Heuristics for integrated blending optimisation in a mining supply chain

Zhou Haonan (a), Mehran Samavati (b)* and Andrew Hill (a)

(a) Rio Tinto Centre for Mine Automation, Australian Centre for Field Robotics, The University of Sydney, Sydney, Australia

(a) address: 8 Little Queen Street, Chippendale, New South Wales, Australia

(b) Management School, University of Sheffield, Sheffield, UK

(b) address: Sheffield University Management School, Conduit Road, S10 1FL, Sheffield, UK

ARTICLE INFO

Keywords:
Scheduling
Integrated blending optimisation
Open-pit mine optimisation
Decomposition-based algorithm
Topological sorting

ABSTRACT

In a mining supply chain, products from mines are blended at port terminals to ensure that a set of blending targets (such as grade and qualities) are achieved. The production scheduling problem of each individual mine and the blending problem for a network of mines and ports constitute the integrated blending optimisation, which involves modelling of material flows from mine-side pits to port-side stockpiles. Due to the problem scale and the bilinear constraints for blending behaviours, the problem is computationally hard to solve by any available optimisers. This paper extends upon a decomposition-based algorithm in the literature, which was first to solve the blending problem for a network of multiple mines and ports over multiple time periods. In our paper, a prune routine is proposed to progressively update the mixed integer program of the production scheduling problem for each mine during a rolling-horizon heuristic. Experiments have shown that this extension produces solutions of higher quality than the original algorithm. Furthermore, a ranking-based topological sorting heuristic is presented for selecting units of mineral deposits, known as 'blocks'. Experiments have shown that the average computation time can be reduced by 75.97% when this heuristic is implemented. On top of these extensions, an adaptive algorithm is adopted from the decomposition-based algorithm, featuring faster convergence and higher solution quality at the same time. Comparing our results to the literature, our adaptive algorithm, on average, yields an improvement in solution quality by 12.67% while reducing computation time by 65.09%.

*Corresponding author
m.samavati@sheffield.ac.uk (M.S. (b))
ORCID(s): 0000-0002-3468-9155 (M.S. (b))

1. Introduction

In open-pit mining, mineral deposits close to the earth's surface are extracted, transported, processed and stored in stockpiles. Orebodies in a pit are divided into grids of 'blocks' for production scheduling purposes. Oftentimes, the blocks are equally sized for convenience of scheduling. Each block is assigned an estimate of its percent composition of metal grade and that of other quality attributes (such as impurities). The blocks are scheduled for extraction and processing as per a set of short-term production targets which stipulate the desired grade and quality attributes. The processed ores from multiple mines are then transported by trains to a set of port terminals where they are blended into port products, known as 'blends', before shipping to the downstream customers. At each port, blending of trainloads is necessary as the grade and quality attributes of mine-side products usually differ from mine to mine (even from plant to plant) and vary with time, all contributing to unsatisfying port-side products. An example of such a network structure is illustrated in Fig. 1, where blocks across mines and blends across ports are 'unfolded' in a uniform view. From origin blocks to final blends, materials can follow various paths passing or bypassing stockpiles while wastes are sent to dump locations. The integrated blending optimisation decides not only the material flows inside each mine site but also the suitable match between mine-side products and port-side products such that the final blends are in line with certain quality targets over time. In other words, such an optimisation problem deals with a network of multiple mines and multiple ports in a multiple-time-period setting, known as MTP-MMPP [6].

A block model is classified into two categories - grade block and blast block. As the term suggests, blasting is required for a blast block so that it become mineable. In contrast, a grade block is already blasted and ready for immediate extraction. In general, a blast block may contain a mix of grades (high and low) and wastes whereas a grade block can contain only one of the three. In practice, a blast block model contains a number of smaller sized grade blocks. For the purpose of planning, however, blast blocks can be treated as if they only differ from grade blocks in chemistry composition and tonnage. To account for the blasting requirement, blast blocks are delayed by a specified number of time periods, which is realised as a constraint in modelling.

The mining of blocks follows a particular order such that internal blocks are accessed by removing the blocks immediately accessible on the mining face, named 'face blocks'. This gives rise to a set of disjunctive precedence constraints - an internal block is accessible when at least one of its adjacent blocks is extracted completely. For instance in Fig. 2, blocks $\{b_1, b_2, b_3, b_4\}$ around block b_0 constitute the adjacent set of blocks, denoted as $\mathcal{A}_{b_0}$, that precede block b_0: to access b_0, at least one of the adjacent block in $\mathcal{A}_{b_0}$ needs to be removed in advance.

Another type of precedence constraint is from a vertical perspective: a specified number of conjunctive blocks immediately above a block must be removed completely before the underlying block. In general, there are two configurations of such a vertical precedence constraint that are being used in planning: Fig. 3 shows the first configuration in which the 5 blocks above the underlying block constitute an instance of vertical precedence constraint pertaining to underlying block; similarly, Fig. 4 displays another configuration in which four additional blocks are also regarded as vertically preceding the underlying block, increasing the total number of preceding blocks to 9.

Apart from the disjunctive and vertical precedence constraints, the precedence relations existing for a set of contiguous blocks must be considered in order to form a feasible extraction sequence of blocks - the separation constraints. Consider a set of contiguous blocks $\mathcal{C} = \{b_0, b_1, b_2, b_3\}$ and a set of face blocks $\mathcal{N}(\mathcal{C}) = \{b_4, b_5\}$ in Fig. 5: blocks in $\mathcal{C}$ are accessible when at least one of the face blocks in the neighbouring set $\mathcal{N}(\mathcal{C})$ is mined completely. In fact, such precedence relations must be respected for any subset $\mathcal{C}'$, where $\mathcal{C}' \subseteq \mathcal{C}$. For example, consider the subset of contiguous blocks $\mathcal{C}' = \{b_2, b_3\}$, supposing neither of b_0 and b_1 is mined completely in the extraction schedule: b_2 and b_3 are adjacent to each other and if they are mined completely during the same time period the disjunctive precedence constraint of a single internal block is not violated (they mutually satisfy each other's precedence requirement); however, the neighbouring set of face blocks $\mathcal{N}(\mathcal{C}')$ is empty, and hence none of the blocks in $\mathcal{C}' = \{b_2, b_3\}$ can be mined in this case. In reality, the total number of separation constraints is too large to be defined explicitly and hence they are a major source of complexity in modelling and optimisation.

For each mine, a production schedule is formed according to a set of short-term production targets which specify the desired grade and quality of each granularity in each time period. A production schedule determines the amount of ore to be extracted from each block during each time period and the associated destination. The extraction sequence in each time period must comply with all precedence constraints applicable. In addition to extraction, stockpiles are used to hold extracted ores that are not processed immediately due to processing capacity limits. Accordingly, reclamation of stacked ores is also part of the decision making in generating a production schedule. As for the destinations of extracted ore, a waste dump is used to store wastes generated during extraction; a stockpile holds either of high or low grade ores before processing; a dry plant splits high grade ores into fines and lumps; and a wet plant upgrades low grade ores to high grade before the splitting. Various operational constraints exist and must be taken into account such as capacities on extraction, transport, storage and processing. For efficiency, it is often desired to maximise waste dumping to fully utilise truck resources and to minimise the occurrence of double-handling - an event in which extracted ore is stacked on stockpiles and reclaimed for processing in the same time period. Many studies have contributed to the production scheduling problem for a single mine in the past. However, little attention was give to the global problem where the production of multiple mines are optimised at the same time.

Instead of defining explicitly all separation constraints,

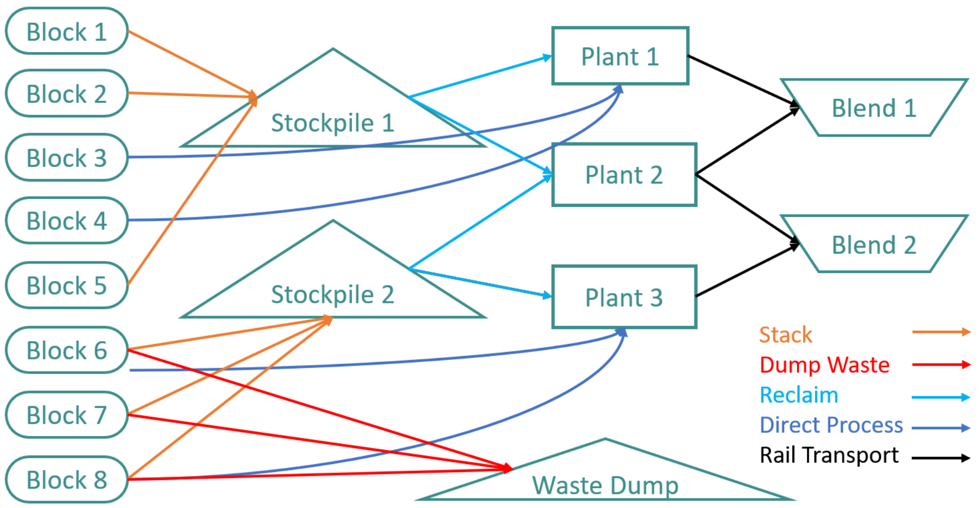

Figure 1: Network structure of open-pit mining supply chain

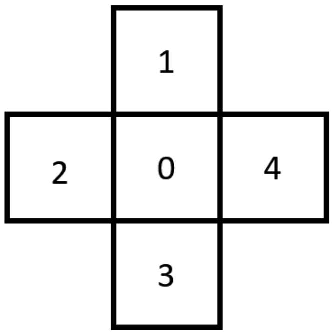

Figure 2: An adjacent set of blocks around a centre block

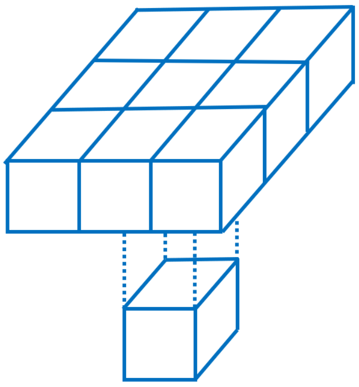

Figure 4: 9-block configuration of vertical precedence

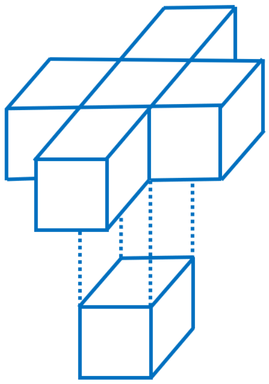

Figure 3: 5-block configuration of vertical precedence

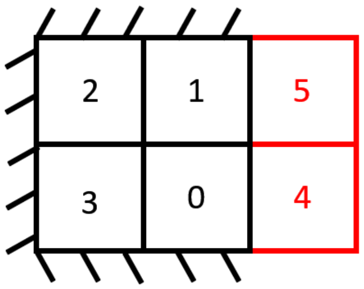

Figure 5: A set of blocks on the same bench

the authors in [6] utilised a delayed constraint generation method in which an inspection of solution feasibility is performed during the solution progress and any violated instances of separation constraints are added to the mixed integer program (MIP) model before the next solve. The rationale of this implementation is that it is often sufficient to find the optimal solution without going through all constraints applicable. The lazy callback function is supported by a few optimisers such as CPLEX and Gurobi. The downside of this approach is that the iterative solve process still can take very long to reach optimality. In this paper, we present an additional algorithm before optimiser solve to restrict the set of blocks available for mining. A ranked TopoSort heuristic is proposed to shortlist a set of blocks with the highest potential to meet the production targets. Results have shown that this modification greatly improves solve speed with only small impacts on solution quality.

