

Long-Term Mine Planning with Large Neighbourhood Search

Michelle Blom, Adrian R. Pearce

School of Computing and Information Systems, The University of Melbourne, Parkville, Victoria, Australia
michelle.blom@unimelb.edu.au, adrianrp@unimelb.edu.au

Pascal Côté

Operations Research and Decision Science, Rio Tinto, Canada
Pascal.Cote@riotinto.com

Abstract. We present a Large Neighbourhood Search based approach for solving complex long-term open-pit mine planning problems. An initial feasible solution, generated by a sliding windows heuristic, is improved through repeated solves of a restricted mixed integer program. Each iteration leaves only a subset of the variables in our planning model free to take on new values. We form these subsets through the use of a novel path-based neighbourhood structure, and neighbourhood formation strategies that exploit the structure of the planning model. We show that our method is able to find near-optimal solutions to problems that cannot be solved by an off-the-shelf solver in a reasonable time, or with reasonable computational resources.

Funding: This work was supported by Rio Tinto, the University of Melbourne’s Research Computing Services, and by the Australian Research Council (OPTIMA ITTC IC200100009).

Key words: Open-pit mining, Long-term planning, Large neighbourhood search

1. Introduction

The long-term scheduling of open-pit mines has received considerable research interest. Much existing work has focused on simplified models in which many of the characteristics of real-world mining have been ignored. Material from different regions of a mine site (called ‘blocks’) is extracted, in each period of a horizon, and sent to one of a number of destinations (e.g., processing plants, waste dumps, and stockpiles). Precedences constrain the order in which blocks can be extracted and capacity constraints exist on the total tons of material mined in each period, and sent to each destination.

We consider a long-term production scheduling problem for a set of open-pit mines, connected by rail to junctions at which blending takes place. The model includes blending constraints, minimum production constraints across regions of each mine (called pits), and capex decisions relating to the opening of pits. We present a Large Neighbourhood Search (LNS) based algorithm for solving this long-term open pit planning problem. We present a mixed-integer program (MIP) modelling of the problem, and evaluate our algorithm on a suite of real world instances.

Large Neighbourhood Search improves upon an existing solution to a problem by repeatedly solving a simplified version of the problem. In this simplification, a subset of variables are fixed to their values in the existing solution and the remainder left to take on a new value. The variables that are *free* to take on a new value form a *neighbourhood* around the current solution. The result is a smaller, restricted MIP. Solving this MIP allows us to search for a new and improved solution in this neighbourhood.

Our approach first applies a sliding windows method, described in Section 5, to find an initial feasible solution to our long-term planning problem. Once an initial solution has been found, we perform a *parallelised* form of LNS over multiple threads of computation. On each thread, we form a neighbourhood with respect to the current best found solution. In our context, this neighbourhood is a collection of blocks. Given such a neighbourhood, we leave the decision variables related to the mining of blocks in our neighbourhood free in our MIP, and fix the mining related decision variables for blocks outside of the neighbourhood to their values in the current best solution. We

solve these restricted MIPs, updating our current best solution to the best solution found over all threads. We describe our neighbourhood formation process, including the variables that we fix for each neighbourhood, in Section 6. We repeat these iterations of LNS until a termination criterion is satisfied. We describe several possible termination criteria in Section 6.3.3.

Much existing work on the use of neighbourhood-based search for long-term mine planning considers small neighbourhood structures based on shifting the mining of blocks from one time period to another, altering the destination of extracted material, and adding or removing blocks from the schedule. The complexity of our model means that we must consider much larger neighbourhoods in order to find different, feasible, and improved solutions.

To limit the size of our neighbourhoods, and to increase the likelihood that each neighbourhood will lead us to an improved solution, we have designed a novel *path-based* neighbourhood structure, and a series of novel neighbourhood formation strategies. We form a neighbourhood by repeatedly selecting a block from the model, forming a path around that block, and adding the blocks in that path to the neighbourhood. A path-based neighbourhood resembles a chain of blocks, where each block in the chain, excepting the first and last, is connected to a predecessor and successor. The predecessor must be completely extracted before the block can be mined, while the successor cannot be mined until the block is completely extracted. For each block in the chain, the neighbourhood includes all blocks in a restricted cone above and below. A restricted cone above (below) a block b contains all the predecessors (successors) of b that are located in b 's bench. We consider a range of strategies for selecting the blocks around which to form paths. These strategies define different methods for weighting the blocks in a model according to their value for the purposes of scheduling, and using those weights to guide selection. We also outline how we can leverage what we know about different sets of model constraints to form neighbourhoods that increase the likelihood of finding new feasible solutions. We show that our approach substantially improves upon the quality of the initial solutions formed through the sliding windows method.

The remainder of this paper is structured as follows. Related work is discussed in Section 2. A MIP model of the long-term mine planning problem we consider is detailed in Section 3. We describe three instances of this model, of increasing complexity, in Section 4. These instances are used to evaluate our LNS approach, described in Sections 5–6. Section 5 outlines our method for finding initial feasible solutions to the problem, while Section 6 describes our LNS algorithm. We evaluate our approach, contrasting its performance against solving the MIP directly, in Section 7.

2. Related Work

The scheduling of mines over long-term horizons (i.e., decades) has been well studied (see Osanloo et al. (2008), Newman et al. (2010), Askari-Nasab et al. (2011), Epstein et al. (2012), Lambert et al. (2014), Lamghari (2017), Zeng et al. (2021), and Fathollahzadeh et al. (2021) for reviews). Epstein et al. (2012) present mixed-integer program (MIP) models characterising a variety of long-term scheduling problems. Of these, the precedence constrained production scheduling problem (PCPSP) is most similar to the problem we tackle in this paper. In the PCPSP, blocks are extracted and sent to one of a number of facilities such as processing plants or stockpiles. Mining precedences constrain the order in which blocks are extracted, while side constraints can be specified to enforce resource capacities.

Numerous approaches have been developed to solve the PCPSP. Cullenbine et al. (2011) present a sliding time window heuristic (STWH) for solving a variation of the PCPSP, without continuous variables, in which a sequence of integer programs (IPs) are solved. In each IP, the full set of problem constraints are 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.

Our sliding windows heuristic, described in Section 5, is modelled on the work of Cullenbine et al. (2011). We consider both continuous and integer variables, however, and multiple destinations for extracted material. For each window, we include a varying number of additional relaxed time periods, rather than a relaxation of the remainder of the planning horizon.

Much existing work applying neighbourhood-based search techniques to mine scheduling problems employ simple neighbourhood structures. With an initial solution formed using the heuristics of Gershon (1987), Sari and Kumral (2016) employ a simple neighbourhood move that changes the period in which a selected block is mined, with a certain probability. Goodfellow and Dimitrakopoulos (2016) employ three neighbourhood structures when solving a stochastic mine planning problem: randomly selecting a block to either remove from the schedule, or alter its time of extraction; altering the destination of material from a randomly selected cluster of blocks; and altering the subsequent destination of material leaving stockpiles or processing plants. Lamghari and Dimitrakopoulos (2020) present hyperheuristics that select among a range of simple perturbative neighbourhood structures: shift the mining of a block to the previous or subsequent time period; move the mining of block to another period; or swap the periods in which two blocks are mined. A range of variations in how blocks and time periods are selected are presented, forming 27 different ways of perturbing a schedule.

Amaya et al. (2009) solve a series of IPs to incrementally improve an initial schedule found using the greedy heuristic of Gershon (1987). All variables in the PCPSP IP are first fixed to their value in this initial schedule. A selection of variables are then unfixed and the resulting IP solved to yield a (potentially) improved solution. These variables are then fixed to their value in this new solution. This process is repeated for all collections of variables generated in accordance with several strategies. The ‘cone above’ strategy defines, for each block b , a set of variables relating to the mining of b and its predecessors. The ‘periods’ strategy defines, for each period t , a set of variables relating to the activities of t . The ‘transversal’ strategy defines, for each block b , a set of variables relating to the mining of all blocks within a defined distance of b . Chicoisne et al. (2012) extend the heuristic of Amaya et al. (2009) with different methods of selecting which variables to fix. These methods select a random block b that has been scheduled for mining in some period t , then: unfix variables relating to the mining of b and its predecessors (or, alternately, its successors); or unfix all variables (to a maximum number) relating to blocks mined in periods $t - 1$, t , and $t + 1$.

Our LNS approach is similar to that of Amaya et al. (2009), in that we repeatedly select blocks around which to form a neighbourhood. Variables related to the mining of these blocks are left unfixed, and the resulting MIP solved to form a new solution. Our approach differs in the type of neighbourhood structure used, and the strategies applied to instantiate it. Simple perturbative neighbourhood structures, such as swapping or shifting the periods in which blocks are mined do not form large enough neighbourhoods to allow new feasible solutions to be discovered for our model. Altering the mining of a block may require changes to how its predecessors and their predecessors are mined, or how its successors and their successors are mined. This is the motivation behind our path-based neighbourhood structure. This type of neighbourhood can be considered as a refinement of the often used cone-above or below structures. The strategies we use to select blocks around which to form paths (Section 6.2) focus on the constraints in our model that are tight or more often responsible for infeasibilities (blending and minimum production constraints) and decisions that are likely to have a substantial impact on the objective (such as when pits are opened).

