each \( j \) is a group of blocks, each group defining a region of the mine site.

Functions

\( \text{size}(t) \) Returns the number of time periods in period-aggregate \( t \in T \).

\( \text{periods}(t) \) Returns the set of time periods in period-aggregate \( t \in T \).

\( \text{blocks}(j) \) Returns the set of blocks in region \( j \in \hat{B} \) of the mine.

\( \text{blocks}(k) \) Returns the set of blocks in group \( k \in A_b \) for some \( b \in B \).

\( \text{nbors}(B') \) Returns the set of blocks outside of \( B' \subseteq B \) that are a dependency (according to mining precedences) of some block inside \( B' \).

Parameters

\( T_{bl1} \) Tons of material \( l \in L \) in block \( b \in B \) at the start of the horizon \( t = 1 \).

\( G_{blq} \) Fractional percentage of \( q \in Q \) in the material of type \( l \in L \) in \( b \in B \).

\( G_{blq} \) Fractional percentage of \( q \in Q \) in the material of type \( l \in L \) in \( b \in B \) after it has been through processing in plant \( p \in P \).

\( T_{s1} \) Size of stockpile \( s \in S \) (in tons) at the start of \( t = 1 \) (material movement restrictions ensure that all stockpiles contain only one material type \( l \in L \)).

\( G_{sq1} \) Fractional percentage of attribute \( q \in Q \) in stockpile \( s \in S \) \( (t = 1) \).

\( G_{sq1p} \) Fractional percentage of attribute \( q \in Q \) in stockpile \( s \in S \) \( (t = 1) \) after it has been through processing in plant \( p \in P \).

\( C_s \) Capacity (tons) of stockpile \( s \in S \).

\( C_{pt} \) Capacity (tons) of plant \( p \in P \) in period-aggregate \( t \in T \).

\( C_{\delta t} \) Excavation capacity (tons) of dig unit type \( \delta \in R^d \) in \( t \in T \).

\( C_{\tau t} \) Available capacity (minutes) of truck type \( \tau \in R^k \) in \( t \in T \).

\( L_t \) Load size (tons) of truck type \( \tau \in R^k \).

\( \Theta_{xdt} \) Minutes required, by truck type \( \tau \in R^k \), to travel from material source \( s \in S \cup B \) to destination \( d \in S \cup P \) and back to \( s \) (cycle time).

\( \tilde{\Theta}_{jdt} \) Average minutes required, by truck type \( \tau \in R^k \), to travel from a block in region \( j \in \hat{B} \) to destination \( d \in S \cup P \) and back to \( j \) (cycle time).
\( \Theta_{pt}^R \) Cycle time (in minutes) assumed for the implicit rehandling of each truckload of material sent to plant \( p \), from the mine deposit \( i.e. \) from blocks and not stockpiles) with truck type \( \tau \in \mathcal{R}^k \).

\( \gamma_b \) ‘Dig factor’ assigned to block \( b \in \mathcal{B} \), \( \gamma_b \geq 1 \), expressing difficulty of mining—the tons of dig unit capacity used by mining \( x \) tons of \( b \) is \( \gamma_b x \).

\( R_p \) Rehandling percentage associated with plant \( p \in \mathcal{P} \).

\( \gamma_b \) ‘Dig factor’ assigned to block \( b \in \mathcal{B} \), \( \gamma_b \geq 1 \), expressing difficulty of mining—the tons of dig unit capacity used by mining \( x \) tons of \( b \) is \( \gamma_b x \).

\( R_p \) Rehandling percentage associated with plant \( p \in \mathcal{P} \).

\( Y_p \) Yield of plant \( p \in \mathcal{P} \).

\( \bar{P}_{jt} \) Maximum number of blocks in region \( j \in \hat{\mathcal{B}} \) allowed to remain in a ‘partially mined’ state at the end of \( t \in \mathcal{T} \).

### Decision variables

\( x_{bldt} \) Tons of material \( l \in \mathcal{L} \) in block \( b \in \mathcal{B} \) hauled to destination \( d \in \mathcal{S} \cup \mathcal{P} \) in \( t \in \mathcal{T} \).

\( x_{sprt} \) Tons of stockpile \( s \in \mathcal{S} \) hauled to plant \( p \in \mathcal{P} \) with truck type \( \tau \in \mathcal{R}^k \) in \( t \in \mathcal{T} \).

\( y_{bt} \) Binary of value 1 iff block \( b \in \mathcal{B} \) is mined by the end of \( t \in \mathcal{T} \).

\( y_{blt} \) Binary of value 1 iff material \( l \in \mathcal{L} \) from \( b \in \mathcal{B} \) is mined by the end of \( t \in \mathcal{T} \).

\( y_{bt} \) Binary of value 1 iff block \( b \in \mathcal{B} \) has been completely extracted by the end of \( t \in \mathcal{T} \).

\( a_{st} \) Tons of material on stockpile \( s \in \mathcal{S} \) at the end of \( t \in \mathcal{T} \).

\( y_{kt} \) Binary of value 1 iff the block group \( k \in \bigcup_{b \in \mathcal{B}} \mathcal{A}_b \) has been completely extracted by the end of \( t \in \mathcal{T} \).

\( u_{s\delta t} \) The tons of stockpile \( s \in \mathcal{S} \) extracted by dig unit \( \delta \in \mathcal{R}^d \) in \( t \in \mathcal{T} \).

In each \( t \in \mathcal{T} \), a fraction \( R_p \) of the material sent to a plant \( p \in \mathcal{P} \), from blocks \( \mathcal{B} \), is diverted to an unmodelled ROM. In the same period, a dig unit extracts this material from the ROM, and trucking resources haul this material to plant \( p \). This implicit rehandling is unscheduled in the sense that which blocks are to be diverted to an ROM are not specified. Variables are created for each truck and dig unit type to denote the total tons of material they haul, and extract, in the process of implicit rehandling.

\( r_{ptt} \) Tons of material hauled to plant \( p \in \mathcal{P} \) by truck type \( \tau \in \mathcal{R}^k \), via implicit rehandling at an unmodelled ROM, in \( t \in \mathcal{T} \).

\( r_{p\delta t} \) Tons of material extracted from an unmodelled ROM (due to implicit rehandling) by dig unit \( \delta \in \mathcal{R}^d \) and hauled to plant \( p \in \mathcal{P} \), in \( t \in \mathcal{T} \).

Variables are defined to model the tons of material extracted from each region \( j \) by each dig unit type \( \delta \), \( e_{j\delta t} \), and transported to a destination by each truck type \( \tau \), \( e_{j\tau t} \).

\( e_{j\delta t} \) Tons extracted from region \( j \in \hat{\mathcal{B}} \) and sent to destination \( d \in \mathcal{S} \cup \mathcal{P} \) with truck type \( \tau \in \mathcal{R}^k \), in \( t \in \mathcal{T} \).