The multiple-time-period setting of MTP-MMPP imposes another layer of difficulty - modelling of materials through intermediate junction nodes, i.e., stockpiles, require a set of bilinear constraints which makes the problem non-convex and hard to solve. In Blom's approach [6], the entire planning horizon is divided into two consecutive horizons, formulating a simpler MIP without those bilinear constraints. The solution of the first horizon, which always contains only a single time period, is fixed after solving and the two horizons move forward to include any unsolved time periods for the next round of solve. As such, the solution for multiple time periods is progressively solved in iterations, a method known as a rolling-horizon heuristic. This approach is subject to sub-optimal solutions compared to solving the problem in one holistic MIP. In our paper, a prune routine is proposed to augment the rolling-horizon heuristic such that the solution space of unsolved time periods is pruned with the knowledge of the solved time periods. Consequently, redundant decision variables regarding the completely mined blocks are removed and the set of constraints is also updated for formulating the next MIP. Results have shown that the prune routine leads to better solutions than the original rolling-horizon heuristic.

At a port terminal, ores departed from various mines are mixed up as per a blending scheme to achieve the desired grade and quality attributes, known as blending targets. The blending scheme determines the amount of ore should be sent from each mine in each time period and the target product at each port. The operational constraint in this case is the port capacity in handling the blending process in each time period. With the production schedule for each mine not known as a priori, the blending problem at each port becomes a harder to solve as the number of mines increases, which renders a direct approach prohibitively expensive.

The contribution of this study is described as follows: the decomposition-based algorithm (DA) in [6] is enhanced by introducing a prune routine to remove completely mined blocks and to update the precedence relation sets of remaining blocks during its rolling-horizon procedure; the aggregation technique used for block selection is replaced by a new ranked topological sorting heuristic (TopoSort); and the model in [6] is extended to account for quantity-based blending targets. Based on the test results of a range of designed scenarios, the enhanced version of DA shows a stronger capability in finding high quality solutions than the existing DA; and DA integrated with a new ranked TopoSort heuristic is the fasted method among all tested methods. Based on the aforementioned observations, a new Adaptive DA is proposed in which fast convergence is achieved via DA with TopoSort first and enhanced DA is activated once the algorithm reaches near convergence. In Section 5, it is demonstrated that the Adaptive DA outperforms the existing DA in solve time and solution quality. Furthermore, a range of test scenarios with extension to quantity-based blending targets are tested in multiple runs by Adaptive DA and the results demonstrate a good consistency of solution quality.

The remainder of this article is structured as follows. First, we identify the existing work related to integrated blending optimisation in Section 2. Section 3 describes the assumptions and modifications to the existing model in [6]. In Section 4, the framework of existing DA and its key components are described before the prune routine along with the enhanced DA are presented. An introduction to our ranked TopoSort heuristic is then presented along with algorithmic structure of our Adaptive DA. In Section 5, the experiment results by our Adaptive DA is presented and compared against existing DA in terms of solve time and solution quality. Concluding remarks in conjunction with an outlook to future work are provided in Section 6.

2. Related Work

While there are considerable studies in the field of operational research in open-pit mining [13, 14, 16, 17], the literature for the problem of blending optimisation in mining supply chains is narrow. These existing studies mainly focus on maximisation of net present value for a single open-pit mine instead of considering the blending targets in an entire supply chain.

The multiple-time-period multiple-mine planning problem (MTP-MMPP) is in nature a hybrid of two optimisation problems - mine-side production scheduling and mine-to-port schedule selection. In light of such a hybrid nature, Blom et al. [6] proposed the DA as the first method ever other than a direct approach to solve the MTP-MMPP. In DA, the blending problem is decomposed into two decision-making components. The input of one component becomes the output of the other component, forming a feedback loop structure. A set of production targets (grade and quality attributes), and a standard deviation for each of these targets, are initialised for each mine in each time period. Afterwards, multiple sets of production bounds are generated by randomly deviating the production targets. In the first component, the mine-side production scheduling problem of each mine is solved with respect to each of the generated production targets and multiple production schedules are generated as its output. For each mine, the composition of the mine products formed across this set of schedules is designed to form a normal distribution, with the given standard deviation, around the production target. Then these production schedules become the input of the second component in which the mine-to-port schedule selection problem is optimised. From the combined decision making, the best combination of production schedules is determined and a blending scheme that dictates the match between mine products and port products is obtained. Then the algorithm refines the production targets of each mine and varies the associated standard deviations by comparing the current solution to the best solution so far. This successive solving process is repeated until all standard deviations have converged to a value that falls below a predefined threshold.

For production scheduling in an open-pit mine, there are usually multiple objectives with hierarchical priorities. For instance, an objective of grade and quality target may have a strict precedence over other objectives such as productivity, resource utility, etc. Therefore, a hierarchical optimisation approach is often preferred. Blom et al. [3] presents a method to solve a multi-objective production scheduling problem for a single mine. The method utilises an optimise-and-prune approach in which a MIP is re-solved multiple times with respect to an ordered sequence of objectives and inferior solutions are pruned progressively. In a similar approach [6], a MIP is solved with respect to the primary objective regarding grade and quality targets; then the achieved grade and quality attributes are enforced as new hard constraints and the MIP is solved again with respect to the secondary objective regarding productivity. There are many difficulties in solving the MTP-MMPP as a holistic MIP. On one hand, modelling the time varying status of stockpiles introduces nonlinear constraints to the model. To avoid such complexity, Blom et al. [6] used a rolling-horizon heuristic. The rolling-horizon heuristic aggregates all time periods into two time windows, i.e., horizons in which the stockpile composition is regarded as constant; a MIP of these two horizons is optimised, after which the production schedule of the first horizon (contains only a single time period) is fixed and both horizons slide forward to include the remaining time periods. It is assumed that the status of each stockpile remains constant in each time horizon. In later work by Blom et al. [3, 4], variants of rolling-horizon heuristic are proposed with differing methods in dividing the planning horizon. The suitability of these methods in different scenarios are discussed.

On the other hand, the enormous number of blocks and their precedence relations in a real-world scenario is probably the biggest challenge in solving the MTP-MMPP. To reduce the size of large scale MIPs, Weintraub et al. [18] propose an approach to aggregate MIP models based on clustering analysis. The results show that the original MIP size is reduced significantly in a single mine setting. Extending this approach, Blom et al. [6] applies aggregation to scenarios of multiple mines and explore the extent to which aggregation size affects the solve time and solution quality. In their approach, an aggregate of blocks assumes the total tonnage of its members while an average value is taken of each grade and quality attributes. Correspondingly, the production scheduling for each mine takes a two-step approach - in the first step, a MIP is established and solved for aggregated blocks; then a new MIP is formed by the blocks in scheduled aggregates and a production schedule is obtained by solving the new MIP. The results show that a maximum aggregation size of 4 blocks provides a good balance between solve time and solution quality.

To deal with the computational difficulty caused by the enormous number of precedence constraints (especially by the separation constraints), studies usually take heuristic-based approaches. A constructive algorithm is presented by Mousavi et al. [11] to obtain a good initial solution for open-pit mine block sequencing. The same authors [12] propose a hybrid algorithm that combines branch-and-bound and simulated annealing in an attempt to obtain the optimum extraction sequences of original-size blocks. A topological sorting heuristic, named TopoSort, is presented by Chicoisne et al. [7] to obtain a starting feasible solution. Their work, however, is mainly focused on a class of open-pit mine production scheduling problem known as C-PIT, the goal of which is to maximise the net present value. Following TopoSort, a number of techniques based on it were proposed to tackle mining problems of extremely large instances. Samavati et al. [15] also uses the TopoSort heuristic in their attempt to solve C-PIT problems of very large size.

The integrated blending optimisation is not restricted to mining supply chains. Instead, it can be adapted to apply a wide range of production planning problems such as oil refinery production [1]. From an operational research point of view, it belongs to a class of extended pooling problems in which streams from sources to destinations (targets) are optimised by bypassing multiple layers of blending pools [2, 9, 10].

3. Modelling

3.1. Assumptions

There are a few assumptions made in the modelling of the MTP-MMPP and they are summarised as follows:

(1) The blending targets of each port product including the metal grade and other quality attributes are known as a priori, that is, these targets are deterministic and stay constant across the planning horizon.
(2) The ore split and chemistry composition in each stockpile is assumed to be constant during a time period and equal to the value at the start of that time period. However, their values in a following time period is updated once the extraction schedule in the preceding time period is determined.
(3) The transportation of produced ores in a time period from mines to ports must be completed by the end of next time period.

3.2. Mine-side Scheduling

At each mine site, the production scheduling problem $\mathcal{O}_m$ is formulated as a MIP with a pair of hierarchical objectives. The primary objective is to minimise the deviation present between the grade and quality attributes of produced ore and a set of production targets. As a lower priority, it is also desired to maximise the productivity across the planning horizon. The productivity is defined such that dumping waste has a positive contribution and that stockpiling has a positive contribution if and only if a wet processing plant exists for upgrading low grade ore.

Most of the MIP formulation of $\mathcal{O}_m$ in [6] remains unchanged with one exception - the constraint regarding the utility of processing plants. In the original formulation, each processing plant is constrained to be fully utilised within a tolerance ϵ, as shown in Eq. 1. The variable $x_{s,d}^{m,t}$ denotes the amount of material sent from source s to processing plant d in mine m in period t. The notation $\mathcal{S}_m$ represents the set of sources (blocks/stockpiles) in mine m. From Eq. 1, the amount of produced ore from each processing plant, denoted by d, in period t is restricted to be equal to its processing capacity C_d^m with a tolerance of ϵ. A potentially more flexible constraint is used in our work, given by Eq. 2. In the new constraint, the upper bound remains unchanged while the lower bound is specified by a multiplicative factor $\epsilon^- \in [0,1]$. Such modification is justified by the fact that it is not always possible to fully utilise a wet/dry processing plant due to the limited availability of low/high grade ores. For brevity, the full MIP formulation is omitted and interested readers are referred to [6] for details.

$$C_d^m - \epsilon \leq \sum_{s \in \mathcal{S}_m} x_{s,d}^{m,t} \leq C_d^m + \epsilon \tag{1}$$

$$\epsilon^- C_d^m \leq \sum_{s \in \mathcal{S}_m} x_{s,d}^{m,t} \leq C_d^m + \epsilon \tag{2}$$

$\mathcal{O}_m$ is solved in two steps - firstly, a MIP is formulated with an objective to minimise the total deviation present between the chemistry composition of mine products and a set of production targets, denoted by $\mathcal{O}_{m,1}$; secondly, the MIP is updated such that the achievable grade and quality figures are imposed as hard constraints and the objective is changed to maximise the total productivity, denoted by $\mathcal{O}_{m,2}$. Each production target is expressed as a pair of production bounds. Zero deviation is achieved when the pertaining quality attribute lies between the upper bound and lower bound.