Lamghari and Dimitrakopoulos (2012) apply Tabu search to improve an initial schedule generated by greedily selecting eligible blocks to be mined in each time period. Their method repeatedly shifts, adds, or deletes the mining of a block to, or from, an eligible period such that the objective is improved. Shifts that reverse recently performed actions form part of a Tabu-list, and are not permitted. Lamghari et al. (2015) alternately form an initial schedule by considering each period t in turn, solving a linear program capturing mining precedences, but ignoring processing and mining capacities, to determine which blocks to mine. A repair heuristic repeatedly selects a block, of those mined in t , to remove from the schedule, while ensuring that mining precedences are not violated. A variable neighbourhood descent (VND) heuristic improves the quality of this initial schedule by repeatedly swapping the mining of two blocks in consecutive periods, shifting the mining of a block (and its successors) from one period to the next, or moving the mining of a block (and its predecessors) forward by one period.

Lamghari and Dimitrakopoulos (2016) compare several heuristics for long-term mine planning with geological uncertainty. These include the Tabu search approach of Lamghari and Dimitrakopoulos (2012); the variable neighbourhood descent (VND) approach of Lamghari et al. (2015); LNS based on network flow techniques (NF); and diversified local search (DLS). NF and DLS were found to be more efficient and more robust than Tabu search and VND. NF employs two neighbourhood structures – forward, and backward – each based on selecting blocks mined in one period to be moved to the next (or the previous) period, continually shifting the mining of blocks forward or backward in time until we achieve a new and feasible solution. DLS alternates the application of VND and NF.

Senécal and Dimitrakopoulos (2020) apply a parallel multi-neighbourhood form of Tabu search. The neighbourhood structures considered involve moving the mining of a block to a different time period, or changing its destination. Single or simultaneous applications of these moves form neighbourhoods around a current solution. The parallel algorithm maintains a pool of moves, grouped according to neighbourhood type. A number of threads operate in parallel, each taking a move group from the pool, computing the result of applying each combination of moves, and maintaining a record of those resulting in the most improvement. Once the pool is empty, the best move found across threads is applied, and Tabu structures are updated. This process is repeated until a stopping criterion is met.

Lamghari and Dimitrakopoulos (2022) define an adaptive LNS method for long-term mine planning with uncertainty. To generate an initial solution, the planning horizon is decomposed into smaller sub-problems, each solved in two stages. The first focuses on extraction decisions, and the second on material destinations. Their LNS employs multiple ways of destroying a schedule—selecting blocks to remove—and then repairing it—adding blocks to the schedule. Heuristics are used to identify blocks for removal, including: random selection; selection with the aim to reduce mining surpluses; selection based on time period mined or destination; and selection based on geological proximity and dependencies. Given a set of removed blocks, repair heuristics are then used to re-insert each block. Their final repair strategy—MIP repair—is most similar to our approach, in that it uses a MIP solver to find new values for the variables associated with removed blocks. Their approach is *adaptive* in the sense that the choice of destroy/repair methods to employ at any given point in the algorithm is determined probabilistically, on the basis of their past performance.

3. Model Formulation

Our long-term mine planning problem is defined in terms of a set of mines connected by rail to blending junctions. Each mine contains a set of pits, where each pit is composed of a set of blocks. A block contains a set of parcels of different material types, including high grade, low grade and

Table 1 Sets and indices involved in our long-term mine planning model.

Set	Description
\mathcal{M}	The set of pits in our model, indexed by m .
\mathcal{R}	Set of saleable products produced, indexed by r .
$\mathcal{N}_{\mathcal{R}}$	Set of product nodes in the overall flow network.
\mathcal{F}_m	Flow network for pit $m \in \mathcal{M}$ with nodes \mathcal{N}_m and arcs \mathcal{A}_m . A directed arc $(i, j) \in \mathcal{A}_m$ connects nodes $i \in \mathcal{N}_m$ and $j \in \mathcal{N}_m \cup \mathcal{N}_{\mathcal{R}}$. The arc $(m, j) \in \mathcal{A}_m$ denotes the path from the blocks in pit m to destination j . A single source node is defined to represent the set of blocks in a pit.
\mathcal{N}_m^s	Nodes in the flow network for pit $m \in \mathcal{M}$ that represent stockpiles.
\mathcal{B}	The set of blocks in the model, indexed by b .
\mathcal{B}_m	The set of blocks in pit $m \in \mathcal{M}$, indexed by b .
\mathcal{D}	Mining precedences between blocks in \mathcal{B} , where each $(i, j) \in \mathcal{D}$ indicates that the extraction of block $i \in \mathcal{B}$ cannot begin until block $j \in \mathcal{B}$ has been depleted.
\mathcal{L}_b	Material types in block $b \in \mathcal{B}_m$ of pit $m \in \mathcal{M}$, indexed by l . Where a variable, set or parameter relates to a block $b \in \mathcal{B}_m$, we do not include the index m in its subscript, but rather let b imply the pit.
\mathcal{P}_{bl}	Parcels of type $l \in \mathcal{L}_b$ in block $b \in \mathcal{B}_m$, pit $m \in \mathcal{M}$, indexed by p .
\mathcal{T}	Set of time periods in the planning horizon, indexed by t .
\mathcal{J}	A collection of <i>sets of pits</i> , indexed by K , where for each pit set $K \in \mathcal{J}$ a minimum production constraint is present.
\mathcal{E}	Set of mineral elements of interest, indexed by e .

waste. The optimization problem aims to find a yearly block extraction sequence that maximizes the net present value of multiple products. Extracted material flows through a network composed of different elements—such as crushers, stockpiles, and processing plants—transforming ore into products. Saleable products are defined by different grade (blending) constraints, imposing minimum and maximum levels on a range of mineral elements. We take into account maximum mining rate constraints based on equipment capacities, and flow network node limits such as crusher and stockpile capacities. Finally, in some particular cases, mines have to contribute to a yearly minimum production imposed by joint venture exploitation and contractual agreements.

3.1. Notation

We view a set of mines as a collection of pits, with constraints formed over sets of pits. We use \mathcal{M} to denote the set of pits, and m to denote a specific pit. A flow network is defined for each pit, as described below. Each of these networks contains a source node, which we also denote by m , representing the pit from which blocks are extracted, and a number of destination nodes. These destinations, capturing stockpiles, crushers, processing plants, and dumps, are connected to a virtual node r , for each saleable product $r \in \mathcal{R}$. Tables 1–3 identify and define the sets, variables, and parameters involved in our long-term mine planning model.

3.2. Objective

The maximisation of net present value (NPV) is the typical objective of long-term mine planning problems. The NPV of a mining system captures the total value of the investment over a planning horizon, defined by the sum of (discounted) cash flows for each time period. Equation (3.2) expresses our objective, where Rev_{mt} denotes the revenue associated with products formed from ore in pit $m \in \mathcal{M}$ in period t (Equation (2)), Cost_{mt} the total costs incurred from the extraction and processing of material from pit m (Equation (3)), and π_t the discount factor for period t .

$$\max \sum_{t \in \mathcal{T}} \pi_t \left(\sum_{m \in \mathcal{M}} \text{Rev}_{mt} - \text{Cost}_{mt} \right) \quad (1)$$

Table 2 Variables involved in our long-term mine planning model.

Variable	Description
f_{pij}^t	Fraction of parcel $p \in \mathcal{P}_{bl}$, of type $l \in \mathcal{L}_b$, from block $b \in \mathcal{B}_m$ in pit $m \in \mathcal{M}$, that flows along the path $(i, j) \in \mathcal{A}_m$ in period $t \in \mathcal{T}$. Where a variable relates to a parcel p , we exclude indices b, l , and m from its subscript, letting p imply the block, type, and pit.
x_b^t	Fraction of block $b \in \mathcal{B}_m$, pit $m \in \mathcal{M}$, extracted by the end of $t \in \mathcal{T}$.
y_b^t	Binary variable that takes on a value of 1 if and only if block $b \in \mathcal{B}_m$, pit $m \in \mathcal{M}$, has been completely extracted by the end of $t \in \mathcal{T}$.
z_b^t	Binary variable that takes on a value of 1 if and only if extraction of block $b \in \mathcal{B}_m$, pit $m \in \mathcal{M}$, has commenced by the end of $t \in \mathcal{T}$.
s_{ip}^t	Fraction of parcel $p \in \mathcal{P}_{bl}$ of type $l \in \mathcal{L}_b$, from block $b \in \mathcal{B}_m$, pit $m \in \mathcal{M}$, in stockpile $i \in \mathcal{N}_m^s$ in period $t \in \mathcal{T}$.
wi_m^t	Binary variable that takes on a value of 1 if and only if pit $m \in \mathcal{M}$ is opened in period $t \in \mathcal{T}$.
wp_m^t	Binary variable that takes on a value of 1 in period $t \in \mathcal{T}$ if and only if pit $m \in \mathcal{M}$ has been opened in, or prior to, t .

Table 3 Parameters involved in our long-term mine planning model.