\( e_{j\delta t} \) Tons extracted from region \( j \in \hat{\mathcal{B}} \), in \( t \in \mathcal{T} \), with dig unit type \( \delta \in \mathcal{R}^d \).

### 3.2. Objectives

The MIPSTP model is solved with a variety of objectives using an optimize-and-prune approach, described in Section 4. Each objective is defined in terms of an expression to be optimized, and a set of associated prune constraints, as per Definition 3.1.

**Definition 3.1** ([—Objective]). Each objective \( z \in \mathcal{Z} \) is defined as a tuple:

\[
\begin{align*}
    z = (\xi(z), PR(z)),
\end{align*}
\]
where $\xi(z)$ is an expression to be optimized ($\xi(z) = \max \psi$ or $\xi(z) = \min \psi$) over a horizon and $PR(z)$ a set of prune constraints. Let $x^z$ denote the optimal solution of $\text{MIP}_STP$ when solved with respect to $z$. After solving $\text{MIP}_STP$ with objective $z$, constraints $PR(z)$ are added to the model, discarding from its feasible solution space all solutions that are inferior to $x^z$ with respect to $z$. The general form of $PR(z)$ is shown in Equation (2).

\[ PR(z) = \begin{cases} 
\psi \geq \psi(x^z) & \text{if } \xi(z) = \max \psi \\
\psi \leq \psi(x^z) & \text{if } \xi(z) = \min \psi.
\end{cases} \tag{2} \]

The presented solver supports over 40 objectives, including: maximizing the utilization of trucks; maintaining desired material movement between selected regions and stockpiles; maintaining bounds on the tons of waste mined in selected regions; and maintaining the grade of stockpile and plant feed within desired bounds. In Section 6, the presented solver is demonstrated in a range of optimization scenarios, composed of a variety of commonly optimized objectives. Two of these objectives are formally defined in this section. Definitions for each of these objectives are provided in the Online Supplement (http://dx.doi.org/10.1080/0305215X.2016.1218002).

### 3.2.1. Maintaining alignment with a longer-term plan ($z_{\text{align}}$)
A longer-term plan identifies the blocks $B_{LTP} \subset B$ that should be extracted by the end of the horizon. This objective aims to maximize the tons of material extracted from blocks in this set. The prune constraints for $z_{\text{align}}$ take the form shown in Equation (2).

\[ \xi(z_{\text{align}}) \equiv \max \sum_{b \in B_{LTP}} \sum_{t \in T} \sum_{l \in L} \sum_{d \in \mathcal{S} \cup \mathcal{P}} x_{bldt}. \tag{3} \]

### 3.2.2. Maintaining product grades within desired bounds ($z_{\text{grade}}$)
This objective aims to keep the levels of each attribute $q \in Q$ in the ore produced by the mine in each period-aggregate $t \in T$ within desired lower ($L_{qt}$) and upper ($\bar{L}_{qt}$) bounds. Let $h_{pt}$ and $\rho_{qt}$ denote the tons of material sent to plant $p \in \mathcal{P}$ in $t$, and the tons of attribute $q$ produced by the mine in $t$, respectively.

\[ \xi(z_{\text{grade}}) \equiv \min \psi_{\text{grade}}, \tag{4} \]

where

\[ \psi_{\text{grade}} = \sum_{q \in Q} \sum_{t \in T} \frac{1}{\text{size}(t)} \max \left[ 0, \rho_{qt} - \bar{L}_{qt} \sum_{p \in \mathcal{P}} Y_p h_{pt} \right] \]

\[ + \sum_{q \in Q} \sum_{t \in T} \frac{1}{\text{size}(t)} \max \left[ 0, L_{qt} \sum_{p \in \mathcal{P}} Y_p h_{pt} - \rho_{qt} \right] \]  \tag{5} \]

\[ h_{pt} = \sum_{\tau \in \mathcal{R}_k} \sum_{s \in \mathcal{S}} x_{sp\tau t} + \sum_{b \in B} \sum_{l \in L} x_{blpt} \forall t \in T, \ p \in \mathcal{P}, \] \tag{6} \]

\[ \rho_{qt} = \sum_{p \in \mathcal{P}} Y_p \left[ \sum_{\tau \in \mathcal{R}_k} \sum_{s \in \mathcal{S}} G_{sq1p} x_{sp\tau t} + \sum_{b \in B} \sum_{l \in L} G_{blqp} x_{blpt} \right] \forall t \in T, \ q \in Q. \] \tag{7} \]

Let $v$ denote the value of $\psi_{\text{grade}}$ in the best found solution to $\text{MIP}_STP$, instantiated with $\xi(z_{\text{grade}})$. The prune constraints for $z_{\text{grade}}$ are shown in Equation (8):

\[ PR(z_{\text{grade}}) \equiv \psi_{\text{grade}} \leq v. \tag{8} \]
Several variations of $z_{\text{grade}}$ are supported in which grade deviation above desired upper bounds, below desired lower bounds, or from the original bounds narrowed by one half, and three quarters, are minimized. Also supported are objectives that minimize deviation between the level of each $q \in Q$, individually, and its original or narrowed bounds.

### 3.3. Constraints

Constraints (9)–(11) restrict the tons of material extracted from each block $b \in B$ to what initially exists, $T_{bl} t$ for $l \in L$, and define the binary indicators $y_{bt}, y_{bl}$ and $\hat{y}_{bt}$.

All variables denoting the movement of material along an invalid path (e.g. material of type $l \in L$ in block $b \in B$ to a stockpile or plant that cannot accept that material) assume a value of zero. The presented scheduler supports restrictions on the regions of a mine in which truck and dig unit types are allowed to operate. All variables denoting an invalid use of a truck or dig unit type (e.g. the extraction of material from a set of blocks by a dig unit not permitted to access those blocks) assume a value of zero.

\[
y_{bt} \geq y_{blt} \quad \forall \ t \in T, b \in B, \ l \in L,
\]

\[
\sum_{t=1}^{t} \sum_{l \in L} x_{blt} \geq T_{bl} \hat{y}_{bt} \quad \forall \ t \in T, \ b \in B,
\]

\[
\sum_{t=1}^{t} \sum_{l \in L} x_{blt} \leq T_{bl} y_{blt} \quad \forall \ t \in T, \ b \in B, \ l \in L.
\]

Constraint (12) sets binary $y_{kt}$ to 1 if and only if all blocks in group $k \in \bigcup_{b \in B} A_b$ have been completely mined by the end of period-aggregate $t \in T$ ($\hat{y}_{bt}$ equals one for every block in group $k$). Constraint (13) prevents block $b$ from being mined in $t$, unless at least one group $k \in A_b$ is completely mined by the end of $t$. In long-term scheduling problems, each block is accessed by extracting the 5 or 9 blocks above it. In the short-term, however, precedences exist between each block and those that lie both above and next to it.