3.3. Mine-to-port Schedule Selection

A random value is drawn from a normal distribution with a specified standard deviation, and this value is added to the production targets of each mine to form a new set of targets. A $\mathcal{O}_m$ is solved for a specified number of times with a different set of production targets each time so that multiple production schedules are generated for each mine. As such, the mine-product composition across the schedules produced for each $\mathcal{O}_m$ is designed to be normally distributed around the given production target, with a specified standard deviation, i.e., the size of the standard deviation controls the diversity of mine-products formed across the schedule set. This schedule set forms the input of a new MIP that determines the best combination of schedules across all mine sites. Hence, the new MIP is referred to as the mine-to-port schedule selection problem, denoted by $\mathcal{O}_\pi$, where port capacities and blending targets are taken into account as constraint and objective, respectively. Consequently, the best combination of schedules is determined such that the match between mine products and port products leads to correctly blended port products with lowest deviation from the blending targets. As a secondary objective, the overall productivity across all mine sites is to be maximised. In formulating $\mathcal{O}_\pi$, a much higher weighting is given to the deviation-based objective in a weighted sum of the two objectives.

The modelling of $\mathcal{O}_\pi$ remains unchanged as in [6] except the following extension to quantity-based blending targets. A new decision variable, denoted by $q_{n,l}^t$ where t, n and l are the index of time periods, products and granularities(i.e., lump and fines), respectively, is defined to represent the amount of blended product of each granularity formed in each time period. The expression of $q_{n,l}^t$ is given in Eq. 3. T_R denotes the tonnes of ore in a trainload. $r_{\pi,n,t}^{m,l,t',j}$ refers to the number of trainloads of ore in granularity l departing from mine m in period t' and blended into product n at port π in period t, as specificed in schedule j. $v_{g,l}^{m,t',j}$ denotes the main grade g in percent in the ore of granularity $l \in \mathcal{L}$ produced at mine m in period t of schedule j. Notations N, Π and M refer to the number of schedules, ports and mines, respectively. Eq. 4 is introduced in $\mathcal{O}_\pi$ enforcing the minimum production requirement for each port product of each granularity (denoted by $Q_{n,l}^*$ where $n \in \mathcal{N}$ and $l \in \mathcal{L}$ refer to products and granularities). The secondary objective of maximising productivity is changed to maximising total production of all desired products, denoted by $z_{\pi,2}$ in Eq. 5.

$$q_{n,l}^t = T_R \sum_{j=1}^{N} \sum_{\pi}^{\Pi} \sum_{m}^{M} \sum_{t'}^{t} r_{\pi,n,t}^{m,l,t',j} v_{g,l}^{m,t',j} \tag{3}$$

$$\sum_{t}^{T} q_{n,l}^t \geq Q_{n,l}^* \quad \forall n \in \mathcal{N}, l \in \mathcal{L} \tag{4}$$

$$z_{\pi,2} = \sum_{t}^{T} \sum_{n \in \mathcal{N}} \sum_{l \in \mathcal{L}} q_{n,l}^t \tag{5}$$

4. Methodology

Based on the framework of DA and its associated MIPs, new techniques and modifications are proposed in this paper with a goal to improve computational efficiency and solution quality. The following in this section briefly summarises the basic DA by Blom et al. [5, 6] and its major technical components, followed by a detailed description of our proposed DA and its new techniques.

4.1. DA

The framework of DA is illustrated in Fig. 6 in which the two sub-problems ($\mathcal{O}_m$ and $\mathcal{O}_\pi$) constitute a cyclic structure. A set of production targets ϕ, including grade and quality specifications, are initialised for each mine at the beginning. The values of these targets are either based on port-side blending targets or sourced from a medium-term plan. Plus, each production target is associated with a standard deviation σ, the value of which can only vary between a set of bounds. The bounds are specified as significant change Δ_q^+ and insignificant change Δ_q^-, where $q \in \mathcal{Q}$ refer to quality attributes. All σ are equal to Δ_q^+ initially whereas they will have converged to Δ_q^- by the termination of DA.

The mine-side component deals with the production scheduling for each mine via a staged solution process. At the first stage, blocks in a mine are amalgamated into larger size aggregates and a MIP of $\mathcal{O}_m$ is solved with respect to these aggregates. At the second stage, blocks in those schedules are selected to formulate a new MIP of $\mathcal{O}_m$, yielding the extraction schedule of blocks. The staged solution process is in place for reducing computation complexity induced by the

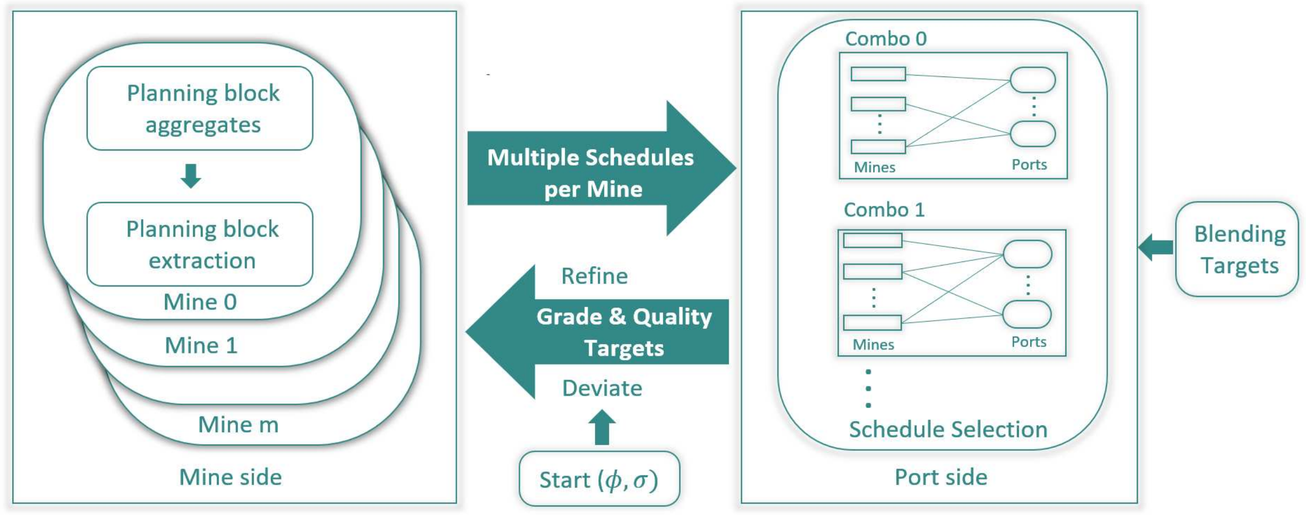

Figure 6: Framework of DA

large number of blocks encountered in practical mine planning. The output of the mine-side component is a pool of candidate schedules for each mine. In order to generate multiple schedules for each mine, a random deviation (drawn from a zero-mean normal distribution) is introduced to each ϕ every time the two-stage solution process is applied.

After the mine-side component, the candidate schedules are forwarded to the port-side component in which the best combination of schedules with respect to a set of predefined blending targets is determined by solving a MIP of $\mathcal{O}_\pi$. Subsequently, a feedback mechanism is in place to refine each ϕ and its corresponding σ such that the subsequent execution of the mine-side component is more likely to generate schedules that better suit the port-side blending targets. Afterwards, the aforementioned cycle is repeated until all σ have converged to the minimum values defined by Δ_q^-. In practice, a maximum number of iterations in conjunction with a maximum computation time are also implemented as additional stopping criteria. For a step-by-step description of DA, interested readers are referred to [6] for the details.

4.2. Aggregation Algorithm

In the original DA, a set of grade and quality targets in terms of upper and lower bounds are generated which are used to construct the quality-based objective function in the MIP formulation of $\mathcal{O}_m$. For brevity, interested readers are referred to [6] for the exact algorithm.

An aggregation algorithm is applied to form a set of aggregated blocks and a $\mathcal{O}_m$ is solved to schedule the mining of the aggregates. Then a second $\mathcal{O}_m$ is solved while being restricted to mining blocks within the aggregates chosen for mining in the first pass. As such, the overall complexity of dealing with an enormous set of blocks is relaxed. Since the authors in [6] did not specify the exact procedure for aggregation, a probably similar aggregation algorithm used in this paper is presented in Algorithm 1. There are two principles followed by the aggregation algorithm. Firstly, blocks residing on the same horizontal slice of earth, called a bench, are selected for aggregation. The intuition behind this principle is that blocks across benches are less likely to be mined in the same time period as compared to blocks on the same bench. In addition, aggregating vertically distant blocks should be avoided since the associated precedence constraints can be hard to satisfy. The second principle is related to the categorisation of blocks, including grade, blast, and pure waste. For ease and accuracy in estimating the aggregated chemistry compositions, only blocks of the same category are allowed for aggregation. For blocks in a bench, the procedure described in Algorithm 1 is applied to each category separately, the combined results of which constitute a set of aggregates for further processing.

In Algorithm 1, k is the index of aggregate and A_k is the aggregate being constructed. $\mathcal{B}$ denotes the set of blocks of same category in a bench and its complement set, namely blocks not in $\mathcal{B}$, is denoted by $\mathcal{B}^c$. Later on, already aggregated blocks are marked as prohibited for aggregation by adding them to $\mathcal{B}^c$. $\mathcal{N}(b)$ refers to blocks adjacent to block b. Besides, $\mathcal{F}$ and $\mathcal{F}_A$ denote blocks on the mining face and aggregates containing blocks on the mining face, respectively. These sets are initialised in step 2 and 3. The algorithm iteratively generates block aggregates and updates a lookup function $\mathcal{A}(A_k)$ that records the aggregated blocks. In step 6 and 7, a maximum number of aggregated blocks, denoted by M, is determined and it must be an integer in the closed interval $[1, M_A]$, where M_A dictates the size limit. Then a block b is randomly selected from $\mathcal{B}$. In step 9, blocks in the prohibited set, $\mathcal{B}^c$, are excluded from the set of neighbouring blocks of b, i.e., $\mathcal{N}(b)$. From step 10 to 14, an aggregate a is formed by blocks in $b \cup \mathcal{N}(b)$. The function $Aggregate(b, \mathcal{N}(b), M)$ in step 11 recursively builds up a with a cap of M blocks. The blocks now contained in a are removed from $\mathcal{B}$ in step 15, followed by addition to $\mathcal{B}^c$, i.e., they are labelled as not available for aggregation. From step 17 to 21, $\mathscr{A}(A_k)$, $\mathscr{F}_A$ and k are updated. The algorithm continues to form new aggregates till $\mathscr{B}$ becomes empty.