Parameter	Description
π_t	Discount factor for period $t \in \mathcal{T}$.
τ_p	Tonnage of parcel p .
σ_r	Revenue per ton of product $r \in \mathcal{R}$.
α_p	Extraction cost per ton for parcel p .
α_m^x	Capex investment cost associated with opening pit $m \in \mathcal{M}$.
ζ_{ep}	Concentration (level) of mineral element $e \in \mathcal{E}$ in parcel p .
ζ_{er}^{min}	Minimum required level of mineral element $e \in \mathcal{E}$ in product $r \in \mathcal{R}$.
ζ_{er}^{max}	Maximum allowed level of mineral element $e \in \mathcal{E}$ in product $r \in \mathcal{R}$.
Λ_n^{max}	Maximum tonnage permitted to exit node n , where n is a node in the overall network flow. Where n is the source node of a pit's flow network, Λ_n^{max} is the maximum mining capacity for that pit.
Ω_{Kt}^{min}	Minimum tonnage to be mined from the pits $K \in \mathcal{J}$ in period $t \in \mathcal{T}$.
Γ_i^t	Maximum capacity of stockpile $i \in \mathcal{N}_m^s$, pit $m \in \mathcal{M}$, in $t \in \mathcal{T}$.

where:

$$\text{Rev}_{mt} = \sum_{b \in \mathcal{B}_m} \sum_{l \in \mathcal{L}_b} \sum_{p \in \mathcal{P}_{bl}} \sum_{\substack{(i,r) \in \mathcal{A}_m \\ |r \in \mathcal{N}_{\mathcal{R}}}} \tau_p \sigma_r f_{pir}^t \quad (2)$$

$$\forall t \in \mathcal{T}, \forall m \in \mathcal{M}$$

$$\text{Cost}_{mt} = \alpha_m^x wi_m^t + \sum_{b \in \mathcal{B}_m} \sum_{l \in \mathcal{L}_b} \sum_{p \in \mathcal{P}_{bl}} \sum_{(m,j) \in \mathcal{A}_m} \tau_p \alpha_p f_{pmj}^t \quad (3)$$

$$\forall t \in \mathcal{T}, \forall m \in \mathcal{M}$$

The revenue generated from pit $m \in \mathcal{M}$ in period $t \in \mathcal{T}$ considers the contribution from each parcel $p \in \mathcal{P}_{bl}$, of each type $l \in \mathcal{L}_b$, from each block in the pit $b \in \mathcal{B}_m$, to each saleable product $r \in \mathcal{R}$, and the revenue generated per ton of that product. The cost associated with pit m in period t includes capex costs incurred when opening the pit, and extraction costs. The former is incurred in the period in which the pit is opened (where $wi_m^t = 1$). Extraction costs consider the material extracted from the pit, moving along paths from its source node m to destinations in its flow network.

3.3. Constraints

Blending constraints (4)–(5) enforce lower and upper bounds on the concentration of each mineral element $e \in \mathcal{E}$ in each formed product $r \in \mathcal{R}$.

$$\sum_{m \in \mathcal{M}} \sum_{b \in \mathcal{B}_m} \sum_{l \in \mathcal{L}_b} \sum_{p \in \mathcal{P}_{bl}} \sum_{(i,r) \in \mathcal{A}_m} (\zeta_{ep} - \zeta_{er}^{\min}) \tau_p f_{pir}^t \geq 0 \quad (4)$$

$$\forall r \in \mathcal{N}_{\mathcal{R}}, \forall e \in \mathcal{E}, \forall t \in \mathcal{T}$$

$$\sum_{m \in \mathcal{M}} \sum_{b \in \mathcal{B}_m} \sum_{l \in \mathcal{L}_b} \sum_{p \in \mathcal{P}_{bl}} \sum_{(i,r) \in \mathcal{A}_m} (\zeta_{ep} - \zeta_{er}^{\max}) \tau_p f_{pir}^t \leq 0 \quad (5)$$

$$\forall r \in \mathcal{N}_{\mathcal{R}}, \forall e \in \mathcal{E}, \forall t \in \mathcal{T}$$

Constraint set (6) enforces capacities on the tonnage of material exiting any given intermediate node n in our overall flow network. This excludes source nodes in the flow network for each pit, and product nodes.

$$\sum_{b \in \mathcal{B}_m} \sum_{l \in \mathcal{L}_b} \sum_{p \in \mathcal{P}_{bl}} \sum_{(n,j) \in \mathcal{A}_m} \tau_p f_{pnj}^t \leq \Lambda_n^{\max} \quad (6)$$

$$\forall m \in \mathcal{M}, \forall n \in \mathcal{N}_m \mid n \neq m, \forall t \in \mathcal{T}$$

Constraint set (7) enforces mining capacities in each pit $m \in \mathcal{M}$. Recall that wp_m^t is a binary indicating whether pit m has been opened by, or in, period $t \in \mathcal{T}$. When set to 1, the pit's mining capacity becomes Λ_m^{\max} .

$$\sum_{b \in \mathcal{B}_m} \sum_{l \in \mathcal{L}_b} \sum_{p \in \mathcal{P}_{bl}} \sum_{(m,j) \in \mathcal{A}_m} \tau_p f_{pmj}^t \leq \Lambda_m^{\max} wp_m^t \quad (7)$$

$$\forall m \in \mathcal{M}, \forall t \in \mathcal{T}$$

Recall that w_i^t is a binary variable signalling whether the pit $m \in \mathcal{M}$ was opened in period t . This binary variable triggers the binary w_p^t , which in turn activates extraction capacity for pit m from period t onward.

$$\begin{aligned} w_p^t &\leq \sum_{\tau=1,2,\dots,t} w_i^\tau \\ \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \end{aligned} \quad (8)$$

Constraint set (9) enforces minimum production constraints across collections of pits, $K \in \mathcal{J}$.

$$\begin{aligned} \sum_{m \in K} \sum_{b \in \mathcal{B}_m} \sum_{l \in \mathcal{L}_b} \sum_{p \in \mathcal{P}_{bl}} \sum_{(m,j) \in \mathcal{A}_m} \tau_p f_{pmj}^t &\geq \Omega_{Kt}^{\min} \\ \forall K \in \mathcal{J}, \forall t \in \mathcal{T} \end{aligned} \quad (9)$$

Mass balance constraints (10) are defined for all intermediate nodes in our overall flow network, excluding source nodes and those representing stockpiles (nodes \mathcal{N}_m^s for pit $m \in \mathcal{M}$).

$$\begin{aligned} \sum_{(i,n) \in \mathcal{A}_m} f_{pin}^t &= \sum_{(n,j) \in \mathcal{A}_m} f_{pnj}^t \\ \forall t \in \mathcal{T}, \forall m \in \mathcal{M}, \forall b \in \mathcal{B}_m, \forall l \in \mathcal{L}_b, \\ \forall p \in \mathcal{P}_{bl}, \forall n \in \mathcal{N}_m \setminus \mathcal{N}_m^s \cup \{m\} \end{aligned} \quad (10)$$

Mass balance constraints (11) are also defined for each node in our flow network representing a stockpile to keep track of the fraction of each parcel present in the stockpile in any given time period.

$$\begin{aligned} s_{ip}^t &= s_{ip}^{t-1} + \sum_{(j,i) \in \mathcal{A}_m} f_{pji}^t - \sum_{(i,j) \in \mathcal{A}_m} f_{pij}^t \\ \forall m \in \mathcal{M}, \forall i \in \mathcal{N}_m^s, \forall b \in \mathcal{B}_m, \forall l \in \mathcal{L}_b, \\ \forall p \in \mathcal{P}_{bl}, \forall t \in \mathcal{T} \mid s_{ip}^0 &= 0 \end{aligned} \quad (11)$$

Constraint set (12) ensures that the fraction of a parcel leaving a stockpile in period $t \in \mathcal{T}$ does not exceed the fraction of the parcel that was present in the stockpile in period $t - 1$.

$$\begin{aligned} \sum_{(i,j) \in \mathcal{A}_m} f_{pij}^t &\leq s_{ip}^{t-1} \\ \forall m \in \mathcal{M}, \forall i \in \mathcal{N}_m^s, \forall b \in \mathcal{B}_m, \forall l \in \mathcal{L}_b, \\ \forall p \in \mathcal{P}_{bl}, \forall t \in \mathcal{T} \mid t &\geq 2 \end{aligned} \quad (12)$$

Stockpile capacities are enforced by constraint set (13), where Γ_i^t denotes the capacity of stockpile $i \in \mathcal{N}_m^s$ in period t .

$$\sum_{b \in \mathcal{B}_m} \sum_{l \in \mathcal{L}_b} \sum_{p \in \mathcal{P}_{bl}} \tau_p s_{ip}^t \leq \Gamma_i^t \quad (13)$$

$$\forall m \in \mathcal{M}, \forall i \in \mathcal{N}_m^s, \forall t \in \mathcal{T}$$

The following sets of constraints ensure that the block extraction sequence is feasible according to the geological model and slope constraints. The first set of constraints (14) enforces mining precedences between blocks. For a given precedence relationship $(i, j) \in \mathcal{D}_m$ between blocks i and j in pit m , extraction of block i cannot begin until block j has been depleted. Moreover, we cannot progress the mining of a block $b \in \mathcal{B}_m$ until extraction of the block has begun, $z_b^t = 1$ (15). A block $b \in \mathcal{B}_m$ is not depleted until the fraction of the block mined by a time period, x_b^t , is equal to 1 (16).