\[
|\text{blocks}(k)| \ y_{kt} \leq \sum_{b' \in \text{blocks}(k)} \hat{y}_{bt} \quad \forall \ t \in T, \ k \in \bigcup_{b \in B} A_b,
\]

\[
\sum_{k \in A_b} y_{kt} \geq y_{bt} \quad \forall \ t \in T, \ b \in B.
\]

Separation constraints are defined to enforce mining precedences between blocks. Blom, Pearce and Stuckey (2016) provide a set of such constraints, which is replicated here. A block $b \in B$ is immediately accessible if $|A_b| = 0$ (i.e. no other blocks must be removed before $b$ can be accessed). The set $P'$ contains all contiguous groups of blocks $B' \subset B$ for which $\neg \exists b' \in B'$ such that $|A_{b'}| = 0$ (i.e. no block inside $B'$ is immediately accessible). Let $\text{nbors}(B')$ return the set of blocks $\{b|b' \in B', k \in A_{b'}, b \in \text{blocks}(k)\}$ (i.e. the ‘neighbours’ of the blocks in $B'$). In the process of solving MIP$_{STP}$, selected instances of constraint (14) are added via a separation algorithm, described in the Online Supplement (http://dx.doi.org/10.1080/0305215X.2016.1218002). This separation algorithm is the same as that used by Blom, Pearce and Stuckey (2016), but does not appear in that work for space reasons. Each contiguous set of blocks mined in $t \in T$ (maximal with respect to set inclusion) must contain a block that is immediately accessible in $t$. This prevents the mining of groups of blocks, each supporting the mining of the other in constraint (12), but neither being immediately accessible. Figure 2 illustrates this, with blocks $b_0$ and $b_1$ scheduled to be mined, despite being surrounded by blocks that are not scheduled for extraction. Constraint (14) states that if a block in $B'$ is mined in $t \in T$, then at least
one block in $\text{nbors}(B')$ must have been entirely extracted by the end of $t$:

$$\sum_{b' \in \text{nbors}(B')} \hat{y}_{b't} \geq \frac{1}{|B'|} \sum_{b' \in B'} y_{b't} \quad \forall \ t \in T, \quad B' \in P'. \quad (14)$$

Constraints (15) and (16) distribute the work involved in the implicit rehandling of material via (unmodelled) ROMs across truck and dig unit types. Constraint (17) distributes the work involved in extracting material from stockpiles across dig unit types:

$$\sum_{\delta \in R^d} r_{\delta t} = R_p \sum_{b \in B} \sum_{l \in L} x_{bltp} \quad \forall \ t \in T, \quad p \in P, \quad (15)$$

$$\sum_{\delta \in R^d} r_{\delta t} = R_p \sum_{b \in B} \sum_{l \in L} x_{bltp} \quad \forall \ t \in T, \quad p \in P, \quad (16)$$

$$\sum_{\delta \in R^d} u_{s\delta t} = \sum_{p \in P} \sum_{\tau \in R^k} x_{sp\tau t} \quad \forall \ t \in T, \quad s \in S. \quad (17)$$

Recall that $e_{\delta t}$ denotes the tons of material extracted from region $j \in \hat{B}$, in $t \in T$, by dig unit type $\delta \in R^d$, and $e_{\delta dt}$ the tons of material extracted from region $j \in \hat{B}$ and sent to destination $d \in S \cup P$, in $t \in T$, by truck type $\tau$:

$$\sum_{\delta \in R^d} e_{\delta t} = \sum_{b \in \text{blocks}(j)} \sum_{l \in L} \sum_{d \in S \cup P} \gamma_b x_{bldt} \quad \forall \ t \in T, \quad j \in \hat{B}, \quad (18)$$

$$\sum_{\tau \in R^k} e_{\delta dt} = \sum_{b \in \text{blocks}(j)} \sum_{l \in L} x_{bldt} \quad \forall \ t \in T, \quad j \in \hat{B}, \quad d \in S \cup P. \quad (19)$$

The capacities of dig unit and truck types, in each period, are enforced by constraints (20) and (21). Parameter $\Theta_{j\delta\tau}$ denotes the average cycle time (in minutes), for truck type $\tau \in R^k$, to haul a truckload of material from a block in region $j \in \hat{B}$ to destination $d \in S \cup P$. Recall that $u_{\delta t}$ denotes the tons of material extracted from stockpile $s \in S$ by dig unit type $\delta \in R^d$ in period-aggregate $t \in T$. The capacity of each dig unit type $\delta$ and truck type $\tau$, in $t \in T$, are respectively denoted by $C_{\delta t}$ (in tons).
and \( C_{\tau t} \) (in minutes):

\[
\sum_{j \in \bar{B}} e_{j\delta t} + \sum_{s \in S} u_{s\delta t} + \sum_{p \in \mathcal{P}} r_{p\delta t} \leq C_{\delta t} \quad \forall \ t \in \mathcal{T}, \delta \in \mathcal{R}^d, \tag{20}
\]

\[
\sum_{p \in \mathcal{P}} \left[ \Theta^R_{p\tau t} r_{p\tau t} + \sum_{s \in S} \Theta^s_{sp\tau t} x_{sp\tau t} \right] + \sum_{j \in \bar{B}} \sum_{d \in S \cup \mathcal{P}} \tilde{\Theta}_{jdt} e_{jdt} \leq C_{\tau t} \quad \forall \ t \in \mathcal{T}, \tau \in \mathcal{R}^k. \tag{21}
\]

Constraints (22)–(24) ensure stockpile capacities are respected in each period-aggregate \( t \in \mathcal{T} \). Recall that \( a_{s t} \) denotes the tons of material on stockpile \( s \in S \) at the end of \( t \in \mathcal{T} \):

\[
a_{s1} = T_{s1} + \sum_{b \in \bar{B}} \sum_{l \in \mathcal{L}} x_{bls1} - \sum_{\tau \in \mathcal{R}^k} \sum_{p \in \mathcal{P}} x_{sp\tau t} \quad \forall \ s \in S, \tag{22}
\]

\[
a_{st} = a_{s,t-1} + \sum_{b \in \bar{B}} \sum_{l \in \mathcal{L}} x_{blst} - \sum_{\tau \in \mathcal{R}^k} \sum_{p \in \mathcal{P}} x_{sp\tau t} \quad \forall \ t \in \mathcal{T}, t > 1, s \in S, \tag{23}
\]

\[
0 \leq a_{st} \leq C_s \quad \forall \ t \in \mathcal{T}, s \in S. \tag{24}
\]

Selected stockpiles \( S' \subseteq S \) can have material deposited on, or reclaimed from, them but not both, in the same time period. Let \( v_{st} \) and \( w_{st} \) denote binary variables, one for each \( t \in \mathcal{T} \) and stockpile \( s \in S' \) (\( v_{st} \) takes on a value of one if material is reclaimed from \( s \) in \( t \), and \( w_{st} \) takes on a value of one if material is deposited onto \( s \) in \( t \)). Constraints (25)–(27) enforce this restriction on each \( s \in S' \). Reclaiming and depositing material from and on a stockpile in the same period allows the MIP to ‘cheat’ by sending material of low quality to the stockpile and reclaiming it at the ‘average’ stockpile grade. Stockpiles are not restricted in this way in an aggregate spanning multiple periods, as in this case it is natural for both depositing and reclaiming to take place.