Algorithm 1 Aggregation Algorithm

```
1:  $k \leftarrow 0$, $A_k \leftarrow \emptyset$
2:  Initialise blocks in the same category $\mathscr{B}$, prohibited set
    $\mathscr{B}^c$, neighbouring set $\mathscr{N}(b)\ \forall b \in \mathscr{B}$ and face set $\mathscr{F}$
3:  Initialise face aggregate set: $\mathscr{F}_A \leftarrow \emptyset$
4:  Initialise aggregate lookup function $\mathscr{A}(A_k)$
5:  while $|\mathscr{B}| \neq 0$ do
6:      $M \leftarrow RandInteger(M_A)$
7:      $M \leftarrow \min(M, |\mathscr{B}|)$
8:      Randomly pick a block $b,\ b \in \mathscr{B}$
9:      $\mathscr{N}(b) \leftarrow \mathscr{N}(b) \backslash \mathscr{B}^c$
10:     if $|\mathscr{N}(b)| > 0$ then
11:         $A_k \leftarrow Aggregate(b, \mathscr{N}(b), M)$
12:     else
13:         $A_k \leftarrow b$
14:     end if
15:     Remove the blocks in $A_k$ from $\mathscr{B}$: $\mathscr{B} \leftarrow \mathscr{B} \backslash A_k$
16:     Mark the blocks in $A_k$ as prohibited: $\mathscr{B}^c \leftarrow \mathscr{B}^c \cup A_k$
17:     Save the aggregate $A_k$: $\mathscr{A}(A_k) \leftarrow \{b | b \in A_k\}$
18:     if any face blocks in $A_k$: $A_k \cap \mathscr{F} \neq \emptyset$ then
19:         Update the face aggregate set: $\mathscr{F}_A \leftarrow \mathscr{F}_A \cup A_k$
20:     end if
21:     $k \leftarrow k + 1$
22: end while
23: Return $\mathscr{A}(A_k)$ and $\mathscr{F}_A$
```

4.3. Rolling-Horizon Heuristic

A rolling-horizon heuristic is used in each mine-side problem for the generation of a mining schedule over a multiple-time-period planning horizon. The T time periods of a planning horizon are discretized into two aggregates of time horizons of size 1 and $T-1$. The first horizon, denoted by H_1, always contains only a single time period. And the second horizon, denoted by H_2, includes the remaining undetermined time periods of the mining schedule. Accordingly, a two-time-horizon MIP of $\mathscr{O}_m$ is formulated and solved, after which the mining schedule of H_1 is fixed and H_1 rolls forward to the next time period in queue (which is also the first time period in H_2). And H_2 is updated such that the time period now contained in H_1 is excluded. The two-time-horizon MIP of $\mathscr{O}_m$ is formulated and solved repeatedly as H_1 and H_2 move towards the last time period and null, respectively, at which point a single-time-period MIP of $\mathscr{O}_m$ is solved. As such, the mining schedule for all time periods is obtained. The details of this rolling horizon heuristic and its variants are described in [6] and [3], respectively.

4.4. Prune Routine and Enhanced DA

In between two consecutive iterations of the rolling-horizon heuristic, it is necessary to update the chemistry and tonnage within blocks and stockpiles before formulating a new MIP of $\mathscr{O}_m$. However, only updating the values of these time-varying data inputs as in [6] results in a successively redundant formulation of $\mathscr{O}_m$, in which 'ghost' blocks of empty tonnage exist, as time horizons move forward. To prevent ghost blocks from creation during rolling-horizon heuristic, a prune routine is proposed here to remove any empty blocks that will appear after fixing the schedule in H_1. Apart from removal of the ghost blocks, it is necessary to update the precedence relations of the remaining blocks. Algorithm 2 outlines the steps of the prune routine.

The inputs of prune routine include a set of blocks ($\mathscr{B}$), their precedence relations (disjunctive sets $\mathscr{A}^\vee$ and vertical sets $\mathscr{A}^\wedge$), a set of face blocks ($\mathscr{F}$), sets of internal blocks adjacent to the face blocks ($\mathscr{A}^\vee_\mathscr{F}$), a set of empty internal blocks ($\mathscr{I}^-$), and a set of empty face blocks ($\mathscr{F}^-$), as shown in the first line of Algorithm 2.

At start, a set of empty blocks to be removed is constructed by uniting $\mathscr{I}^-$ and $\mathscr{F}^-$. And a set of new face blocks, denoted by $\mathscr{F}_n$, is initialised with zero element. From step 3 to 15, each member in $\mathscr{B}^-$ is removed from $\mathscr{B}$, $\mathscr{A}^\vee$, $\mathscr{A}^\wedge$, $\mathscr{F}$ and $\mathscr{F}_n$. Meanwhile, blocks exposed to the mining face due to removal of preceding blocks are identified and added to $\mathscr{F}_n$, as shown in step 6 and 9. Notice that the sets pertaining to the empty blocks are also removed in step 7 and 10.

It is necessary to update the adjacent relations of face blocks after the removal process. By definition, an adjacent set of a face block contains only internal blocks, namely neighbouring face blocks are excluded. Accordingly, step 16 to 18 make sure that the new face blocks next to any remaining face block are excluded in the corresponding adjacent set. Then in step 19, any remaining face blocks and new face blocks are combined to form a new face block set. Note that the adjacent set of a face block is empty, i.e., $|\mathscr{A}^\vee(f)| = 0\ \ \forall f \in \mathscr{F}$. Accordingly, the steps from line 20 onwards ensure that the adjacent sets of new face blocks follow the aforementioned definitions.

It was found later in tests that this prune routine results in better solutions at a cost of solve time comparing to the original DA. Therefore, we name this new variant of DA Enhanced DA. Surprisingly, the observation of better solution and longer solve time is actually unexpected and contradicts with our intuition. Initially, the removal of redundant variables and constraints from a MIP formulation is expected to save some solve time by downsizing the solution space. Instead, the solve time increases and so does the solution quality in terms of total productivity. A speculation around this is DA itself is subject to sub-optimality due to the usage of the rolling-horizon heuristic and of the decomposition strategy. These techniques are used to simplify the nonlinear large-scale optimisation problem of MTP-MMPP into linear smaller-sized sub-problems that make up partial solutions to the entire problem. Although the prune routine may not affect the optimality of each individual sub-problem but the results have shown that it motivates Enhanced DA to search more extensively in the solution space. However, the exact mechanism of how the prune routine leads to better solutions has not been identified and requires further investigation.

Algorithm 2 Prune Routine

```
    Input: $\mathscr{B}$, $\mathscr{A}^\vee$, $\mathscr{A}^\wedge$, $\mathscr{F}$, $\mathscr{A}^\vee_{\mathscr{F}}$, $\mathscr{I}^-$, $\mathscr{F}^-$
    Output: Updated $\mathscr{B}$, $\mathscr{A}^\vee$, $\mathscr{A}^\wedge$, $\mathscr{F}$, $\mathscr{A}^\vee_{\mathscr{F}}$
 1: Initialise blocks to be removed: $\mathscr{B}^- \leftarrow \mathscr{I}^- \cup \mathscr{F}^-$
 2: Initialise new face blocks: $\mathscr{F}_n \leftarrow \emptyset$
    Remove empty blocks and identify new face blocks:
 3: for $b^- \in \mathscr{B}^-$ do
 4:     if $b^-$ is a face block then
 5:         Remove $b^-$ from set $\mathscr{F}$
 6:         Add each $b \in \mathscr{F}^\vee_{\mathscr{F}}(b^-)$ to $\mathscr{F}_n$ if $b \notin \mathscr{B}^-$
 7:         Remove set $\mathscr{A}^\vee_{\mathscr{F}}(b^-)$
 8:     else $b^-$ is an internal block:
 9:         Add each $b \in \mathscr{A}^\vee(b^-)$ to $\mathscr{F}_n$ if $b \notin \mathscr{F}, b \notin \mathscr{B}^-$
10:         Remove $b^-$ from sets $\mathscr{A}^\vee_{\mathscr{F}}(f)\ \forall f \in \mathscr{F}$
11:     end if
12:     Remove set $\mathscr{A}^\vee(b^-)$
13:     Remove $b^-$ from sets $\mathscr{A}^\vee(b)$ and $\mathscr{A}^\wedge(b)$, $\forall b \in \mathscr{B}$
14:     Remove $b^-$ from set $\mathscr{B}$
15: end for
    Update face blocks and their adjacent relations:
16: for each remaining face block $f \in \mathscr{F}$ do
17:     Remove blocks in $\mathscr{A}^\wedge_{\mathscr{F}}(f)$ that are now face blocks
18: end for
19: Update face set: $\mathscr{F} \leftarrow \mathscr{F} \cup \mathscr{F}_n$
20: for each new face block $f \in \mathscr{F}_n$ do
21:     Remove any face blocks in $\mathscr{A}^\vee_{\mathscr{F}}(f)$
22:     Neutralise disjunctive precedence of $f$: $\mathscr{A}^\vee(f) \leftarrow \emptyset$
23: end for
```

4.5. Ranked TopoSort Heuristic

In this section, we present a new topological sort heuristic in an attempt to reduce the computation time of DA. A topological ordering of blocks represents a feasible mining sequence that satisfies all precedence constraints. Such a sequence can be found by searching through a directed acyclic graph (DAG) that depicts the precedence relations existing between any given a set of blocks. The heuristic to generate a topological order is often referred to as TopoSort and is commonly utilised in many mining optimisation studies such as [7, 8, 15]. Our new TopoSort is equipped with a ranking mechanism to better suit the needs of blending.