$$z_i^t \leq y_j^t \quad \forall (i, j) \in \mathcal{D}, \forall t \in \mathcal{T} \quad (14)$$

$$x_b^t \leq z_b^t \quad \forall b \in \mathcal{B}, \forall t \in \mathcal{T} \quad (15)$$

$$y_b^t \leq x_b^t \quad \forall b \in \mathcal{B}, \forall t \in \mathcal{T} \quad (16)$$

Constraint sets (17)–(19) enforce coherence in the state of a block over time.

$$x_b^t \leq x_b^{t+1} \quad \forall b \in \mathcal{B}, \forall t \in \mathcal{T} \quad (17)$$

$$y_b^t \leq y_b^{t+1} \quad \forall b \in \mathcal{B}, \forall t \in \mathcal{T} \quad (18)$$

$$z_b^t \leq z_b^{t+1} \quad \forall b \in \mathcal{B}, \forall t \in \mathcal{T} \quad (19)$$

Constraint set (20) connects the material flow variables f_{pmj}^t associated with the extraction of a block $b \in \mathcal{B}_m$ to its progression variables. Constraint set (21) restricts the mining of a parcel to its available tonnage.

$$\sum_{(m,j) \in \mathcal{A}_m} \sum_{l \in \mathcal{L}_b} \sum_{p \in \mathcal{P}_{bl}} f_{pmj}^t = x_b^t - x_b^{t-1} \quad (20)$$

$$\forall m \in \mathcal{M}, \forall b \in \mathcal{B}_m, \forall t \in \mathcal{T} \mid x_b^0 = 0$$

$$\sum_{t \in \mathcal{T}} \sum_{(m,j) \in \mathcal{A}_m} f_{pmj}^t \leq \tau_p \quad (21)$$

$$\forall m \in \mathcal{M}, \forall b \in \mathcal{B}_m, \forall l \in \mathcal{L}_b, \forall p \in \mathcal{P}_{bl}$$

4. Problem Instances

We evaluate our LNS method on three real world mining projects.

4.1. Test Case 1 (T1)

T1 is a copper deposit comprising of 64 blocks, a 15 year planning horizon, and a single saleable product. The model contains no blending constraints, and no minimum production constraints. For confidentiality reasons, no further details can be revealed.

4.2. Oyu Tolgoi project

Oyu Tolgoi (OT) is a large copper and gold deposit in the south of the Gobi region in Mongolia, jointly owned by the Mongolian government and Rio Tinto. It is composed of both open-pit and underground mines, with production expected to reach over 500 kt of copper per annum by 2027.

We capture only the open pit mines of the deposit in our model. The model is comprised of two large open pits, each containing more than 100 blocks, with each block containing parcels of multiple material types. The model has a supply chain network comprised of multiple dumps, stockpiles and crushers. Blending constraints ensure the production of saleable gold, silver, and copper products. Minimum production constraints are not present in this model. The OT model is defined over a horizon of 15 years, and contains 4.4 million variables and 3.4 million constraints.

4.3. Pilbara Iron Ore System

The Pilbara Iron Ore system is composed of over 200 pits distributed across 17 hubs. These pits contribute to five iron ore products whose composition is controlled by blending constraints. These products are transported through a 200km rail system that connects the 17 hubs to 4 ports. Minimum production constraints are enforced across subsets of pits. The model is defined over a horizon of 15 years, and is composed of 11.8 million variables and 9.4 million constraints.

5. Recovery of a Feasible Solution

Sliding windows is a popular technique for tackling large-scale time-indexed optimization problems. It decomposes such problems by splitting the planning horizon into *windows*, and solving the problem for each window in turn. We model our instantiation of the heuristic on the work of Cullenbine et al. (2011).

Let W denote the number of time periods in each window (its width). The heuristic starts by solving the MIP over periods 1 to W . Once this sub-problem is solved, we move the start of our window forward to period $W - O$, where O represents an *overlap*. This parameter controls the extent to which consecutive problems overlap in terms of the horizon. As the window moves forward to period $W - O$, all variables in periods 1 to $W - O - 1$ are fixed to the solution found after solving the first sub-problem. This process is repeated until the last time period in our horizon has been scheduled.

There are two major drawbacks of the heuristic, as it has been described above. When minimum production targets are present across the planning horizon, it may be the case that these targets can only be satisfied by stockpiling material in early time periods. With the short-term view of each window, the necessity of doing this may not be apparent, resulting in infeasible sub-problems as our window moves forward. The second drawback is that in the optimal solution to our MIP, for a given instance, it may be that certain capex decisions are triggered in early time periods. The width of the window may be too small, however, to see the future benefits of triggering these decisions. Increasing window size increases the complexity of each sub-problem and the time required to generate an initial solution. We address both of these limitations by extending each window with

Table 4 Four neighbourhood formation strategies used during LNS.

STRATEGY	SECTION	DESCRIPTION
Blending	6.2.1	Seeks to support schedule changes while respecting blending constraints.
Timing	6.2.2	Changes to the scheduling of multiple blocks, across multiple time periods, are supported.
Pit links	6.2.3	Exploits the presence of minimum production constraints.
Trigger	6.2.4	Seeks to improve NPV by allowing key capex decisions, such as the opening or closing of a pit, to be changed.

H relaxed time periods. Instead of solving sub-problems containing W time periods, we solve sub-problems containing $W + H$ time periods, with all integer variables in the last H of these periods becoming continuous.

6. Large Neighbourhood Search

We use a novel path-based neighbourhood formation process to construct neighbourhoods in our LNS-based approach (Section 6.1). In conjunction with this process, we define a series of novel strategies to guide neighbourhood formation during each iteration of solving. These strategies seek to exploit characteristics of the problem model, and of the current best found solution. We use four neighbourhood formation strategies, as outlined in Table 4, and described in detail across Sections 6.2.1 to 6.2.4.

6.1. Path-Based Neighbourhoods

Given the highly constrained nature of our model, we observe that the likelihood of finding new, and better, solutions is low unless our neighbourhood allows changes in the extraction sequence across multiple time periods, involving blocks that span precedence chains, and that have the potential to increase NPV if the timing of their extraction changes.

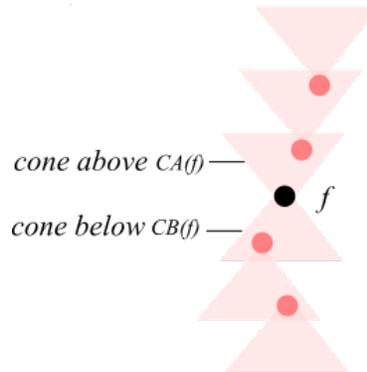
We may have a block b that, if mined earlier, would increase NPV. To achieve this change, and retain a feasible schedule, we may need to adjust the scheduling of blocks in a precedence chain involving b . Let \mathcal{B} denote the set of blocks across all pits. We form neighbourhoods by repeatedly selecting a *focal* block $f \in \mathcal{B}$, forming a *path* around that block, $p(f) \subset \mathcal{B}$, and adding the blocks in $p(f)$ to our neighbourhood. This process is repeated until our neighbourhood reaches an upper bound on its size, \bar{N} .

6.1.1. Block Selection To form neighbourhoods with blocks that are likely to have an impact on NPV, we select focal blocks randomly on the basis of a weighted distribution. Each block is assigned a weight, designed to represent its value for the purposes of scheduling. We consider several different methods for weighting blocks: random; objective; minimum dependencies; and mixed criteria. These methods are described in Table 5, and form the basic approach for weighting blocks for neighbourhood selection. Section 6.2 describes four further strategies for adjusting these weights by taking into specific features of our model such as blending and minimum production constraints, and capex decisions.

6.1.2. Path Formation To form a path around a block f , we first consider the blocks in its *restricted cone above*, $CA(f)$. These are the predecessors of f that belong to the same bench as f . Both f , and $CA(f)$, are added to $p(f)$. We then consider the blocks in its *restricted cone below*, $CB(f)$, which are also added to $p(f)$. To expand this path, we randomly select a block b in $CA(f)$ according to their values, as per Section 6.1.1. Block b and $CA(b)$ are added to $p(f)$. At the same time, we select a block b' in $CB(f)$, adding both b' and $CB(b')$ to $p(f)$. We continue this process

Table 5 Block selection strategies used in neighbourhood formation.

STRATEGY	DESCRIPTION
Random (RAND)	Focal blocks are selected randomly from \mathcal{B} .
Objective (OBJ)	The contribution of a block to the objective function is used as its weight. In this paper, we use the Iron (Fe) content of the block as a proxy for its contribution to the objective function.
Minimum Dependencies (MD)	Blocks with fewer predecessors and successors are given higher weights than those with more. The weight of block with n predecessors and successors in total, is set to $1/n$ for $n > 0$ and 0 otherwise. The idea is to bias selection toward potential bottleneck blocks that, when scheduled, have a substantial influence on the overall schedule.
Mixed Criteria (MIX)	Each LNS solving thread is assigned one of the three weighting methods to use – random is assigned to the first thread, objective to the second, and minimum dependencies to the third, with the pattern repeated across the remaining threads.