\[
\sum_{\tau \in \mathcal{R}^k} \sum_{p \in \mathcal{P}} x_{sp\tau t} \leq C_s v_{st} \quad \forall \ t \in \mathcal{T}, \text{size}(t) = 1, s \in S', \tag{25}
\]

\[
\sum_{b \in \bar{B}} \sum_{l \in \mathcal{L}} x_{blst} \leq C_s w_{st} \quad \forall \ t \in \mathcal{T}, \text{size}(t) = 1, s \in S', \tag{26}
\]

\[
v_{st} + w_{st} \leq 1 \quad \forall \ t \in \mathcal{T}, \text{size}(t) = 1, s \in S'. \tag{27}
\]

Each plant \( p \in \mathcal{P} \) must be supplied an amount of material, in each \( t \in \mathcal{T} \), equal to its capacity \( C_{pt} \):

\[
\sum_{\tau \in \mathcal{R}^k} \sum_{s \in S} x_{sp\tau t} + \sum_{b \in \bar{B}} \sum_{l \in \mathcal{L}} x_{blpt} = C_{pt} \quad \forall \ t \in \mathcal{T}, p \in \mathcal{P}. \tag{28}
\]

A practical schedule mines blocks from start to finish in a continuous manner, avoiding the need to return to a block at a later time to complete its extraction. To encourage the continuous mining of blocks, constraints are added restricting the number of blocks that can exist in a ‘partially mined’ state at the end of \( t \in \mathcal{T} \) in region \( j \in \bar{B} (\bar{P}_{jt}) \):

\[
\sum_{b \in \text{blocks}(j)} (y_{bt} - \tilde{y}_{bt}) \leq \bar{P}_{jt} \quad \forall \ t \in \mathcal{T}, j \in \bar{B}. \tag{29}
\]

### 4. A multi-objective rolling-horizon-based algorithm

Blom, Pearce and Stuckey (2016) present a rolling-horizon-based algorithm for constructing a single production schedule, optimized with respect to two fixed objectives. A horizon of \( T \) periods is
discretized into two period-aggregates of size 1 and \( T - 1 \) (the size of a period-aggregate refers to the number of periods it contains). A two-period MIP model of the scheduling problem is solved, first with an objective to minimize deviation between the grade of produced ore and desired bounds. Constraints are added to this MIP, preventing this deviation from exceeding that experienced in the obtained solution. The two-period MIP is re-solved with an objective to maximize equipment productivity. The activities of period 1 are fixed to those scheduled in this second solution. The algorithm next considers a two-period scheduling problem covering periods 2 to \( T \), repeating this process to fix the activities of period 2. The algorithm terminates once all \( T \) periods have been scheduled.

This section presents an adaptation of the rolling-horizon-based algorithm of Blom, Pearce and Stuckey (2016) in which a horizon of \( T \) periods is discretized into \( N \ll T \) period-aggregates of non-uniform and increasing size (Listing 1). The result is a fine discretization of time at the start of the horizon, and an increasingly coarse discretization toward the end. A procedure for discretizing a horizon in this way is provided in the Online Supplement (http://dx.doi.org/10.1080/0305215X.2016.1218002). The first aggregate in this discretized horizon has a size of one. The presented rolling-horizon-based algorithm solves a series of \( N \)-period instances of \( MIP_{STP} \) (Section 3) to build a schedule progressively. Schedule generation is optimized with respect to an arbitrary sequence of objectives \( Z \), ordered from highest to lowest priority. A feasible schedule is an instantiation of variables \( x \) that satisfies the constraints of Section 3.3. The presented algorithm is described using the following notation.

### Additional notation

- \( t' \) Time periods in the original (non-discretized) horizon.
- \( x \) Set of all decision variables defined in Section 3.
- \( x_{t'} \) The variables in \( x \) defining the activities of period \( t' \).
- \( \Phi_{t'} \) State of the mine at the start of period \( t' \), defining the tons remaining in each block and the tons on, and grade of, each stockpile.
- \( M_T \) Instantiation of \( MIP_{STP} \) over a discretized horizon \( T \).
- \( x^z \) Solution of model \( M_T \) when solved with respect to objective \( z \).
- \( \alpha_s, \alpha_b \) Split and branch factor used to control the number of schedules generated by the CRH algorithm (described in Section 5).

The presented rolling-horizon-based algorithm schedules the activities of each period \( t' \) in turn via repeated calls to \texttt{ProgressSchedule} (Step 5). With this procedure, a schedule is progressively built, one period at a time. Starting at \( t' = 1 \), this procedure discretizes the \( T \)-period horizon into \( N \) (or fewer) period-aggregates (Step 13). \( MIP_{STP} \) (Section 3) is instantiated over the discretized horizon \( T \) to form \( M_T \) (Step 14).

Model \( M_T \) is repeatedly solved with a series of different objectives. In Steps 15–18 of \texttt{ProgressSchedule}, each objective \( z \) in the sequence \( Z \) is considered in turn. For each \( z \in Z \), \( M_T \) is solved with respect to \( \xi(z) \) (Step 16), to find the optimal solution \( x^z \). In practice, time limits are imposed on the solving of each MIP, and \( x^z \) denotes the best solution found in this time limit. Prune constraints \( PR(z) \) are added to \( M_T \), discarding from the feasible solution space all solutions that are lower in quality (with respect to \( z \)) than \( x^z \). Each \( x^z \) defines a candidate instantiation for \( x_{t'} \). Upon consideration of the last objective \( z \) in \( Z \), the algorithm sets \( x_{t'} = x_T^z \), fixing the activities of period \( t' \). If \( M_T \) is infeasible (has no solution), \texttt{ProgressSchedule} will fail, and schedule generation will stop (Steps 6 and 17). One reason for such a failure is that the decisions made in periods prior to \( t' \) have led to a mine state in which there is insufficient processable ore available to supply processing equipment to their full capacity. The modelling of the entire horizon in \( M_T \), albeit in a coarse manner, aims to reduce the likelihood of such failures, however feasibility is not guaranteed. The case study of Section 6 shows that these failures do not occur often. In one of the nine optimization scenarios considered, the presented CRH algorithm produces six schedules instead of the expected nine (three schedules became
Listing 1 A rolling-horizon-based algorithm for the generation of a schedule $x$, given a horizon of $T$ periods, objectives $Z$ and discretization factor $N$