The diagram in Fig. 7 shows an example of a DAG with 6 vertices, each denoting a block. An arrow is directed toward vertex i from vertex j if j precedes i in priority, representing a precedence relation between block i and j. In other words, the number of incoming arrows directed toward a given vertex indicate its degree of constraint. For example, block b_4 is preceded by blocks b_1 and b_2 while itself precedes block b_6 in Fig. 7 Notice that a DAG is not only able to represent vertical precedence relations between blocks but also can denote any existing horizontal precedence relations. Therefore, a realistic DAG of mining blocks is always in 3 dimensional space with a multitude of degrees of constraint. Initially, all vertices with zero degree of constraint are simply blocks on the mining face, i.e., $\mathscr{F}$ as mentioned previously. Picking any block from $\mathscr{F}$ does not break any precedence

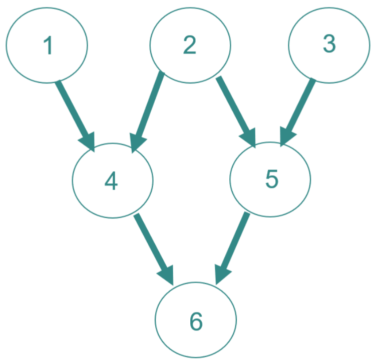

Figure 7: An example of DAG with 6 vertices/blocks

constraints. Hence, $\mathscr{F}$ is also called the eligibility set in the context of TopoSort.

As a typical way of selecting a vertex during TopoSort, a weight is computed and assigned to each vertex and the vertex of highest weight in $\mathscr{F}$ is selected with highest probability. A weight function in TopoSort is usually problem-specific and depends on the definition of the objective function. In many literature that included TopoSort, much emphasis has been given to maximisation of net present value whereas little attention has been given to grade and quality targets. Here, a new ranked TopoSort heuristic is proposed in which each vertex is assigned a rank representing its fitness with respect to a given set of grade and quality targets, denoted by $\hat{\phi}$. Since the objective of $\mathcal{O}_m$ is to minimise the total deviation present between chemistry of produced ores and $\hat{\phi}$, blocks of lower ranks (smaller deviations) are preferred.

Given a set of grade and quality targets $\hat{\phi}^{m,t}$ in period t at mine m, the corresponding set of qualities $\mathscr{Q}$ and the set of granularities $\mathscr{L}$, the rank of block $b \in \mathscr{F}$ is computed as in Eq. 6 - a weighted sum of deviation present between the chemistry of a block b and a set of quality targets.

$$
\begin{aligned}
Rank(b) = &\frac{1}{2} T_b^{hi} \sum_{l\in\mathscr{L}} \sum_{q\in\mathscr{Q}} \frac{S_{b,l}^{hi}}{\Delta_q^+} \Big[\Big| G_{b,l,q}^{hi} - \hat{\phi}_{l,q}^{m,t} \Big| \\
&+ \Big| G_{b,l,q}^{hi} - \hat{\phi}_{l,q}^{m,t+1} \Big| \Big] \\
&+ \frac{1}{2} T_b^{lo} \sum_{l\in\mathscr{L}} \sum_{q\in\mathscr{Q}} \frac{S_{b,l}^{lo}}{\Delta_q^+} \Big[\Big| G_{b,l,q}^{lo} \mathscr{R}_{l,q} / \mathscr{Y}_l - \hat{\phi}_{l,q}^{m,t} \Big| \\
&+ \Big| G_{b,l,q}^{hi} \mathscr{R}_{l,q} / \mathscr{Y}_l - \hat{\phi}_{l,q}^{m,t+1} \Big| \Big] \qquad (6)
\end{aligned}
$$

Specifically, the first half of the first double summation denotes the accumulated deviation incurred by mining the high grade portion of block b with respect to the grade and quality targets in time period t. And the second half computes the deviation with respect to the grade and quality targets in the next time period. Note that the deviation existing for each granularity l is weighted by the high grade tonnage of that granularity in b, indicated by the multipliers T_b^{hi} (tonnes of high grade ore in block b) and $S^{hi}_{b,l}$ (ore split of granularity l in block b). Furthermore, an adjusting factor, known as significant change Δ^{+}_{q}, is used as the denominator to account the fact that the deviations of different attributes may vary from each other by a few orders of magnitude. For instance, a difference of 0.01% between the target and actual level of phosphorus may be just as significant as a difference of 1.0% for iron. Similarly, the second double summation computes the accumulated deviation in terms of mining low grade portion of block b. The additional multiplier and divider inside the absolute operators account for wet processing where low grade ore is upgraded into a high grade concentrate. $\mathcal{R}_{l,q}$ denotes the recovery rate of quality q in granularity l in this concentrate, which is proportional to the input feed of low grade ore. And $\mathcal{Y}_l$ represents the portion of granularity l in the input feed that makes up the concentrate, known as yield factor. The values of these wet-processing parameters can be estimated from historical production data.

Given a set of blocks $\mathcal{B}$, a set of grade/quality targets $\hat{\phi}$ and relevant parameters including precedence relations (disjunctive sets $\mathcal{A}^{\vee}$ and vertical sets $\mathcal{A}^{\wedge}$), chemistry composition G, granularity split S,extraction capacity C_e and so forth, the Ranked TopoSort Heuristic in Algorithm 3 iteratively builds up a candidate set of blocks whose order is a feasible mining sequence, denoted by $\mathcal{T}$. The algorithm alternately selects blocks from an eligibility set $\mathcal{E}$ according to their ranks in each time period (indexed by t) and updates $\mathcal{E}$ until a sufficient number of blocks are chosen. The output $\mathcal{T}$ is a set of blocks selected for the formulation of mine-side scheduling problem $\mathcal{O}_m$.

In initialisation (step 1 and 2), the output set $\mathcal{T}$ is empty and the total tonnage of selected blocks T_e in time period $t = 1$ is zero. The while loop ensures that all blocks are selected in the worst case scenario. The eligibility set $\mathcal{E}$ is a subset of all face blocks ($f \in \mathcal{F}$) and it excludes those that are not immediately available due to vertical precedence constraints, i.e., $|\mathcal{A}^{\wedge}(f)| > 0$. $\mathcal{E}$ is reconstructed at the start of each iteration (step 4). From step 5 to 11, the current $\mathcal{E}$ is evaluated such that a maximum number of N_R blocks are selected for consideration of forming the output $\mathcal{T}$, where N_R takes even value only. Whenever the number of eligible blocks is greater than N_R, the first half of selected blocks are made up from the top ranked ones whereas the second half are chosen randomly. The partial random selection encourages the algorithm to look for a wide range of possibilities by investing in future eligible blocks instead of focusing too much on short-term gain. The resulting set of selected blocks is denoted by $\mathcal{P}$. The algorithm then starts from the first picked block and add it to the output set.

Before the algorithm proceeds its operation with each selected block, the accumulated tonnage T_e in time period t is calculated (step 13) and the time period index increments by 1 when 1.5 times the current extraction capacity C^t_e is exceeded (step 14). Such saturated inspection is also a measure to consider a wide range of blocks in addition to the partial selection in step 7. In fact, the number of total selected blocks in $\mathcal{T}$ by saturating the extraction capacity is almost always less than the total number of blocks in a real-world mine. Due to the saturated selection, some of the blocks may not be mined at all while some of them are mined either partially or completely across multiple time periods in the schedule formed by the rolling-horizon heuristic. Accordingly, Eq. 6 is formulated such that the rank of an eligible block depends on its average contribution to meeting production targets of two consecutive time periods. In addition, the algorithm will stop if all time periods have saturated with selected blocks (step 15). Otherwise, the accumulated tonnage T_e is reset to the currently selected block's tonnage and the algorithm continues.

In step 16, the selected block $p \in \mathcal{P}$ is added to the output set $\mathcal{T}$. Next in step 17, p is considered mined completely, and hence is removed from all relevant sets. Meanwhile, new blocks exposed to the mining face are identified (step 18) and added to the set of face blocks (step 19). Subsequently, the internal blocks adjacent to those new face blocks ($\mathcal{F}_n$) are labelled among which new face blocks in the next iteration will spawn (step 20 - 22). Additionally, any instances of new face blocks ($\mathcal{F}_n$) in the adjacent sets of face blocks $\mathcal{A}^{\vee}_{\mathcal{F}}$ are removed to be consistent with the definition (step 23 - 25).

4.6. DA with TopoSort

The Ranked TopoSort Heuristic (referred to as TopoSort hereafter) can be used as an alternative to the aggregation step in DA since both procedures serve to reduce the total number of blocks selected for solving production scheduling problem $\mathcal{O}_m$. Moreover, the block selection in DA with TopoSort is likely to be more efficient comparing to DA with aggregation where an additional MIP of $\mathcal{O}_m$ needs to be solved with randomly aggregated blocks. The algorithmic structure of solving $\mathcal{O}_m$ for each mine via DA with TopoSort is described in Algorithm 4.

To generate multiple production schedules for each mine, the iteratively changing production targets $\vec{\phi}^{i}_{m}$ are deviated by adding some random number (drawn from a zero-mean normal distribution the standard deviation of which is $\vec{\sigma}^{i}_{m}$) and a pair of production bounds $\left[L^{m,t}_{l,q}, U^{m,t}_{l,q}\right]$ are generated for each quality attribute in each granularity (step 5). The gap between the upper and lower bounds is exactly twice the value of a predefined insignificant change (Δ^{-}_{q}) in the corresponding quality attribute ($q \in \mathcal{Q}$). Details of this bound generation procedure is described in Algorithm 2 in [6]. As a slight change to the original procedure, the deviated production targets ($\hat{\phi}$) are retained for rank assignment in TopoSort. Then a number of suitable blocks are selected via TopoSort (step 6), followed by a data input preparation routine (step 7) where all relevant data are extracted from their respective universe. Afterwards a MIP of $\mathcal{O}_m$ is solved via the rolling-horizon heuristic mentioned in Section 4.3 and [6]. If a feasible production schedule ($\vec{s}_{m,j}$) is not found, step 5 to 8 is repeated. Otherwise, $\vec{s}_{m,j}$ is considered as a candidate schedule of mine m and hence is added to Ω^{i}_{m}.