Figure 1 Visual depiction of the formation of a *path* of blocks, around a focal block $f \in \mathcal{B}$, during the construction of a neighbourhood.

of expanding our path upwards, and downwards, until the cones above and below our selected blocks are both empty, or we have reached the maximum neighbourhood size, \bar{N} . Figure 1 visually depicts the formation of a path-based neighbourhood.

6.2. Neighbourhood Formation Strategies

Section 6.1 describes the basic method we use to form path-based neighbourhoods. We define several variations of this approach, each designed to exploit different aspects of the model structure, and current best solution, in order to maximise the likelihood of finding a better solution. While these strategies exploit specific features of our model, these features are typical of real-world long-term mine planning models.

6.2.1. Blending Blending constraints exist in our model, for each time period. Each block makes a certain contribution to each of these constraints, for each of the available paths along which material from the block can travel. This contribution may be positive, negative, or zero. These constraints are satisfied if the sum of these contributions, over the set of blocks in our model, is greater than zero. Using the Blending strategy, we keep track of the total contribution of blocks already in the neighbourhood being formed, N , to each of the blending constraints, c ,

denoted $c(N)$. While the blocks in N will be mined across different time periods, we consider an approximation in which we imagine the total contribution of the neighbourhood to each blending constraint, as if it was being mined all at once.

The Blending strategy restricts focal block selection to the set of blocks with non-zero weights, and that do not make a negative contribution to any blending constraint c where $c(N) < 0$, given the current state of N . This is effectively achieved by setting the weight of blocks that do not satisfy this criterion to zero when selecting focal blocks. To form a path around a focal block, the process of Section 6.1.2 is followed, with no restrictions on which blocks are selected to extend the path.

6.2.2. Timing For a given focal block, f , let S_f and F_f denote the time periods in which mining starts, and finishes, in the current best solution. If the focal block is not scheduled for extraction, we set $S_f = F_f = |\mathcal{T}| + 1$. When using the Timing strategy, we start by following the neighbourhood formation process described in Section 6.1. Once we select a focal block for which $S_f < |\mathcal{T}| + 1$, we restrict all subsequent focal block selections to the set of blocks with both non-zero weights and that either start being extracted, or are mined to completion, in a period between S_f and F_f . The idea is to form a neighbourhood with subsets of blocks mined at similar times.

Once a neighbourhood has been formed, we reject it, and form another, if all focal blocks involved in its construction are not scheduled for mining in the current best found solution. The intuition is that these neighbourhoods have a low likelihood of leading to a better solution.

6.2.3. Pit Links We consider instances of our model with minimum production constraints. Each of these constraints is defined over a subset of pits. To find an improved solution, we need to be able to make changes to an existing schedule while still satisfying these constraints.

Under the Pit Links strategy, we adjust the weights of blocks, for the purpose of focal block selection, as the neighbourhood is constructed. Initially, blocks are weighted as per Section 6.1.1. After the first path of blocks is added to the neighbourhood, these values are adjusted for subsequent focal block selections. Let \mathcal{M}_N denote the subset of pits that are covered by the current neighbourhood, N . A pit m is covered by N if there exists a block $b \in N$ that belongs to m . We set the weights of all blocks b to zero if b does not belong to a pit in \mathcal{M}_N and b does not belong to a pit that is linked, by a minimum production constraint, to a pit in \mathcal{M}_N . If the weights of all remaining blocks are zero, and we have not reached the maximum neighbourhood size, we reset all block weights to their original values.

6.2.4. Trigger Our model involves decisions, such as the opening of a pit, that are associated with capex costs. Adjusting the timing of when a pit is opened, or even if the pit is never opened, has the potential to achieve significant changes in NPV. In our model, binary variables, denoted *pit triggers*, are associated with the opening of each pit (variables w_m^t). We can associate each pit trigger variable with a subset of blocks. The scheduling of these blocks determines the timing of these decisions. Under the Trigger strategy, we do not form path-based neighbourhoods. Instead, we randomly select a trigger variable, and include its associated blocks in the neighbourhood being constructed. We repeat this process until we reach a desired neighbourhood size.

6.3. Solving Strategies

In addition to neighbourhood construction strategies and parallelism, we employ a number of other strategies to improve the performance of the overall algorithm. These are: RINS-based techniques when solving each MIP (Section 6.3.1); variations on what variables are fixed for a given neighbourhood (Section 6.3.2); and varying termination criteria (Section 6.3.3).

6.3.1. RINS Relaxation Induced Neighbourhood Search (RINS) is a neighbourhood construction approach that utilises a current incumbent solution and information obtained by solving the linear relaxation of a MIP (Danna et al. 2005). The idea is that variables that take on the same value in the incumbent, and in the MIPs linear relaxation, will likely take on the same value in a good solution. We borrow this idea to allow larger neighbourhood sizes, in terms of numbers of blocks, while retaining reasonable solve times in each iteration of LNS. When using this strategy, we first solve the linear relaxation of each MIP formed during LNS. We fix each integer variable that takes on the same value in the current best solution (the incumbent), and in the solution to the linear relaxation, to that value. The MIP is then re-solved.

6.3.2. Fixing variables For a given neighbourhood, N , we form a restricted MIP based on N as follows. For each block b that *does not belong to* N , we fix the mining decisions relating to that block to their values in the current best solution. The decisions we consider fixing for a block b are the block start and depletion binaries z_b^t and y_b^t and the flow variables f_{pij}^t .

We consider two different levels of fixing: S/D; and S/D + Flow. Under S/D, we fix only the block start and depletion variables for blocks that do not belong the neighbourhood N . Under S/D + Flow, which we abbreviate to S/D + F in our analysis of results, we fix the block start and depletion variables for blocks outside of N , and the flow variables for blocks that *do not belong to the same pit as any block in* N .

We anticipate that for most blocks, we will not be able to radically change the time at which they are mined in a single iteration of LNS. We define an *unfix window* UW such that for any block b whose decision variables are to be unfixed in an LNS solve, they are unfixed in the window $[S_b - UW, F_b + UW]$ where S_b denotes the period in which b is first extracted, and F_b the period in which it is depleted. If b is never fully depleted, or is never extracted at all, $F_b + UW$ is set to the end of the planning horizon. If b is never extracted, $S_b - UW$ is set to the start of the planning horizon. The purpose of the unfix window is to control the size of each LNS MIP.

6.3.3. Termination Criteria We define two criteria to determine when to terminate LNS. The first is a *time limit*, L_T , where LNS terminates after L_T seconds. The second is an *improvement rate*, L_{imp} . LNS terminates after a minimum number of iterations, L_{iter} , have been performed and the rate of solution improvement, R_{imp} , falls below $L_{imp}\%$. We define R_{imp} as the sum of percentage improvements across all iterations thus far, divided by the number of iterations.

7. Results

All experiments have been conducted on a machine with an Intel Xeon Platinum 8176 chip (2.1GHz), and 1TB of RAM. All MIPs are solved with Gurobi 9.11 using the following parameter settings: NumericFocus (1); MIPFocus (1); PrePasses (2); AggFill (200); and Heuristics (0.25). All other parameters are set to their default values. Through informal experimentation, this parameter setting was found to elicit the best performance from Gurobi when solving the full MIP models of our three problem instances.

We first evaluate our sliding windows method for generating initial feasible solutions across varying window size (W), relaxed horizon length (H), and overlap (O). Tables 6–8 report the quality of solutions found by sliding windows, in terms of their gap to either a known optimal solution or a best known upper bound, for the T1, OT, and Pilbara models. We vary W from 1 to 7, H from 0 to 4, and O from 0 to 2. A ‘–’ indicates either that the combination of parameter values could not be used in conjunction, or that sliding windows did not yield a feasible solution. For a given window size and horizon length, increasing the overlap generally results in a better initial solution at the cost of increased run times. Increasing the window size generally results in a better initial solution at the cost of increased run times. Controlling the window size appears to be the most effective method for achieving a desired trade off between solution quality and solve time.

Table 6 T1: Quality of solutions formed by sliding windows, recording the gap % to the known optimal solution and the time (s) required to find these solutions for varying window size (W), relaxed horizon length (H), and overlap (O). MIPs are terminated at a MIP gap of 0.1%.

W	$H = 0, O = 0$		$H = 0, O = 1$		$H = 0, O = 2$		$H = 2, O = 1$		$H = 4, O = 1$	
	Gap (%)	Time (s)								
1	20.3	15	–	–	–	–	–	–	–	–
2	20.2	11	18.2	26	–	–	17.2	34	17.2	42
3	16.9	10	16.6	18	16.1	27	17.2	32	17.2	38
4	19.7	10	19	23	14.1	20	7.1	31	8.3	44
5	16.2	11	7.5	25	6.2	25	8.5	47	7.5	46
6	17.6	13	13.9	28	8.5	35	15.1	46	15.1	68
7	6.5	21	8	39	13.9	35	7.1	61	7.1	75

Table 7 OT: Quality of solutions formed by sliding windows, recording the gap % to the best bound found after 148 hours of solving the full model, and the time (s) required to find these solutions for varying W , H , and O . MIPs are terminated at a MIP gap of 0.1%.