1: $x \leftarrow \emptyset$
2: $t' \leftarrow 1$
3: Initialize starting mine state $\Phi_1$
4: while $t' \leq T$ do
5:   $x_{t'} \leftarrow \text{ProgressSchedule}(t', \Phi_{t'}, T, N, Z)$
6:   if $x_{t'} \equiv \emptyset$ return $\emptyset$
7:   Compute new mine state $\Phi_{t'+1}$ on the basis of $x_{t'}$
8:   $t' \leftarrow t' + 1$
9: end while
10: return $x$

procedure ProgressSchedule($t'$, $\Phi_{t'}$, $T$, $N$, $Z$)
11: $x_{t'} \leftarrow 0$
12: $T \leftarrow \text{Discretize}(N, t', T)$
13: $M_T \leftarrow \text{Instantiate MIPSTP over } T \text{ and mine state } \Phi_{t'}$
14: for $z$ in $Z$ do
15:   $x_{t'} \leftarrow \text{SolveAndPrune}(M_T, z)$
16:   if $x_{t'} \equiv \emptyset$ return $\emptyset$
17: end for
18: return $x_{t'}$
19: end procedure

infeasible). The rolling-horizon-based algorithm of Blom, Pearce and Stuckey (2016), adapted in this article, has reported a failure rate of 3.7% on average across all conducted experiments.

The new mine state $\Phi_{t'+1}$ is computed to reflect the tons of material, of each type $l \in L$, remaining in each block at the end of period $t'$, in addition to the size and grade of each stockpile $s \in S$ (Step 7). The presented algorithm rolls forward by one period (Step 8), and repeats Steps 11–20 with $t' = 2$. The discretized horizon $T$ now covers periods $\{2, \ldots, T\}$. Steps 4–9 repeat until the activities of all periods have been fixed.

In each subsequent call to ProgressSchedule, where $t' > 1$, parameters $T_{st}$, $G_{sq}$ and $G_{sqp}$, for stockpile $s \in S$, attribute $q \in Q$ and plant $p \in P$, in the constraints of Section 3.3, are replaced with $T_{st'}$, $G_{sq'}$ and $G_{sq'p}$, as defined in $\Phi_{t'}$. The grade of each stockpile $s \in S$, at the start of period $t'$, is known, allowing us to model the changing state of stockpiles without nonlinear constraints.

5. Candidate generation via split-and-branch (CRH)

This section presents a concurrent rolling-horizon-based (CRH) algorithm. This algorithm implements a split-and-branch approach for the generation of multiple candidate schedules given a sequence of objectives $Z$, ordered from highest to lowest priority (Listing 2). A split factor, $\alpha_{s}$, and branch factor, $\alpha_{b}$, control both the number and diversity of schedules generated, and are user-defined parameters. CRH marks $\alpha_{s}$ periods in the horizon as split points, as shown in Figure 3, labelling $t' = 1$ as the first split point, and distributing the remaining $\alpha_{s} - 1$ split points evenly across the horizon. The set of these split points is denoted $S\mathcal{P}$. The CRH algorithm, defined in Listing 2, fixes the activities performed in a frontier of candidate schedules, $\mathcal{X}$, one period at a time, until reaching a split point $t' \in S\mathcal{P}$. At each split point $t' \in S\mathcal{P}$, a branch occurs, where each candidate $x \in \mathcal{X}$ is replaced by $\alpha_{b}$ new schedules. The schedules spawned from $x$ are equivalent to $x$, for all $t'' < t'$, but diverge in the activities scheduled for period $t'$. Up to $\alpha_{b}^{t'}$ schedules are generated by the algorithm (the branch
The CRH algorithm for multiple schedule generation, given a horizon of $T$ periods, objectives $Z$, discretization factor $N$, split points $SP$ and branching factor $\alpha_b$

1: $\mathcal{X} \leftarrow \emptyset$
2: $t' \leftarrow 1$
3: $x_1 \leftarrow \emptyset$  > $x_i$ refers to the $i$th schedule constructed by the algorithm
4: Initialize starting ($t' = 1$) mine state $\Phi_{11}$ for schedule $x_1$
5: $\mathcal{X} \leftarrow \{x_1\}$
6: $\Phi \leftarrow \{\Phi_{11}\}$
7: while $t' \leq T$ do
8:   for each $x_i$ in $\mathcal{X}$ do
9:      $(x_i, M_T) \leftarrow \text{ProgressSchedule'}(t', \Phi_{it'}, T, N, Z)$
10:         if $x_i \equiv \emptyset$ then
11:            Remove $x_i$ from $\mathcal{X}$
12:            continue
13:         endif
14:      Compute new mine state $\Phi_{it'+1}$ for schedule $x_i$ on the basis of the activities performed in period $t'$. Replace $\Phi_{it'}$ in $\Phi$ with $\Phi_{it'+1}$
15:      if $t' \in SP$ then
16:         for $j = 1$ to $\alpha_b - 1$ do
17:            $n \leftarrow |\mathcal{X}| + 1$
18:            $(x_{nt'}, \Phi_{nt'+1}) \leftarrow \text{ReplicateDiversify}(M_T, \Phi_{it'})$
19:            $\mathcal{X} \leftarrow \mathcal{X} \cup \{x_{nt'}\}$
20:            $\Phi \leftarrow \Phi \cup \{\Phi_{nt'+1}\}$
21:         end for
22:      endif
23:   end for
24:   $t' \leftarrow t' + 1$
25: end while
26: return $\mathcal{X}$

factor to the power of the number of split points). Schedules that reach a dead end in period $t'$ (the constraints of Section 3.3 cannot be satisfied) are removed from the frontier.

The set of candidate schedules, $\mathcal{X}$, is initially empty (Step 1). An empty schedule $x_1$ is created, in which no activities have been fixed (Step 3), and added to $\mathcal{X}$ (Step 5). The notation $x_i$ refers to the $i$th schedule constructed by the algorithm. A mine state, $\Phi_{1t'}$, describing the tons of material remaining in each block, and the composition of each stockpile, at the start of each period $t'$ is maintained alongside $x_1$ (the subscript 1 in $\Phi_{1t'}$ denotes that this is the state of the mine at the start of period $t'$ if schedule $x_1$ is performed). For each time period $t'$ in our horizon, CRH concurrently fixes the activities of period $t'$ in each schedule in $\mathcal{X}$ (Steps 8–23). At $t' = 1$, CRH progresses schedule $x_1$ (the only schedule currently in $\mathcal{X}$) by fixing the activities of period 1 (Step 9). A slight variation of the ProgressSchedule procedure of Listing 1 is defined in which, given a time period $t'$, mine state $\Phi_{it'}$, parameters $T$ and $N$, and a sequence of objectives $Z$, both schedule $x_i$ (with the activities of period $t'$ fixed) and the model $M_T$ (constructed and instantiated with the final objective in $Z$) are returned. This procedure is denoted ProgressSchedule'. If ProgressSchedule' fails to progress schedule $x_i$, this schedule is removed from $\mathcal{X}$ (Steps 10–13).