Algorithm 3 Ranked TopoSort Heuristic

```
    Input: $\mathscr{B}, \hat{\phi}, \mathscr{A}^{\vee}, \mathscr{A}^{\wedge}, \mathscr{F}, \mathscr{A}^{\vee}_{\mathscr{F}}, S, G, \mathscr{Y}, \mathscr{R}, T^k_b, C_e$
    Output: $\mathscr{T}$
 1: Initialise a candidate set: $\mathscr{T} \leftarrow \emptyset$
 2: Initialise a tonnage counter for $t = 1$: $T_e \leftarrow 0$
    Pick blocks from $\mathscr{B}$ till a total tonnage larger than 1.5 times the extraction capacity in each time period is reached:
 3: while $|\mathscr{B}| > 0$ do
 4:     Initialise an eligibility set: $\mathscr{E} \leftarrow \{f | f \in \mathscr{F} \wedge |\mathscr{A}^{\wedge}(f)| = 0\}$ (i.e., face blocks with no vertical precedence)
        Pick a predefined number, denoted by $N_R$, of blocks from the eligibility set:
 5:     if $|\mathscr{E}| > N_R$ then
 6:         Compute the rank of each block in $\mathscr{E}$ using Eq. 6 and sort $\mathscr{E}$ by ranks in ascending order.
 7:         Pick the first $N_R/2$ blocks in the sorted $\mathscr{E}$ and pick another $N_R/2$ blocks from the remaining blocks in $\mathscr{E}$ randomly.
 8:         Assign all picked blocks to a set $\mathscr{P}$
 9:     else
10:         Pick all blocks in $\mathscr{E}$: $\mathscr{P} \leftarrow \mathscr{E}$
11:     end if
        Remove each picked block in $\mathscr{P}$ from its inclusive sets then update face set and precedence sets:
12:     for each picked block $p \in \mathscr{P}$ do
13:         Update tonnage counter: $T_e \leftarrow T_e + \sum_k T^k_p$
14:         Increment time period index when 1.5 times the current extraction capacity is surpassed: $t \leftarrow t+1$ if $T_e > 1.5C^t_e$.
15:         If t updates and $t >$T stop and output $\mathscr{T}$. Otherwise, reset tonnage counter for the next time period: $T_e \leftarrow \sum_k T^k_p$.
16:         Add $p$ to output set: $\mathscr{T} \leftarrow \mathscr{T} \cup \{p\}$
17:         Remove $p$ from its inclusive sets $\mathscr{F}, \mathscr{B}, \mathscr{A}^{\vee}(b)$ and $\mathscr{A}^{\wedge}(b)\ \forall b \in \mathscr{B}$, and $\mathscr{A}^{\vee}_{\mathscr{F}}(f)\ \forall f \in \mathscr{F}$
18:         Identify new face blocks after removing $p$: $\mathscr{F}_n \leftarrow \{n | n \in \mathscr{A}^{\vee}_{\mathscr{F}}(p)\}$
19:         Update face set: $\mathscr{F} \leftarrow \mathscr{F} \cup \mathscr{F}_n$
20:         for $f \in \mathscr{F}_n$ do
21:             Identify internal-adjacent blocks of face block $f$: $\mathscr{A}^{\vee}_{\mathscr{F}}(f) \leftarrow \{n | n \in \mathscr{A}^{\vee}(f) \wedge n \notin \mathscr{F}\}$
22:         end for
23:         for $f \in \mathscr{F}_n$ do
24:             Remove face block $f$ from inclusive sets: $\mathscr{A}^{\vee}_{\mathscr{F}}(i) \leftarrow \mathscr{A}^{\vee}_{\mathscr{F}}(i) \setminus \{f\}\ \ \forall i \in \mathscr{F}, i \neq f$
25:         end for
26:     end for
27: end while
```

Algorithm 4 $\mathscr{O}_m$ in DA with TopoSort

```
    Input: number of mines $M$, number of schedules for each mine $N$, production targets $\vec{\phi}^i_m$ and associated standard
           deviations $\vec{\sigma}^i_m$ of each mine $m$ in iteration $i$ of DA
    Output: Set of production schedules generated by solving $\mathscr{O}_m$ for mine $m$ in iteration $i$ of DA, denoted by $\Omega^i_m$
 1: for $m \in [1, 2, ..., M]$ do
 2:     $\Omega^i_m \leftarrow \emptyset$
 3:     for $j \in [1, 2, ...N]$ do
 4:         do
 5:             Generate a set of production bounds $[L^{m,t}_{l,q}, U^{m,t}_{l,q}]$ using $\vec{\phi}^i_m$ and $\vec{\sigma}^i_m$ via Algorithm 2 in [6] and
                retain the deviated production targets $\hat{\phi}^{m,t}_{l,q} = \phi^{m,t}_{l,q} + \Delta^{j,t}_{l,q}$ for each quality $q \in \mathscr{Q}$ in granularity
                $l \in \mathscr{L}$ at mine $m$ in each period $t \in [1, 2, ...T]$.
 6:             Find a topological ordering of blocks $\mathscr{T}$ via Algorithm 3.
 7:             Initialise all data input pertaining to $\mathscr{T}$ and mine $m$.
 8:             Solve a $\mathscr{O}_m$ of production bounds $[L^{m,t}_{l,q}, U^{m,t}_{l,q}]$ via the rolling-horizon heuristic (Algorithm 3 in [6])
                with the prune routine (Algorithm 2) implemented and output the production schedule $\vec{s}_{m,j}$.
 9:         while $\vec{s}_{m,j}$ is not feasible
10:         Add $\vec{s}_{m,j}$ to $\Omega^i_m$
11:     end for
12: end for
13: Return $\Omega^i_m$
```

Algorithm 5 Framework of Adaptive DA

```
1: Initialise the best solution and its objective values: $\vec{S}_{best} \leftarrow \emptyset$, $\vec{z}_{best} \leftarrow \emptyset$
2: Maximum standard deviations: $\vec{\sigma}^{+} \leftarrow \{\sigma^{+}_{l,q} = \Delta^{+}_{q} | l \in \mathscr{L}, q \in \mathscr{Q}\}$
3: Minimum standard deviations: $\vec{\sigma}^{-} \leftarrow \{\sigma^{-}_{l,q} = \Delta^{-}_{q} | l \in \mathscr{L}, q \in \mathscr{Q}\}$
4: Iteration indexes: $i \leftarrow 1$, $i_{stuck} \leftarrow 0$, $i_{conv} \leftarrow$ Eq. 7
5: Initialise mine-side production targets: $\vec{\phi}^{i}_{m}$ for all $m \in \mathscr{M}$
6: Initialise standard deviations: $\vec{\sigma}^{i}_{m} \leftarrow \vec{\sigma}^{+}$
7: Go_Deep ← FALSE
8: while $i \leq \text{MAX}_i$ do
9:     if $i_{stuck} \geq i_{conv} \vee$ Go_Deep is TRUE then
10:        if Go_Deep is False then
11:            Go_Deep ← TRUE
12:            if $z^{\eta}_{best} > 0$ then
13:                $\sigma^{i,m,t}_{l,q} \leftarrow (\sigma^{+}_{q} + \sigma^{-}_{q})/2 \;\; \forall m \in \mathscr{M}, t \in [1,2,...,T], l \in \mathscr{L}, q \in \mathscr{Q}$
14:            end if
15:        end if
16:        Solve each $\mathcal{O}_m$ to find $N$ production schedules ($\Omega^{i}_{m}$) for mine $m$ via the procedures of Enhanced DA
           (Section 4.4).
17:    else
18:        Solve each $\mathcal{O}_m$ to find $N$ production schedules ($\Omega^{i}_{m}$) for mine $m$ via the procedures of DA with TopoSort
           (Algorithm 4).
19:    end if
20:    Solve $\mathcal{O}_\pi$ for $\Omega^{i}_{m} \cup \{\vec{S}_{best,m}\}$ where $\vec{S}_{best,m} \in \vec{S}_{best}$ is the production schedule of mine $m \in \mathscr{M}$ in the best
       solution ($\vec{S}_{best}$) so far. The output is the best solution and its associated objective values ($\vec{z}_i$) in iteration $i$.
21:    if $z_i \geq z_{best}$ then
22:        $i_{stuck} \leftarrow i_{stuck} + 1$
23:        $\sigma^{i+1,m,t}_{l,q} \leftarrow \max\{\sigma^{-}_{q}, \gamma\sigma^{i}_{l,q}\} \;\; \forall m \in \mathscr{M}, t \in [1,2,...,T], l \in \mathscr{L}, q \in \mathscr{Q}$
24:        if $\vec{\sigma}^{i+1} \rightarrow \vec{\sigma}^{-}$ then
25:            **Terminate the algorithm and save the results**.
26:        end if
27:    else
28:        $i_{stuck} \leftarrow 0$
29:        if Go_Deep is FALSE then
30:            if $|\phi^{i,m,t}_{l,q} - v^{m,t}_{l,q}| > \sigma^{i,m,t}_{l,q}$ then
31:                $\sigma^{i+1,m,t}_{l,q} \leftarrow \min\{\sigma^{+}_{q}, \sigma^{i}_{l,q}/\gamma\} \;\; \forall m \in \mathscr{M}, t \in [1,2,...,T], l \in \mathscr{L}, q \in \mathscr{Q}$
32:            end if
33:        end if
34:        Update production targets for the next iteration: $\vec{\phi}^{i+1}_{m} \leftarrow \vec{v}_m$ where $\vec{v}_m$ is the chemistry composition in
           the produced ore at mine $m \in \mathscr{M}$ in the best solution so far.
35:    end if
36: end while
```

4.7. Adaptive DA

It is realised through multiple tests that DA integrated with TopoSort outperforms DA in [6] in solve time while yielding equally good solutions. As mentioned in Section 4.4, it has been observed that Enhanced DA that incorporates the prune routine (Algorithm 2) leads to better solutions at a cost of greater solve time. As such, an Adaptive DA is proposed to take advantage of both DA with TopoSort and Enhanced DA. Specifically, the strategy of DA with TopoSort is utilised in the first stage of Adaptive DA such that mine-side production targets $\vec{\phi}^{i}_{m}$ are refined quickly; once their standard deviations $\vec{\sigma}^{i}_{m}$ have converged closely to insignificant thresholds $\vec{\sigma}^{-}$, the second stage of Enhanced DA is activated to look for good solutions in a local area of solution space. It is shown in Section 5 that the Adaptive DA outperforms DA in [6] in both solve time and solution quality. The framework of Adaptive DA is presented in Algorithm 5.

The main architecture of DA remains the same in Adaptive DA - mine-side subproblem $\mathcal{O}_m$ and port-side subproblem $\mathcal{O}_\pi$ are solved in succession to form a solution to a MTP-MMPP problem. By solving $\mathcal{O}_m$ with respect to varying sets of production targets $\vec{\phi}^{i}_{m}$, the generated schedules ($\vec{\Omega}^{i}_{m}$) will produce ore the chemistry composition of which is clustered around $\vec{\phi}^{i}_{m}$ with a spread determined by the associated standard deviations ($\vec{\sigma}^{i}_{m}$) [6]. $\vec{\phi}^{i}_{m}$ along with the schedule in the best solution thus far ($\vec{S}_{best,m}$) are used to form a $\mathcal{O}_\pi$ with respect to a set of blending targets defined in a given MTP-MMPP. Upon obtaining the solution to MTP-MMPP ($\vec{S}_{best}$) and its associated objective values ($\vec{z}_{best}$) by solving the $\mathcal{O}_\pi$, a feedback mechanism (step 21 to 34) updates $\vec{\phi}^i_m$ and $\vec{\sigma}^i_m$ and the whole cycle repeats until all parameters in $\vec{\sigma}^i_m$ are converged to a set of predefined insignificant values ($\vec{\sigma}^-$).