W	$H = 0, O = 0$		$H = 0, O = 1$		$H = 0, O = 2$		$H = 2, O = 1$		$H = 4, O = 1$	
	Gap (%)	Time (s)								
1	99.9	152	–	–	–	–	–	–	–	–
2	99.9	110	99.9	150	–	–	99.9	162	37.7	168
3	99.9	97	99.9	112	99.9	144	99.9	122	37.8	125
4	99.9	92	99.9	98	99.9	103	48.3	105	27.1	108
5	68.3	85	59.6	92	48.4	90	27.1	98	27.1	102
6	27.1	87	27.1	85	27.1	84	27.1	89	27.1	88
7	27.1	78	27.1	87	27.1	87	27.1	90	27.1	90

Table 8 Pilbara: Quality of solutions formed by sliding windows, recording the gap % to the best bound found after 148 hours of solving the full model, and the time (ks, 1000s) required to find these solutions for varying W , H , and O . MIPs are terminated at a MIP gap of 10%, or after 5000s.

W	$H = 0, O = 0$		$H = 0, O = 1$		$H = 0, O = 2$		$H = 2, O = 1$		$H = 4, O = 1$	
	Gap (%)	Time (ks)								
1	12.2	1.9	–	–	–	–	–	–	–	–
2	9.9	2.1	8.9	3.1	–	–	9.6	15.6	–	–
3	9.3	2.7	8.9	3.0	8.2	7.2	8.3	14.3	–	–
4	9.6	4.1	8.2	4.9	9.6	7.3	–	–	–	–

7.1. T1

Table 9 records the quality of solutions found by LNS for model T1, across varying neighbourhood sizes (number of blocks), focal block selection methods (Minimum dependencies-MD, random selection-RAND, objective contribution-based selection-OBJ, and mixed criteria-MIX), and variable fixing strategies (S/D and S/D+F). Recall that the S/D fixing method fixes the values of block start and depletion time binary variables for blocks outside of our neighbourhood, while S/D+F additionally fixes the value of the material flow variables for those blocks. Also recall that to form our path-based neighbourhoods, we repeatedly select a focal block, using a chosen strategy, form of path of blocks around that focal block, and add that path to our neighbourhood. The three key strategies for selecting focal blocks alternately select blocks randomly, favour blocks with a higher contribution to the objective function, or with fewer predecessors and successors (minimum dependencies). The mixed criteria setting alternates between using these three methods when forming

Table 9 T1: Average time (s) spent by LNS, and average quality of solutions found (% gap to known optimal) across 10 differently seeded runs, each over 10 threads. Initial solution with a gap of 16.6% obtained by sliding windows ($H = 0, O = 1, W = 3$). Solutions compared given varying focal block selection methods (MD, MIX, OBJ, and RAND), varying neighbourhood size (N), differing variable fixing methods (S/D and S/D + F), and use of RINS (Yes/No). LNS MIPs solved to a gap of 0.1%. LNS is terminated when the improvement rate falls below 1%. Best performing settings are in bold. Final row starts with a initial solution with a gap of 20.3% ($H = 0, O = 0, W = 1$).

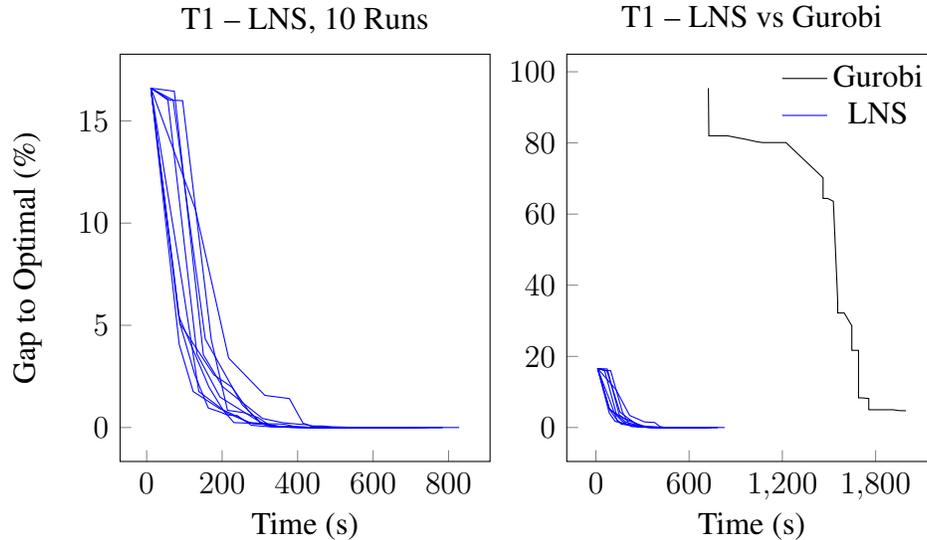
N	Focal Block	Use of RINS	Fixing Method	Avg Time (Min, Max) (s)	Avg Gap (Min, Max) (%)
Initial solution: gap of 16.6%, sliding windows ($H = 0, W = 1, O = 0$)					
100s MIP time limit					
10	MD	Yes	S/D	126, (114,137)	15.6 (15.5,15.9)
20	MD	Yes	S/D	597, (479,683)	0.1 (0,0.3)
30	MD	Yes	S/D	734, (683,817)	0 (0,0)
40	MD	Yes	S/D	1350, (1230,1464)	0 (0,0)
30	MD	Yes	S/D + F	740 (686,817)	0 (0,0)
30	MIX	Yes	S/D	676, (620,736)	0 (0,0)
30	OBJ	Yes	S/D	410, (360,445)	16.1 (15.5,16.3)
30	RAND	Yes	S/D	707, (647,807)	0 (0,0)
30	MD	No	S/D	1022, (730,1619)	10 (0,15.6)
200s MIP time limit					
30	MD	No	S/D	1562, (1226,1928)	0 (0,0)
100s MIP time limit, <i>Timing</i> neighbourhood formation strategy applied					
30	MD	Yes	S/D	716, (645,897)	0 (0,0)
Initial solution: gap of 20.3%, sliding windows ($H = 0, W = 1, O = 0$)					
100s MIP time limit					
30	MD	Yes	S/D	811, (725,977)	0 (0,0)

neighbourhoods across the threads of computation used by LNS. We also consider the impact of the RINS-based strategy of Section 6.3.1 when solving each MIP during LNS.

For each parameter setting considered in Table 9, we record the average quality of solutions found, in terms of their gap to a known optimal solution, across 10 differently seeded LNS runs. We start LNS from two different initial solutions, one with a gap of 16.6% to the known optimal (the first 11 entries in Table 9), and the second with a gap of 20.3% (the last entry in Table 9). The purpose of this is to analyse how the performance of LNS changes when it starts from solutions that differ in quality. The LNS explores 10 neighbourhoods in parallel, and is terminated when the improvement rate falls below 1%. Each MIP solve is afforded either 100 or 200 seconds, and is terminated at a gap of 0.1%. Best performing settings are in bold. Alongside the average quality of solutions, we report the quality of the best and worst solution found across the 10 runs, and the average, minimum, and maximum solve time. These solve times do not include the time spent by sliding windows to form an initial feasible solution. For T1, we set the unfix window UW to the length of the planning horizon.

Using objective contribution to guide focal block selection during LNS is a poor choice for T1 as the LNS gets stuck in a local optima. However, using minimum dependencies, random selection, or a mixture of criteria (including objective contribution), avoids this problem. Fixing flow variables in addition to block start and depletion binaries has no discernable benefit for T1. The RINS strategy of solving the LP relaxation of each MIP, and then fixing the integer variables that

Figure 2 T1: Gap to optimal for solutions found during 10 differently seeded runs of LNS, over time, using the settings shown in the first bold entry of Table 9 and 10 LNS threads. Run time is inclusive of the time spent during sliding windows (with $H = 0$, $O = 1$, and $W = 3$). Starting feasible solution had a gap of 16.6%. LNS is compared against the performance of Gurobi on the same model.



take on the same value across the LP relaxation solution and the current best solution found during LNS, appears to be effective at reducing overall run times. Where RINS was not applied, we had to increase the time limit afforded to each MIP solve from 100s to 200s in order to find the optimal solution. Without RINS, and with the increased MIP time limits, we more than double the LNS run time.

Increasing neighbourhood size has the effect of both increasing the run time of LNS, but increasing the quality of the final solution. A neighbourhood size of 30 blocks allows us to find the optimal solution across all 10 runs when using an appropriate focal block selection strategy and RINS. Of the proposed neighbourhood formation strategies, only Timing is applicable. The performance of LNS when switching between the use of no strategy, and Timing, across the 10 LNS threads (five use no strategy, five use Timing), and applying no strategy across all 10 threads, is similar (entries 3 and 11 in Table 9). Most of the results in Table 9 have started with a solution with a gap of 16.6%. The last row starts with a solution with a gap of 20.3%. LNS is still able to find the optimal solution, yet requires more time.