Once the activities of period $t'$ in schedule $x_i \in \mathcal{X}$ have been fixed (Step 9), CRH checks if $t'$ is a split point (Step 15). If $t' \in SP$, the algorithm computes $\alpha_b - 1$ alternative sets of activities for $t'$ in $x_i$ via ReplicateDiversify (Step 18). ReplicateDiversify takes, as input, the MIP model formed
Figure 3. At each split point $t' \in SP, \alpha_b$ new candidate schedules are constructed. In this figure, $\alpha_b = 3$.

Figure 4. A demonstration of the CRH algorithm for generating multiple schedules, given $\alpha_s = 2, \alpha_b = 3$ and $N = 3$. Three schedules are concurrently progressed after the first split point $t' = 1$, with six schedules created at the second, $t' = 7$. CPLEX® 12.6’s populate is used to find additional solutions to each MIP.
by ProgressSchedule, \( M_T \), after it has been solved with respect to each objective in \( Z \). Using the populate method of CPLEX\textsuperscript{®} 12.6, ReplicateDiversify fills a solution pool with solutions of \( M_T \) that have diverse instantiations of its integer variables—see Danna et al. (2007) for a description of the populate method. These integer variables indicate whether a block is mined, or completely extracted. Diversity in their instantiation results in schedules that mine blocks in a different sequence. Each repeated call to ReplicateDiversify selects the next solution in this pool, from which a new schedule is constructed and added to \( \mathcal{X} \) (Step 19).

At \( t' = 1 \), Steps 16–21 take the first \( \alpha_b - 1 \) solutions in the pool populated by ReplicateDiversify, and use these solutions to fix the activities of period \( t' = 1 \) in \( \alpha_b - 1 \) new schedules, \( x_2, \ldots, x_{\alpha_b} \). A mine state, \( \Phi_{t'} \), is maintained alongside each schedule \( x_i \), recording the tons of each block, and state of each stockpile, at the start of each period \( t' \), on the basis of the actions performed in \( x_i \). Each newly constructed schedule, together with its corresponding mine state, are added to \( \mathcal{X} \) and \( \Phi \) (Steps 19 and 20).

After the first iteration of Steps 7–25, \( \mathcal{X} \) contains \( \alpha_b \) schedules with the activities of period 1 fixed. Steps 7–25 are repeated, progressing each of these schedules, and updating their associated mine states in \( \Phi \), until the algorithm reaches the next split point \( t' \in \mathcal{SP} \). For schedule \( x_i \), ReplicateDiversify copies the activities of \( x_i \), for each \( t'' < t' \), into each of the \( \alpha_b - 1 \) schedules it creates. The activities of period \( t' \), in these new schedules, are determined via the population of a solution pool, as described above. CRH terminates once all \( T \) periods have been scheduled in each of the \( \alpha_b^{\alpha'} \) schedules constructed. Figure 4 demonstrates how the CRH algorithm creates and progresses multiple candidate schedules given a 13-period horizon, and parameters \( \alpha_z = 2, \alpha_b = 3 \) and \( N = 3 \).

6. Case study

The presented solver is evaluated on two data sets derived from a currently operating open-pit mine producing several million tons of a single product each year. In the first data set, denoted DS1, there are 404 grade blocks, 434 blast blocks, 5 material types, 5 quality attributes, 10 modelled stockpiles, 4 plants, 4 dig unit types and 3 truck types. Of the 838 blocks, 161 have no associated mining precedences (i.e., are immediately accessible), and 206 have disjunctive precedences (more than one way of accessing the block). In the second data set, denoted DS2, there are 526 grade blocks and 296 blast blocks. Of the 822 blocks, 321 have no associated precedences. Each block \( b \) has at most one way of accessing it \( |A_b| \leq 1 \). The number of material types, quality attributes, stockpiles, plants, dig unit types and truck types, is the same in both data sets. Equipment capacities, target grades and bounds on production grades, in each week of a 13-week horizon, differ between the two data sets. Each data set represents the state of the mine at two distinct time points, one year apart. Valid source-to-destination pathways are the same for both DS1 and DS2. This section demonstrates the efficiency with which the CRH solver finds schedules in a range of scenarios, and how the properties of these schedules change in response to changes in the objective sequence. All experiments in this article have been run on a machine with an Intel Xeon E5-1650 v2 3.5 GHz processor and 64 GB RAM. All MIPs have been solved with CPLEX\textsuperscript{®} 12.6.

For data set DS1, five distinct scenarios are considered (Table 1), labelled DS1\textsubscript{a−e}. These scenarios include the following objectives: maximizing alignment of the schedule with a longer-term plan (\( z_{\text{align}} \), Section 3.2.1); encouraging the continuous mining of blocks (\( z_{\text{cont}} \), mining blocks continuously until completely extracted); keeping stockpile sizes within desired bounds in each time period (\( z_{\text{piles}} \)); keeping the tons of waste mined in each period within desired bounds (\( z_{\text{waste}} \)); keeping the tons of material mined from specific regions within desired bounds (\( z_{\text{regions}} \)); and minimizing deviation between the grade of ore produced in each period and desired bounds (\( z_{\text{grade}} \), Section 3.2.2). Formal definitions for each of these objectives are provided in the Online Supplement (http://dx.doi.org/10.1080/0305215X.2016.1218002).

Scenarios DS1\textsubscript{b−d} replicate DS1\textsubscript{a} with a single objective removed: DS1\textsubscript{b} does not keep the tons of waste mined within desired bounds; DS1\textsubscript{c} does not align schedules with a longer-term plan; DS1\textsubscript{d}
Table 1. Optimization scenarios created for data set DS1.

| Scenario | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Schedules created | Time  |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------ |-------------------|-------|
| DS1a     | Minimize deviation from grade bounds $z_{\text{grade}}$ (narrowed by half), $z_{\text{align}}$, maintain alignment to a longer-term plan $z_{\text{align}}$, encourage the continuous mining of blocks $z_{\text{cont}}$, and keep stockpile sizes $z_{\text{piles}}$, tons of waste mined $z_{\text{waste}}$, and tons mined in specific regions $z_{\text{regions}}$, within bounds. | 9                 | 530   |
| DS1b     | $z_{\text{grade}}$, $z_{\text{align}}$, $z_{\text{cont}}$, $z_{\text{piles}}$, $z_{\text{waste}}$, $z_{\text{regions}}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | 9                 | 537   |
| DS1c     | $z_{\text{grade}}$, $z_{\text{align}}$, $z_{\text{cont}}$, $z_{\text{piles}}$, $z_{\text{waste}}$, $z_{\text{regions}}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | 9                 | 414   |
| DS1d     | $z_{\text{grade}}$ (upper bounds), $z_{\text{align}}$, $z_{\text{cont}}$, $z_{\text{piles}}$, $z_{\text{waste}}$, $z_{\text{regions}}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | 9                 | 558   |
| DS1e     | $z_{\text{align}}$, $z_{\text{cont}}$, $z_{\text{piles}}$, $z_{\text{waste}}$, $z_{\text{regions}}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 9                 | 323   |