Apart from the unchanged key elements in DA, a few new parameters are introduced to control the workflow of Adaptive DA. i_{stuck} denotes the number of consecutive iterations for which the algorithm has failed to find a better solution. i_{conv} is computed as per Eq. 7, specifying a threshold on i_{stuck} for which the algorithm works under the strategy of DA with TopoSort. Once i_{conv} is exceeded, the algorithm switches its strategy to that of Enhanced DA by activating a control flag Go_Deep (Step 9). Notice that the logical 'OR' ($\vee$) ensures that switching to Enhanced DA is an irreversible operation. Before Enhanced DA is executed for the first time, all standard deviations are conditionally reset to the average of two extremes - $\vec{\sigma}^+$ and $\vec{\sigma}^-$ (step 13). The satisfying condition is that the total deviation present between the chemistry composition of produced ore and mine-side production targets (denoted by z^η_{best}) is positive (step 10).

$$i_{conv} = \max\left\{3, RoundDown\left(\frac{\log(\sigma^-_{\text{metal}}/\sigma^+_{\text{metal}})}{\log\gamma}\right)\right\} \tag{7}$$

After solving a $\mathcal{O}_\pi$ in each iteration, the mine-side production ($\vec{\phi}$) and their associated standard deviations $\vec{\sigma}$ are refined and feedback to $\mathcal{O}_m$ of the next iteration. When no better solution is found, $\vec{\sigma}$ are reduced by a factor of γ with a lower bound specified in $\vec{\sigma}^-$ (step 23). Otherwise a better solution is found, $\vec{\sigma}$ are conditionally increased by a factor of $1/\gamma$ with a cap specified in $\vec{\sigma}^+$. The condition is twofold. Firstly (step 29), the algorithm must be running in the first stage of fast converging $\vec{\phi}$, i.e., under the strategy of DA with TopoSort. Secondly (step 30), the magnitude of deviation present between chemistry composition of produced ore ($v^{m,t}_{l,q}$) and the desired production target ($\phi^{i,m,t}_{l,q}$) must be greater than the associated standard deviation ($\sigma^{i,m,t}_{l,q}$). $\vec{\phi}$ are replaced by achieved chemistry composition $\vec{v}$ only when a better solution is obtained.

5. Numerical Results

In this section, the new variants of DA are evaluated with several conceptually designed test cases. The first 2 test cases involve the blending of metal grade without considering other quality attributes. For each scenario, 10 test runs were conducted using each algorithm including the old DA. Consequently, the results in terms of CPU time and solution quality are compared among those algorithms. Furthermore, the best performing algorithm - Adaptive DA is evaluated extensively with more complicated test cases in which multiple quality attributes are considered (Fe, Silica and Alumina) and a set of minimum production requirements are enforced. Lastly, the sensitivity of Adaptive DA with respect to blending targets is discussed.

A summary of each test scenario is given in Table 1 including the number of blocks, total reserve of minerals and extraction capacity. These quantities are the total of 3 mines. Each mine has its own set of operational constraints (such as stockpile capacity, transport capacity, processing plant capacity etc.) and precedence constraints (horizontal and vertical). Ores in fines and lumps are produced in each mine. Both grade blocks and blast blocks exist in each mine while the number of grade blocks accounts for $54 - 62\%$ of the entire population. There are 2 ports where 2 blends of each granularity are produced. The blending targets and port capacity are the same for each port. The overall planning horizon spans 5 weeks, which are divided into weekly time periods. In addition, the number of quality attributes and the existence of quantity-based blending targets are varied among test cases(indicated in the last two columns of Table 1).

Across all test cases, the number of blocks available for production planning in a mine ranges from 42 to 270 and the total number of blocks in the 3-mine and 2-port network ranges from 142 to 592. All experiments are run on a personal computer with an Intel Core i7 CPU at base speed 2.60 Ghz, 16 Gigabytes of RAM. The program is coded with Python programming language and Gurobi optimiser 8.1.

5.1. Metal Grade Only

First of all, the performance of two variants of DA - DA with TopoSort and Adaptive DA are evaluated and compared to DA. For simplicity, we only consider a single quality attribute as our blending target - iron grade. The box plot in Fig. 8 presents the solve times of applying each algorithm to test case 1 and 2 in Table 1. For each test case, 10 experiments are run with each algorithm. The results of both test cases show that solve time is reduced dramatically via either of the two variants of DA. For DA with TopoSort, the reduction in solve time in percent of DA's solve time is 75.97% on average with a standard deviation of 11.07%. As for Adaptive DA, the average reduction in solve time is 65.09% with a standard deviation of 14.71%.

As far as solution quality is concerned, the box plot in Fig. 9 presents the productivity values obtained by applying the aforementioned algorithms to test case 1 and 2 in Table 1. The total deviation present between iron grade of blended products and the desired blending targets is zero across all tests. Therefore, only productivity is presented and compared. For case 1, no noticeable difference in productivity exists between applying DA and DA with TopoSort, whereas significant improvement in productivity is observed by applying Adaptive DA. As for case 2, the productivity obtained by applying DA with TopoSort is clustered around 500 kt while that of applying DA has a slightly higher value tendency but with a wider range. Similar to case 1, there is an apparent improvement on productivity by applying Adaptive DA to case 2. Quantitatively, the gain in productivity by applying Adaptive DA is 12.67 % on average with a standard deviation of 7.25% across all tests of case 1 and 2 combined.

Table 1
Summary of test scenarios.

Case No.	No. of Blocks	Reserve (kt)	Extraction Capacity (kt/week)	No. of Qualities	Quantity-based Targets?
1	362	4310	270	1	No
2	592	7001	340	1	No
3	142	1550	126	3	Yes
4	142	1550	202	3	Yes
5	552	5970	307	3	Yes

Notes: Each scenario contains 3 mines and 2 ports and the planning horizon is set to 5 weeks for all tests. Quantities shown are the total across all mines. Reserve includes the total of all grades and wastes.

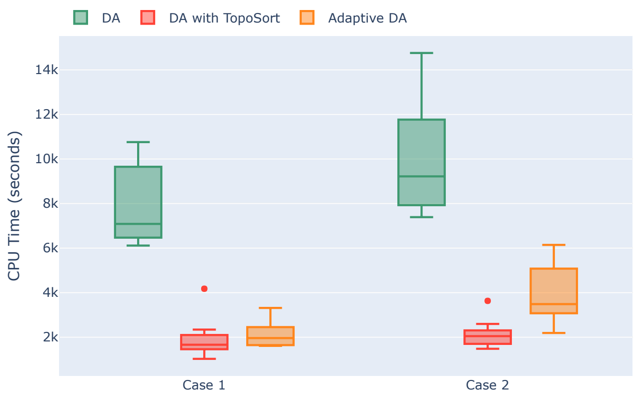

Figure 8: CPU time performance

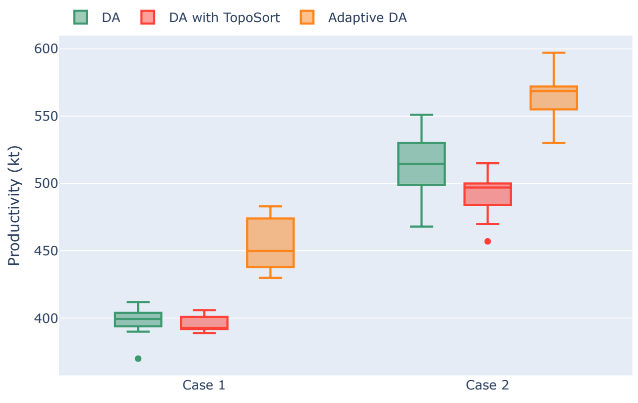

Figure 9: Productivity performance

5.2. Multiple Quality Attributes

In light of the performance gain in both solve time and solution quality by applying Adaptive DA, we now present its performance in more demanding scenarios in which the blending targets include multiple quality attributes as well as a set of quantity-based targets. Apart from desired iron grade, percent compositions of impurities including silica and alumina are specified. And a minimum amount of total production is stipulated for each blended product of each granularity.

Table 2 describes the statistics of test results by applying Adaptive DA in solving 3 varying scenarios. In all tested results, the total deviation present between quality attributes of blended products and the specified blending targets is zero. The first metric presented in Table 2 is the gap in percent to maximum achieved productivity among all test runs. And the second metric is solve time. The average across 10 test runs and its standard deviation are reported.

Table 2
Statistics of test results via Adaptive DA.

Case No.	Gap to η_{Max} (%)		CPU Time (s)	
	Avg.	Std.	Avg.	Std.
3	2.18	1.74	2567	840
4	2.03	1.16	3162	689
5	3.18	1.55	26900	9702

Notes: An average (Avg.) and standard deviation (Std.) are taken across 10 test runs for each scenario. The largest value among tested results of each scenario is taken as the maximum productivity (η_{Max}).

As shown in Table 1, case 3 differs from case 4 only in terms of the extraction capacity at each mine. Comparing to these two scenarios, the number of blocks almost quadrupled in case 5 (from 142 to 552) and the overall extraction capacity across all mines is increased to approximately 2.4 times of case 3 and 1.5 times of case 4. Despite these differences, the gap to maximum achieved productivity ranges from 2.03% to 3.18% across all tests, on average. The associated standard deviation of solving each scenario never exceeded 2.0%. These small gaps and deviations indicate that Adaptive DA is able to provide solutions with consistent quality.

As for solve time, it is observed that both increasing extraction capacity and number of blocks lead to increased solve time. The variation in solve time grows with the increased number of blocks.

6. Conclusion

In this paper, we presented a range of modifications to an existing method for integrated blending optimisation in mine planning. It has always been such a challenge to address blending optimisation for a network of multiple mines and ports over multiple time periods (MTP-MMPP). A decomposition based algorithm (DA) in [6] was claimed to be the first approach to solve MTP-MMPP. This approach in its original modelling did not consider the existence of quantity-based blending targets which is an important consideration alongside quality-based blending targets for a mining business. Accordingly, a set of minimum production requirements are included in the modelling in this paper and the secondary objective is changed from productivity maximisation to production maximisation.

To avoid the use of nonlinear constraints in modelling time-varying compositions of stockpiles, a rolling-horizon heuristic is used in DA to aggregate time periods into larger horizons so that a production schedule of multiple time periods can be formed progressively as time horizons roll towards the last time periods. It was discovered during experiments that solution quality can be improved by implementing a prune routine which removes completely mined blocks and updates precedence relations in between rolling horizons. Hence, we name the modified DA with prune routine 'Enhanced DA'. Although the prune routine was introduced with an intention to reduce solve time, the experiment results demonstrated that the gain mainly comes in solution quality. To explain the observation, the algorithm is subject to sub-optimality due to its usage of rolling-horizon heuristic (which decomposes the multiple-time-period scheduling problem into progressive solve of discrete time horizons) and the decomposition of a holistic MINLP into two MIPs, i.e., the scheduling problem $\mathcal{O}_m$ and the schedule selection problem $\mathcal{O}_\pi$. So our speculation is that changes made in variables and constraints by the pruning algorithm may not affect each individual sub-problem but it does in a way motivates the algorithm to search more extensively in the solution space. However, the exact mechanism behind such improvement still requires future investigation.