Figure 2 compares the performance of LNS, using the settings shown in the first bold entry of Table 9, and 10 LNS threads, against solving the T1 model directly with Gurobi. The parameter settings used for the full model solve are defined at the start of Section 7. These settings were chosen to extract the best performance from Gurobi when solving the full model. We plot the gap between the solutions found during LNS to the known optimal solution, over the 10 differently seeded runs, against those found by Gurobi over time. In the plots of LNS solution quality over time, run time *is inclusive* of the time spent during sliding windows to form the first solution.

7.2. OT

Table 10 reports the average quality of solutions found by LNS to the OT model across varying neighbourhood sizes, focal block selection methods, variable fixing strategies, and neighbourhood formation strategies. We report solution quality in terms of the gap to the best known upper bound on the objective of the model, found after solving the full model with Gurobi for 148 hours. For OT, we set the unfix window UW to the length of the planning horizon. OT has blending constraints,

Table 10 OT: Average time (ks) spent by LNS, and average quality of solutions found (gap to best upper bound found after 148 hours of solving the full model) across 10 seeded runs, each over 10 threads. Initial solution with a gap of 27.1% found by sliding windows ($H = 2, O = 1, W = 5$). Columns are defined as per Table 9. LNS MIPs are terminated at a gap of 0.05%, or after 600s. LNS is terminated after 12 hrs or when the improvement rate fell below 1%. Final row starts with a initial solution with a gap of 99.9% ($H = 0, O = 0, W = 4$). Best settings are in bold.

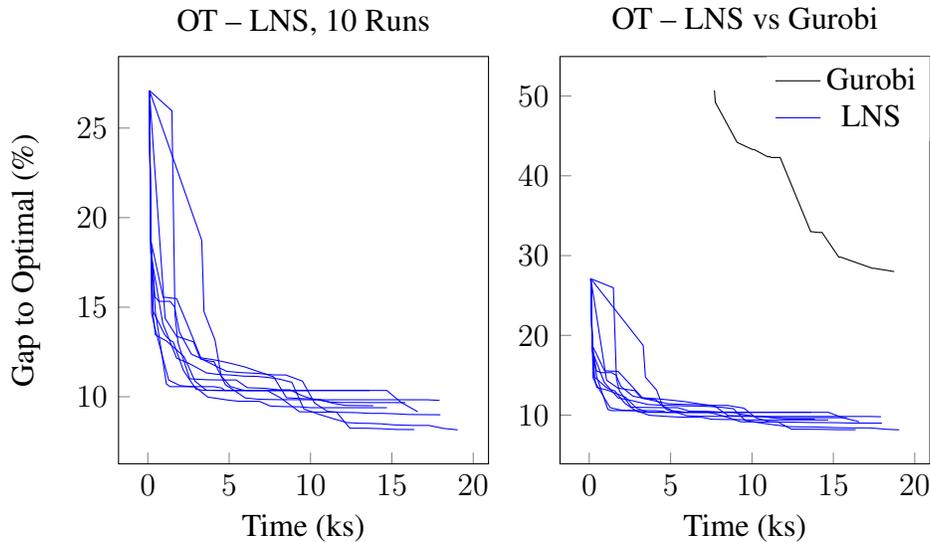
N	Focal Block	Use of RINS	Fixing Method	Avg Time (Min, Max) (s)	Avg Gap (Min, Max) (%)
Neighbourhood formation strategy: <i>Timing and Blending</i>					
50	OBJ	Yes	S/D	2.3, (1.5,3.0)	17.5 (12.5,26.0)
100	OBJ	Yes	S/D	15.5, (9.7,18.9)	10.3 (9.1,11.3)
100	OBJ	Yes	S/D + F	15.1, (11.2,20.3)	10.7 (8.9,11.3)
100	OBJ	No	S/D	12.3, (5.5,19.1)	13.0 (9.1,24.0)
150	OBJ	Yes	S/D	22.7, (20.6,27.7)	10.5 (9.0,11.5)
100	MD	Yes	S/D	19.4, (17.4,21.2)	9.9 (7.8,11.2)
100	RAND	Yes	S/D	20.3, (18,23.5)	9.6 (7.9,10.8)
100	MIX	Yes	S/D	18.9, (16.8,21.2)	10.0 (8.4,10.7)
Neighbourhood formation strategy: <i>Timing only</i>					
100	OBJ	Yes	S/D	8, (5,17.5)	11.4 (10.2,13.1)
Neighbourhood formation strategy: <i>Blending only</i>					
100	OBJ	Yes	S/D	17.1, (12.7,21.3)	10.5 (8.8,12.0)
Neighbourhood formation strategy: <i>None</i>					
100	OBJ	Yes	S/D	5.6, (1.7,10.6)	13.9 (11.1,26.0)
Neighbourhood formation strategy: <i>Timing and Blending</i> Sliding windows ($H = 0, W = 4, O = 0$)					
100	OBJ	Yes	S/D	43.7, (43.3,44.3)	9.8 (8.2,11.2)

but no minimum production constraints. Consequently, both the Timing and Blending strategies are applicable. We contrast the performance of LNS when either, or both, of these strategies are applied. Most entries in Table 10 start with an initial solution with a gap of 27.1%, obtained by sliding windows with parameters $H = 2, O = 1$, and $W = 5$. The last entry in Table 10 applies LNS to a starting solution with a gap of 99.9%, obtained by sliding windows using $H = 0, O = 0$, and $W = 4$. LNS MIPs are solved to a gap of 0.05%, or to a time limit of 600s. LNS is terminated after either 12 hours or when the improvement rate falls below 1%. Best settings are in bold.

As with T1, fixing flow variables, in addition to block start and depletion binaries, does not appear to have a discernable benefit. Most entries in Table 10 involve use of the RINS strategy when solving LNS MIPs. Without this strategy (the fifth entry in Table 10), MIP solves are more likely to be cut off before reaching a desired gap, given the available time limit (of 600s). This results in a smaller improvement rate over time, and in general an earlier termination. The first 8 entries of Table 10 involve both the Timing and Blending strategies being used. Entries 9-11 show the average quality of solutions found when using only Timing, only Blending, or neither. It seems that the Blending strategy has most benefit for this instance, although applying both strategies appears to be better than applying just one or neither. Although using a focal block selection method other than OBJ seems to find solutions with slightly smaller gaps to the best known bound, on average, this comes at a cost of increased runtimes.

Figure 3 compares the performance of LNS, using the settings shown in the first bold entry of Table 10, and 10 LNS threads, against solving the OT model directly with Gurobi.

Figure 3 OT: Optimality gap over 10 differently seeded runs of LNS, over time, using the settings shown in the first bold entry of Table 10 and 10 LNS threads. Run time is inclusive of the time spent during sliding windows (with $H = 2$, $O = 1$, and $W = 5$). Starting feasible solution had a gap of 27.1%. LNS is compared against a full solve with Gurobi. The full solve reaches a gap of 14.9% after 72 hours of solving. All five LNS solves reach a gap of under 11% within 5.5 hours of solving.



7.3. Pilbara

Solving the Pilbara model with Gurobi does not yield a feasible solution after 7 days of solve time. We compare the quality of solutions found by LNS with the best upper bound on the optimal objective found after solving the full model for 148 hours. The Pilbara model contains blending constraints, capex decisions, and minimum production constraints. All the neighbourhood formation strategies of Table 4 are applicable.

Table 11 reports the average quality of solutions found by LNS to the Pilbara model across varying parameter settings. We use a fixed neighbourhood size of 200 blocks across all experiments. For T1 and OT, we did not utilise the unfix window strategy described in Section 6.3.2. We were able to leave variables related to the mining of blocks in our neighbourhood unfixed across the entire planning horizon. As the Pilbara model is substantially larger than T1 and OT, we find that setting a small unfix window (UW) is required to keep each MIP solve manageable. For each parameter setting, we report the average quality of solutions found by LNS, in terms of their gap to the best known upper bound on the optimal objective value, over 10 differently seeded runs, each utilising 10 threads of computation. LNS MIPs were solved to a gap of 0.05%, or a time limit of 400s. Each LNS run was terminated after 12 hours. The entries in Table 11 have started with an initial solution with a gap of 8.2%, found by sliding windows with $H = 0$, $O = 1$ and $W = 4$. The best settings have been highlighted in bold.