Table 2. Minimum and maximum kilotons of waste mined across schedules generated in DS1a (where the mining of waste is optimized in the fifth objective) and DS1b (where the mining of waste is not optimized).

| Scenario | Waste mined per period (kt)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 1     | 2     | 3     | 4     | 5     | 6     | 7     | 8     | 9     | 10    | 11    | 12    | 13    |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------- |-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| DS1a     | Min 250 250 250 189.2 298.5 250 250 250 143.7 250 194 101                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | 250   | 250   | 250   | 189.2 | 298.5 | 250   | 250   | 250   | 143.7 | 250   | 194   | 101   |
|          | Max 250.5 252.2 250 418.1 699.9 250 298.8 501.7 371.2 250 526.1 269.1 167.6                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 250.5 | 252.2 | 250   | 418.1 | 699.9 | 250   | 298.8 | 501.7 | 371.2 | 250   | 526.1 | 269.1 | 167.6 |
| DS1b     | Min 100 179.3 100 100 360.7 140.8 100 220.6 193.8 100 150.5 103.9 100                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 100   | 179.3 | 100   | 100   | 360.7 | 140.8 | 100   | 220.6 | 193.8 | 100   | 150.5 | 103.9 | 100   |
|          | Max 199 310.5 154.9 231.7 802.2 187.9 234.7 446.5 401.2 340 643.7 154.3 167.6                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 199   | 310.5 | 154.9 | 231.7 | 802.2 | 187.9 | 234.7 | 446.5 | 401.2 | 340   | 643.7 | 154.3 | 167.6 |

aims to keep product grades below desired *upper* bounds only (narrowed by one half); and DS1c ignores objectives relating to production grade entirely. Table 1 records the number of schedules generated for each scenario and the time taken by the present authors’ solver to form those schedules (using parameters $\alpha_s = 2$, $\alpha_b = 3$ and $N = 2$).

Table 2 compares the kilotons of waste mined, in each period, in the schedules generated for scenario DS1a (which aims to mine at least 250 kt of waste each period) to those generated for DS1b (which includes no waste-related objective). The schedules formed for DS1a mine more waste, on average, than those generated for DS1b.

A longer-term plan identifies the set of blocks which should be removed by the end of the scheduling horizon. The schedules generated for scenario DS1a, mine, on average, 79.9% (standard deviation 2%) of the blocks in the relevant longer-term plan. An average of 68.4% (1.5%) of these blocks are mined across the schedules generated for scenario DS1c, where one does not optimize with respect to longer-term plan alignment.

In all of the schedules formed for DS1a, product grades lie within upper and lower site bounds, with very slight average deviations above (and below) narrowed site bounds present in some periods (no more than 0.1 $\Delta$, where $\Delta$ denotes the gap between the target grade of an attribute and its non-narrowed bounds). The schedules generated for DS1d, in which only deviation above upper grade bounds (narrowed by half) is considered, exhibit similarly small deviations above these bounds. In the schedules generated for DS1e, in which no optimization with respect to product grade occurs, attribute grades fall below narrowed lower bounds, and exceed narrowed upper bounds, by up to 3 $\Delta$, on average. These results show that the CRH solver can keep product grades within desired upper and lower bounds by minimizing deviation from *narrowed* upper and lower bounds.

The mine in the presented case study is primarily concerned with the levels of two contaminants, $q_2$ and $q_3$. In DS1, all attributes have associated lower and upper bounds. The second data set, DS2, defines upper bounds for attributes $q_2$ and $q_3$, hard cutoff bounds on all remaining contaminants, and a lower bound on metal content (attribute $q_1$). Table 3 lists each of the optimization scenarios applied to the second data set, DS2, the number of schedules found in each scenario, and the time taken to generate those schedules. Each scenario has been solved with $\alpha_s = 2$, $\alpha_b = 3$ and $N = 2$. Schedule generation is considerably less time consuming on this data set. The addition of disjunctive precedences in DS1 adds significant complexity to the problem.

The schedules created for scenarios DS2a–d are compared in terms of how well they keep attributes $q_2$ and $q_3$ below upper bounds, and $q_1$ above its lower bound. Across all schedules generated for
Table 3. Optimization scenarios created for DS2.

| Scenario | Description | Schedules created | Time (s) |
|----------|-------------|-------------------|----------|
| DS2a     | Minimize deviation of $q_3$ in produced ore above upper bound (narrowed by half), $z_{\text{grade}}^{a,0.5}$, minimize deviation of $q_2$ above upper bound (narrowed by half), $z_{\text{grade}}^{b,0.5}$, minimize deviation of $q_1$ below lower bound ($z_{\text{grade}}^{c}$), encourage the continuous mining of blocks ($z_{\text{regions}}$), keep tons of material mined in specific regions within bounds ($z_{\text{waste}}$), keep tons of waste mined within bounds ($z_{\text{waste}}$) | 9 | 162 |
| DS2b     | $z_{\text{grade}}^{a,0.75}$ ($q_3$ above upper bound, narrowed by three quarters), $z_{\text{grade}}^{a,0.75}$, $z_{\text{grade}}^{b,0.75}$, $z_{\text{regions}}$, $z_{\text{cont}}$, $z_{\text{waste}}$ | 9 | 205 |
| DS2c     | $z_{\text{grade}}^{a}$, $z_{\text{grade}}^{b}$, $z_{\text{regions}}$, $z_{\text{cont}}$, $z_{\text{waste}}$ | 9 | 146 |
| DS2d     | $z_{\text{grade}}^{a}$, $z_{\text{grade}}^{b}$, $z_{\text{regions}}$, $z_{\text{cont}}$, $z_{\text{waste}}$ | 9 | 183 |

Table 4. Number of schedules generated, and solver runtime, for varying values of $\alpha_a$ and $\alpha_b$.

| Parameters | Schedules generated | Time (s) |
|------------|---------------------|----------|
| $\alpha_a = 2$, $\alpha_b = 3$ | 9 | 162 |
| $\alpha_a = 3$, $\alpha_b = 3$ | 27 | 234 |
| $\alpha_a = 4$, $\alpha_b = 3$ | 81 | 476 |
| $\alpha_a = 2$, $\alpha_b = 4$ | 16 | 205 |
| $\alpha_a = 2$, $\alpha_b = 5$ | 25 | 273 |

The scenarios DS2a–d, product grades exhibit only slight deviations below the lower bound for $q_1$ (less than 0.1Δ in magnitude, on average). The schedules generated for scenarios DS2b and DS2d perform best at maintaining $q_2$ and $q_3$ below upper bounds (narrowed by half). This is unsurprising, as the objectives of DS2b and DS2d aim to keep $q_2$ and $q_3$ below upper bounds (narrowed by three quarters). Scenario DS2a, in contrast, aims to keep $q_2$ and $q_3$ below non-narrowed upper bounds only.