Inspired by studies that utilise a topological ordering heuristic (TopoSort) to quickly generate feasible extraction sequence [8, 7, 15], a new heuristic named Ranked TopoSort was proposed that features a ranking mechanism which sort the eligible blocks in order of their fitness to achieve the desired grade and quality targets. The Ranked TopoSort replaces the aggregation planning stage of DA and serves to select blocks for mining. The experiment results showed that solve time can be reduced by 75% on average when Ranked TopoSort is used in DA. As a drawback, lower solution quality in terms of productivity (when blending targets are achieved with zero deviation) is likely to occur.

An Adaptive DA was proposed to combine the benefits of Enhanced DA and DA with TopoSort. It was realised that a great amount of time is spent on obtaining a set of converged blending targets, after which local search for solutions of higher quality become a priority. Therefore, Adaptive DA takes a two-stage approach to address the blending optimisation. During the first stage, DA with TopoSort is used for fast convergence of blending targets and their associated standard deviations. As these values become close to convergence, the algorithm enters the second stage where Enhanced DA becomes the active approach to search for solutions of higher quality. Comparing to DA without modification, solution quality in terms of productivity can be increased by 12.67% on average when blending targets are achieved with zero deviation.

Additional experiments were conducted for Adaptive DA, where multiple quality attributes are specified in blending targets and the extension to quantity-based blending targets are considered. From the test results, it can be concluded that Adaptive DA maintains a consistent performance in providing solutions of good quality regardless of the number of blocks and the extraction capacity. For the largest instance of 552 blocks, the average solve time was less than 7.5 hours on a personal computer with i7 Core running at base speed 2.6 Ghz and 16 Gigabytes of RAM.

Glossary

Abbreviations

DA	Decomposition-based Algorithm
DAG	Directed Acyclic Graph
MINLP	Mixed Integer Nonlinear Programming
MIP	Mixed Integer Programming
MTP-MMPP	Multiple Mines and Ports Planning Problem in Multiple-time-period Setting
TopoSort	Topological Sorting Heuristic

Problem Definitions

$\mathcal{O}_\pi$	Schedule selection problem between mines and ports
$\mathcal{O}_m$	Production scheduling problem of mine m
$\mathcal{O}_{m,1}$	Sub-problem of $\mathcal{O}_m$ that minimises deviation present between produced chemistry and production targets
$\mathcal{O}_{m,2}$	Sub-problem of $\mathcal{O}_m$ that maximises productivity

Indices and Sets

Δ_q^+, Δ_q^-	Significant and insignificant change for each quality attribute, $q \in \mathcal{Q}$
$\mathcal{A}^\vee(b)$	Set of blocks that precedes block b horizontally
$\mathcal{A}^\vee_{\mathcal{F}}(f)$	Set of internal blocks adjacent to face block f
$\mathcal{A}^\wedge(b)$	Set of blocks that precedes block b vertically
$\mathcal{B}^-$	Empty blocks to be removed, $\mathcal{I}^- \cup \mathcal{F}^-$
$\mathcal{C}$	Set of contiguous blocks that form a continuous mining sequence
$\mathcal{E}$	Set of eligible blocks for mining in TopoSort
$\mathcal{F}^-$	Empty face blocks to be removed

$\mathscr{F}_n$	New face blocks identified after removal of empty blocks
$\mathscr{I}^-$	Empty internal blocks to be removed
$\mathscr{N}(\mathscr{C})$	Set of face blocks adjacent to the blocks in $\mathscr{C}$
$\mathscr{N}(b)$	Set of blocks adjacent to block b
$\mathscr{P}$	Set of blocks selected for mining in TopoSort
$\mathscr{T}$	Set of topological ordered blocks considered for mining after applying TopoSort
ϕ	Production targets on grade and quality attributes
σ	Standard deviations for product targets
A_k, $\mathscr{A}(A_k)$	Aggregate of blocks, indexed by k, and the set of blocks inside
b, $\mathscr{B}$	Blocks, particularly in Algorithm 1 refer to those are of same category (i.e. grade, blast and waste)
b, $\mathscr{B}^c$	Blocks prohibited for aggregation in Algorithm 1
b^-, $\mathscr{B}^-$	Blocks to be removed in rolling-horizon heuristic
d, $\mathscr{D}_m$	Destinations (stockpiles, plants, waste dumps) in mine m
f, $\mathscr{F}$	Blocks on the mining face
f, $\mathscr{F}_A$	Aggregates that contain face blocks
H_1, H_2	Time horizons
l, $\mathscr{L}$	Granularities (fines and lump)
m, $\mathscr{M}$	Mines
q, $\mathscr{Q}$	Grade and quality attributes
s, $\mathscr{S}_m$	Sources (blocks and stockpiles) in mine m

Parameters, Variables and Expressions

ϵ, ϵ^-	Relative tolerance and multiplicative tolerance
$\hat{\phi}^{m,t}$	Production targets of mine m in time t
$\hat{\phi}^{m,t}_{l,q}$	Target of quality q in granularity l at mine m in time t
$\mathscr{R}_{l,q}$	Percentage of quality q in the granularity l that will be recovered after upgrading (wet processing)
$\mathscr{Y}_l$	Percentage of granularity l that will be recovered after upgrading (wet processing)
Ω^i_m	Mining schedules generated for mine m in iteration i
Π	Number of ports
$\vec{\phi}^i_m$	Production targets of mine m in iteration i
$\vec{\sigma}^i_m$	Standard deviations on production targets of mine m in iteration i
$\vec{s}_{m,j}$	Mining schedule j of mine m
C^m_d	Processing capacity of plant d in mine m
C_e, C^t_e	Extraction capacity (in time t)
$G^{hi}_{b,l,q}$	Percentage of quality q in the granularity l within the high grade ore of block b
$G^{lo}_{b,l,q}$	Percentage of quality q in the granularity l within the low grade ore of block b
M	Number of mines
M_A	Maximum number of blocks in an aggregate
N	Number of schedules for each mine-side scheduling problem, $\mathscr{O}_m$
N_R	Maximum number of blocks for selection in TopoSort
$Q^*_{n,l}$	Minimum production requirement for product n in granularity l
$q^t_{n,l}$	Quantity of product n in granularity l in time t
$r^{m,l,t',j}_{\pi,n,t}$	Number of trainloads in granularity l departed from mine m in time t' and blended into product n at port π in time t, as specified in schedule j
$S^{hi}_{b,l}$	Percentage of granularity l in the high grade ore of block b
$S^{lo}_{b,l}$	Percentage of granularity l in the low grade ore of block b
T^{hi}_b, T^{lo}_b	Tonnage of high/low grade ore in block b
T_e	Accumulated tonnage of selected blocks in TopoSort
T_R	Tonnage of a nominal trainload
$v^{m,t',j}_{g,l}$	Proportion of metal grade g in ore of granularity l produced in mine m in time t', as specified in schedule j
$x^{m,t}_{s,d}$	Amount of material sent from source s to destination d in mine m during time t
$z_{\pi,2}$	Secondary objective in mine-to-port sub-problem, $\mathscr{O}_\pi$

Acknowledgements

This work was supported by the Rio Tinto Centre for Mine Automation and the Australian Centre for Field Robotics, University of Sydney, Australia.

References

[1] Amos, F., Rönnqvist, M., Gill, G., 1997. Modelling the pooling problem at the new zealand refining company. Journal of the Operational Research Society 48, 767–778.

[2] Audet, C., Brimberg, J., Hansen, P., Digabel, S.L., Mladenović, N., 2004. Pooling problem: Alternate formulations and solution methods. Management science 50, 761–776.

[3] Blom, M., Pearce, A.R., Stuckey, P.J., 2017. Short-term scheduling of an open-pit mine with multiple objectives. Engineering Optimization 49, 777–795.

[4] Blom, M., Pearce, A.R., Stuckey, P.J., 2018. Multi-objective short-term production scheduling for open-pit mines: a hierarchical decomposition-based algorithm. Engineering Optimization 50, 2143–2160.

[5] Blom, M.L., Burt, C.N., Pearce, A.R., Stuckey, P.J., 2014. A decomposition-based heuristic for collaborative scheduling in a network of open-pit mines. INFORMS Journal on Computing 26, 658–676.

[6] Blom, M.L., Pearce, A.R., Stuckey, P.J., 2016. A decomposition-based algorithm for the scheduling of open-pit networks over multiple time periods. Management Science 62, 3059–3084.

[7] Chicoisne, R., Espinoza, D., Goycoolea, M., Moreno, E., Rubio, E., 2012. A new algorithm for the open-pit mine production scheduling problem. Operations Research 60, 517–528.

[8] Gershon, M., 1987. Heuristic approaches for mine planning and production scheduling. International Journal of Mining and Geological Engineering 5, 1–13.

[9] Haverly, C.A., 1978. Studies of the behavior of recursion for the pooling problem. Acm sigmap bulletin , 19–28.

[10] Misener, R., Floudas, C.A., 2009. Advances for the pooling problem: Modeling, global optimization, and computational studies. Applied and Computational Mathematics 8, 3–22.

[11] Mousavi, A., Kozan, E., Liu, S.Q., 2015. Optimisation of open pit mine block sequencing. Ph.D. thesis. Queensland University of Technology.

[12] Mousavi, A., Kozan, E., Liu, S.Q., 2016. Open-pit block sequencing optimization: a mathematical model and solution technique. Engineering Optimization 48, 1932–1950.

[13] Samavati, M., Essam, D., Nehring, M., Sarker, R., 2017a. A local branching heuristic for the open pit mine production scheduling problem. European Journal of Operational Research 257, 261–271.

[14] Samavati, M., Essam, D., Nehring, M., Sarker, R., 2017b. A methodology for the large-scale multi-period precedence-constrained knapsack problem: an application in the mining industry. International Journal of Production Economics 193, 12–20.

[15] Samavati, M., Essam, D., Nehring, M., Sarker, R., 2018a. A new methodology for the open-pit mine production scheduling problem. Omega 81, 169–182.

[16] Samavati, M., Essam, D., Nehring, M., Sarker, R., 2019. Production planning and scheduling in mining scenarios under ipcc mining systems. Computers & Operations Research .

[17] Samavati, M., Essam, D.L., Nehring, M., Sarker, R., 2018b. Open-pit mine production planning and scheduling: A research agenda, in: Data and Decision Sciences in Action. Springer, pp. 221–226.

[18] Weintraub, A., Pereira, M., Schultz, X., 2008. A priori and a posteriori aggregation procedures to reduce model size in mip mine planning models. Electronic Notes in Discrete Mathematics 30, 297–302.