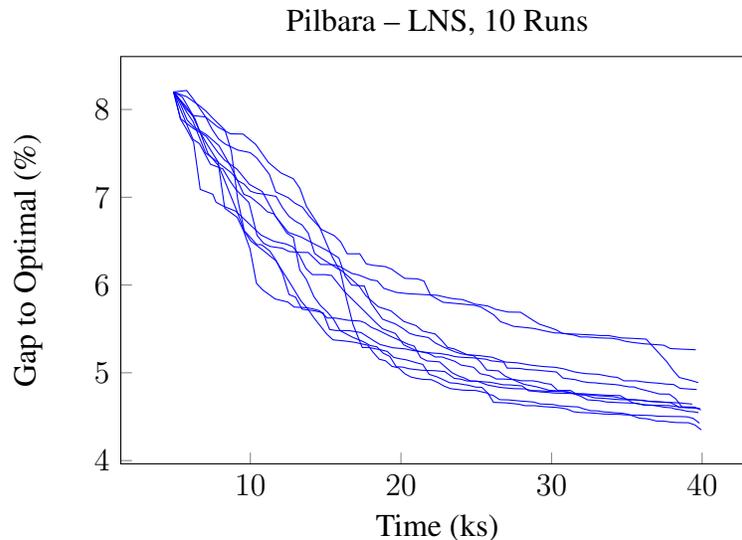
In contrast to T1 and OT, use of the RINS solving strategy was not helpful for the Pilbara model (contrast entries 1 and 3 in Table 11). The cost of two solves for each LNS MIP under the RINS strategy (an LP relaxation, and then a further restricted MIP) was higher than a single solve of the original LNS MIP. Moreover, we need to fix the additional flow variables to keep each LNS MIP at a manageable size (contrast entries 1 and 2 in Table 11). For T1 and OT, there was no substantial difference in the quality of solutions obtained with the two variable fixing strategies. Entries 1 and 4 in Table 11 highlight the impact of increasing the unfix window from 2 to 4.

The Trigger neighbourhood formation strategy appears to be the most effective for the Pilbara model. Entries 5-7 in Table 11 contrast the performance of LNS when only the Trigger, Timing

Table 11 Pilbara: Average time (ks) spent by LNS, and average quality of solutions found (% gap to best upper bound found after 148 hours of solving the full model) across 10 seeded runs, each over 10 threads. Columns are defined as per Table 9. Initial solution with a gap of 8.2% found by sliding windows ($H = 0$, $O = 1$, and $W = 4$). Neighbourhood size of 200 blocks. LNS MIPs terminated at a gap of 0.05%, or after 400s. LNS is terminated after 12 hrs. Best settings are in bold.

Neighbourhood Strategies Used	Focal Block Selection	Fixing Method	Use of RINS	Unfix Window (UW)	Avg Gap (Min, Max) (%)
None	OBJ	S/D+F	No	2	4.7 (4.5, 4.9)
None	OBJ	S/D	No	2	8.2 (8.2, 8.2)
None	OBJ	S/D+F	Yes	2	4.8 (4.4, 5.0)
None	OBJ	S/D+F	No	4	4.8 (4.6, 5.1)
Trigger	OBJ	S/D+F	No	2	4.5 (4.4, 4.6)
Timing, Blending	OBJ	S/D+F	No	2	4.7 (4.6, 4.9)
Pit Links, Timing	OBJ	S/D+F	No	2	4.6 (4.3, 4.9)
All	OBJ	S/D+F	No	2	4.5 (4.2, 4.7)
– Blending	OBJ	S/D+F	No	2	4.5 (4.2, 4.8)
– Pit Links	OBJ	S/D+F	No	2	4.5 (4.3, 4.8)
All	MD	S/D+F	No	2	4.5 (4.3, 4.8)
All	RAND	S/D+F	No	2	4.5 (4.2, 4.7)
All	MIX	S/D+F	No	2	4.5 (4.3, 4.8)

Figure 4 Pilbara: Optimality gap over 10 differently seeded runs of LNS, over time, using the settings shown in the first bold entry of Table 11 and 10 LNS threads. Run time is inclusive of the time spent during sliding windows (with $H = 0$, $O = 1$, and $W = 4$).



and Blending, and Timing and Pit Links strategies have been applied, respectively. The difference in the quality of solutions obtained when using the Trigger strategy versus not using any strategy (average gaps of 4.5 versus 4.7) was statistically significant (Wilcoxon signed rank test applied to the two populations of solutions: $p = 0.027$). The Blending strategy was least effective, resulting in no improvement, on average, in solution quality relative to using no strategy. Entry 8 in Table 11 reports the average quality of solutions found when applying all four neighbourhood formation strategies. Entries 9 and 10 consider the performance of LNS when we use all strategies *but* Blend-

ing and Pit Links, respectively. It is clear that the inclusion of the less effective strategies, such as Blending, does not compromise the performance of LNS.

The performance of LNS under objective, minimum dependencies, random, and mixed criteria focal block selection was similar. This is evident by comparing entries 8 and 11-13 in Table 11. Figure 4 shows the performance of LNS, using the settings shown in the first bold entry of Table 11.

8. Conclusion

We have presented a Large Neighbourhood Search (LNS) based method for solving long-term open-pit mine planning problems, demonstrating the effectiveness of the approach on three real world mining projects of varying complexity. We have proposed a novel path-based neighbourhood structure, and a range of neighbourhood formation strategies, designed to maximise the likelihood that our neighbourhoods will lead us to improved, feasible solutions. Our LNS-based method is able to find, within hours, near optimal solutions to planning problems that cannot be solved with an off-the-shelf solver in a reasonable time, or with reasonable computational resources. Future work will consider how our LNS approach performs on a stochastic two-stage version of our long-term mine planning problem. In this context, new neighbourhood formation strategies may be possible that specifically target stochastic elements of the model.

References

- Amaya, J., Espinoza, D., Goycoolea, M., Moreno, E., Prevost, T., and Rubio, E. (2009). A Scalable Approach to Optimal Block Scheduling. In *APCOM*, pages 567–575.
- Askari-Nasab, H., Pourrahimian, Y., Ben-Awuah, E., and Kalantari, S. (2011). Mixed Integer Linear Programming Formulations for Open Pit Production Scheduling. *Journal of Mining Science*, 47:338–359.
- Chicoisne, R., Espinoza, D., Goycoolea, M., Moreno, E., and Rubio, E. (2012). A New Algorithm for the Open-Pit Mine Production Scheduling Problem. *Operations Research*, 60(3):517–528.
- Cullenbine, C., Wood, R. K., and Newman, A. (2011). A sliding time window heuristic for open pit mine block sequencing. *Optimization Letters*, 5:365–377.
- Danna, E., Rothberg, E., and Pape, C. L. (2005). Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, 102(1):71–90.
- Epstein, R., Goic, M., Weintraub, A., Catalan, J., Santibanez, P., Urrutia, R., Cancino, R., Gaete, S., Aguayo, A., and Caro, F. (2012). Optimizing Long-Term Production Plans in Underground and Open-Pit Copper Mines. *Operations Research*, 60:4–17.
- Fathollahzadeh, K., Asad, M. W. A., Mardaneh, E., and Cigla, M. (2021). Review of solution methodologies for open pit mine production scheduling problem. *Int. J. Min. Reclam. Environ.*, 35(8):564–599.
- Gershon, M. (1987). Heuristic approaches for mine planning and production scheduling. *International Journal of Mining and Geological Engineering*, 5(1):1–13.
- Goodfellow, R. C. and Dimitrakopoulos, R. (2016). Global optimization of open pit mining complexes with uncertainty. *Appl. Soft Comput.*, 40:292–304.
- Lambert, W. B., Brickey, A., Newman, A. M., and Eurek, K. (2014). Open-Pit Block-Sequencing Formulations: A Tutorial. *Interfaces*, 44(2):127–142.
- Lamghari, A. (2017). Mine planning and oil field development: A survey and research potentials. *Math. Geosci.*, 49(3):395–437.
- Lamghari, A. and Dimitrakopoulos, R. (2012). A diversified tabu search approach for the open-pit mine production scheduling problem with metal uncertainty. *European Journal of Operational Research*, 222(3):642–652.
- Lamghari, A. and Dimitrakopoulos, R. (2016). Network-flow based algorithms for scheduling production in multi-processor open-pit mines accounting for metal uncertainty. *European Journal of Operational Research*, 250(1):273–290.
- Lamghari, A. and Dimitrakopoulos, R. (2020). Hyper-heuristic approaches for strategic mine planning under uncertainty. *Computers and Operations Research*, 115:104590.
- Lamghari, A. and Dimitrakopoulos, R. (2022). An adaptive large neighborhood search heuristic to optimize mineral value chains under metal and material type uncertainty. *International Journal of Mining, Reclamation and Environment*, 36(1):1–25.
- Lamghari, A., Dimitrakopoulos, R., and Ferland, J. A. (2015). A hybrid method based on linear programming and variable neighborhood descent for scheduling production in open-pit mines. *Journal of Global Optimization*, 63:555–582.
- Newman, A. M., Rubio, E., Caro, R., Weintraub, A., and Eurek, K. (2010). A Review of Operations Research in Mine Planning. *Interfaces*, 40:222–245.
- Osanloo, M., Gholamnejad, J., and Karimi, B. (2008). Long-term open pit mine production planning: a review of models and algorithms. *International Journal of Mining, Reclamation, and Environment*, 22:3–35.
- Sari, Y. A. and Kumral, M. (2016). An improved meta-heuristic approach to extraction sequencing and block routing. *J. South Afr. Inst. Min. Metall.*, 116(7):673–680.
- Senécal, R. and Dimitrakopoulos, R. (2020). Long-term mine production scheduling with multiple processing destinations under mineral supply uncertainty, based on multi-neighbourhood tabu search. *International Journal of Mining, Reclamation and Environment*, 34(7):459–475.

Zeng, L., Liu, S. Q., Kozan, E., Corry, P., and Masoud, M. (2021). A comprehensive interdisciplinary review of mine supply chain management. *Resources Policy*, 74:102274.