Each of the above experiments use a discretization factor of $N = 2$. Each schedule has thus been generated by solving a series of two-time-period MIPs. The following experiment considers the influence of this parameter on the degree to which the resulting schedules satisfy desired objectives. Consider a variation of DS2a, denoted DS2e, in which an objective to keep stockpiles sizes below upper bounds is inserted in the fourth position. Let $s_1$ and $s_2$ denote two stockpiles in the case study mine with preferred upper bounds $\bar{S}_{s_1}$ and $\bar{S}_{s_2}$ on their size at the end of each week. For $N = 2, 3$ and 4, 9 schedules are generated in 184, 319 and 443 s, respectively. For $N = 2$ and 3, excess kilotons are present on $s_1$ and $s_2$ in three time periods (the magnitude of these deviations is smaller for $N = 3$). For $N = 4$, excess kilotons are present on $s_1$ in only one time period.

Increasing the split ($\alpha_s$) or branch factor ($\alpha_b$) increases both solve times and the number of schedules generated by the CRH algorithm. Recall that $\alpha_s^{go}$ schedules are generated by CRH. For varying $\alpha_s$ and $\alpha_b$, Table 4 reports the number of schedules generated, together with solve times, for scenario DS2d. Schedule construction is parallelized, so that the generation of 81 schedules ($\alpha_s = 4$ and $\alpha_b = 3$) takes less than three times more wall clock time than the generation of 9. Each of these schedules produce ore with a different grade profile, extraction sequence and schedule of stockpile to plant movements.

7. Concluding remarks

This article has presented: a tool for the short-term scheduling of an open-pit mine, demonstrated on a currently operating site producing several million tons of ore each year; and a novel concurrent rolling-horizon-based algorithm for the generation of multiple distinct production schedules, each optimized with respect to a sequence of objectives. The presented system supports over 40 objectives, from which a planner can construct a diverse range of optimization scenarios. The tool presented in
this article for the short-term scheduling of open-pit mines advances the current state of the art with its ability to generate multiple candidate schedules, with respect to sequences of multiple objectives, each of which satisfies hard constraints on the utilization of processing plants. This system is capable of generating these schedules within minutes, whereas existing tools require days of parameter adjustment to achieve satisfactory results.

**Disclosure statement**

No potential conflict of interest was reported by the authors.

**Funding**

This work is funded by an Australian Research Council (ARC) Linkage grant [LP110100115] in conjunction with an industry partner.

**References**

Alves, Maria João, and João Clímaco. 2007. “A Review of Interactive Methods for Multiobjective Integer and Mixed-Integer Programming.” *European Journal of Operational Research* 180 (1): 99–115.

Alves, Maria João, and João Clímaco. 2009. “Multi-Objective Mixed Integer Programming.” In *Encyclopedia of Optimization*, edited by C. A. Floudas and P. M. Pardalos, 2454–2460. New York: Springer-Verlag. doi:10.1007/978-0-387-74759-0_421.

Askari-Nasab, H., Y. Pourrahimian, E. Ben-Awuah, and S. Kalantari. 2011. “Mixed Integer Linear Programming Formulations for Open Pit Production Scheduling.” *Journal of Mining Science* 47 (3): 338–359.

Blom, M. L., A. R. Pearse, and P. J. Stuckey. 2016. “A Decomposition-Based Algorithm for the Scheduling of Open-Pit Networks over Multiple Time Periods.” *Management Science*. doi:10.1287/mnsc.2015.2284.

Chinchuluun, A., and P. M. Pardalos. 2007. “A Survey of Recent Developments in Multiobjective Optimization.” In *Encyclopedia of Optimization*, edited by C. A. Floudas and P. M. Pardalos, 2454–2460. New York: Springer-Verlag. doi:10.1007/978-0-387-74759-0_421.

Cullenbine, C., R. K. Wood, and A. Newman. 2011. “A Sliding Time Window Heuristic for Open Pit Mine Block Sequencing.” *Optimization Letters* 5 (3): 365–377.

Danna, E., M. Fenelon, Z. Gu, and R. Wunderling. 2007. “Generating Multiple Solutions for Mixed Integer Programming Problems.” In *Proceedings of the Twelfth Conference on Integer Programming and Combinatorial Optimization (IPCO)*, Ithaca, NY, 25–27 June 2007, Vol. 4513 of the series Lecture Notes in Computer Science, 280–294. Berlin: Springer. doi:10.1007/978-3-540-72792-7_22.

Ehrgott, M., and X. Gandibleux. 2002. “Multiobjective Combinatorial Optimization—Theory, Methodology, and Applications.” In *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, Vol. 52 of International Series in Operations Research and Management Science, edited by M. Ehrgott and X. Gandibleux, 369–444. New York: Springer-Verlag. doi:10.1007/978-3-036-48107-3_8.

Ehrgott, M., and M. Wiecek. 2005. “Mutiobjective Programming.” In *Multiple Criteria Decision Analysis: State of the Art Surveys*, Vol. 78 of International Series in Operations Research and Management Science, 667–708. New York: Springer-Verlag. doi:10.1007/978-3-87-230815-1_17.

Espinoza, D., M. Goycoolea, E. Moreno, and A. Newman. 2012. “MineLib: A Library of Open Pit Mining Problems.” *Annals of Operations Research* 206 (1): 93–114.

Gershon, M. 1987. “Heuristic Approaches for Mine Planning and Production Scheduling.” *International Journal of Mining and Geological Engineering* 5 (1): 1–13.

Goodwin, G. C., M. M. Seron, R. H. Middleton, M. Zhang, B. F. Hennessy, P. M. Stone, and M. Menabde. 2006. “Receding Horizon Control Applied to Optimal Mine Planning.” *Optimal Control Applications to Management Sciences* 42 (8): 1337–1342.

Hustrulid, W., and M. Kuchta. 2006. *Open Pit Mine Planning and Design*. 2nd ed. London: Taylor & Francis.

Lambert, W. B., A. Brickey, A. M. Newman, and K. Eurek. 2014. “Open-Pit Block-Sequential Formulations: A Tutorial.” *Interfaces* 44 (2): 127–142. doi:10.1287/inte.2013.0731.

Lambert, W. B., and A. M. Newman. 2014. “Tailored Lagrangian Relaxation for the Open Pit Block Sequencing Problem.” *Annals of Operations Research* 222 (1): 419–438.

Newman, A. M., E. Rubio, R. Caro, A. Weintraub, and K. Eurek. 2010. “A Review of Operations Research in Mine Planning.” *Interfaces* 40 (3): 222–245.

O’Sullivan, D., and A. Newman. 2015. “Optimization-Based Heuristics for Underground Mine Scheduling.” *European Journal of Operational Research* 241 (1): 248–259.

Osanloo, M., J. Gholamnejad, and B. Karimi. 2008. “Long-Term Open Pit Mine Production Planning: A Review of Models and Algorithms.” *International Journal of Mining, Reclamation, and Environment* 22 (1): 3–35.