Planning for Mining Operations with Time and Resource Constraints

Nir Lipovetzky and **Christina N. Burt** and **Adrian R. Pearce** and **Peter J. Stuckey**
Computing and Information Systems
The University of Melbourne
Parkville, Australia

Abstract

We study a daily mine planning problem where, given a set of blocks we wish to mine, our task is to generate a mining sequence for the excavators such that blending resource constraints are met at various stages of the sequence. Such time-oriented resource constraints are not traditionally handled well by automated planners. On the other hand, the remaining problem involves finding node-disjoint sequences with state-dependent travel times on the arcs, which are highly challenging for a Mixed-Integer Program (MIP). In this paper, we address the problem of finding feasible sequences using a combined MIP and planning based decomposition approach. The MIP takes care of the resource constraints, and the planner solves the remaining sequence problem. We extend the notion of finding feasible sequences to finding *good* feasible sequences, by devising a heuristic objective function in the MIP, which improves the resulting search space for the planner. We empirically analyse the scalability of our approach on a benchmark data set, before demonstrating its effectiveness on a real world case study provided by our industry partner. These results demonstrate that by using a heuristic MIP, it is possible to obtain better makespan results with a suboptimal planner than by using an optimal planner with an uninformed MIP.

Introduction

Daily open-pit mine planning is the problem of generating feasible sequences of *blocks* for excavating-equipment to mine, such that *blending* requirements are met at the product stockpiles. A feasible sequence is a series of actions for each excavator, including movements between blocks, and mining the blocks themselves. The task of generating a feasible sequence involves allocating excavating equipment to blocks. The scope of this paper is limited to excavators, rather than also considering truck/hauling equipment.

In a surface mine, the blocks are pre-determined sections of ore that are marked for mining in this schedule. Naturally, there is a *precedence* ordering on the blocks for the case where some blocks must be mined to reach the blocks behind. A number of excavators have the task of mining this set of blocks. The excavators move from block to block, forming a *mine sequence*. The order of the blocks in these sequences will affect the makespan and the ability to achieve the required blend of the formed stockpiles. The excavators may not mine more than one block simultaneously, leading to a state-wise node-disjoint path sub-structure to the problem. Furthermore, the time required for an excavator to travel between two blocks is dependent on what has already been mined. That is, as blocks are removed, new pathways may be formed. Therefore, the travel time is dependent on the *state* of the mine. In this paper, we address the problem of finding parallel plans that yield good, but not necessarily optimal, makespans such that these criteria are met.

The computational difficulty for this problem arises not in the size of the instances—these are actually quite small—but in the layering of several difficult sub-structures. The state-wise node-disjoint path sub-structure, which imposes that excavator movements are contiguous and do not intersect at blocks at any given moment, is analogous to the problem of finding optimal multi-commodity network flow which is already $\mathcal{NP}$-complete for the case of Euclidean traversal times between blocks (Even, Itai, and Shamir 1975). The precedence constraints alone can undesirably convert a polynomial-time solvable problem to $\mathcal{NP}$-complete (Lenstra and Rinnooykan 1978). On top of this, we have the resource constraints, which are known to convert polynomial-time solvable problems to $\mathcal{NP}$-hard complexity (Blazewicz, Lenstra, and Rinnooykan 1983).

This challenging problem is of great practical importance to mining operations. In this setting, long-term plans are used to guide the derivation of short-term plans which, in turn, are handed down to operational planners who must implement the final, weekly, fine-resolution plan. It is at this stage that undetected infeasibilities in the plans frequently become apparent. Infeasibilities arise due to inaccurate allocation of attributes of the ore in any given block, unavailability of some equipment due to maintenance and breakdowns, or, quite simply, the impossibility of achieving the plan due to the time required to move equipment. The latter can only be coarsely estimated in higher level planning, where the plan fidelity is not sufficiently detailed to account for this fine-grain information (Sandeman et al. 2010).

The current approach in industry is to solve the problem using manual block picks. That is, a highly trained human planner utilises their experience of plan actualisation to devise excavator sequences in an ad hoc manner on a day-to-day basis. Output from planning tools, such as mine scheduling software, are used to guide these plans. However, they lack a tool that explicitly gives a mining sequence for each excavator, and their experience is necessary to create better plans on-the-fly. Such a tool would give mine planners the necessary information to improve equipment utilisation and efficiency of the operation, as well as provide the flexibility to adjust the schedule on-the-fly.

Solving problems with state-dependent traversal times is achievable in planning [see, e.g., (Benton, Coles, and Coles 2012)]. However, mixed-integer programming has not been used, to the best of our knowledge, to solve an operational planning problem that accounts for equipment movement. This is most likely because even when the traversal times are not state-dependent, i.e., we assume the shortest Euclidean travel time between any two blocks can be achieved, the problem is still computationally challenging. This, in combination with the quantity of variables required to capture the discretisation of a week into minutes (and the symmetries that arise) renders the full problem intractable for present day MIP solvers. However, the very nature of solution construction in planning, i.e., searching through states, makes it particularly amenable to this aspect of the problem. Conversely, one of the main challenges for planning is to reason over highly constrained resources, even when all actions have the same unitary duration (Nakhost et al. 2012). For modern planning systems, encoding this problem just as a planning problem is not tractable.

In this paper, we present a MIP–planning tool for solving this problem for our industry partner. We achieve this by matching the complementary strengths of both MIP and planning in a decomposition approach. We first approximate the projection of the problem onto the resource related variables, and formulate this problem as a MIP. We derive a guiding objective function which attempts to anticipate a 'good' solution space for the planner. The output from the MIP is an allocation of blocks to stockpiles such that the resource constraints are met. This is the input to the planner, which then efficiently finds the state-wise node-disjoint excavation sequences. The key insights are that the problem itself has structure that we can exploit in both MIP and planning paradigms, and that the MIP output can be heuristically manipulated to the advantage of the planner.

In the remainder of the paper, we will provide a full description of the application, as well as a brief background to Mixed-Integer Programming and Related Work. We then present further details of our methodology, including the MIP and planning models. We demonstrate the scalability of our method on benchmark problems, before illustrating its performance on a case study, with discussions and conclusions following.

Background

Problem Description

The Rio Tinto operated Yandicoogina mine is a 54 million tonnes/annum iron ore mine in Western Australia (22.77°S 119.23°E). The main product from Yandi is iron ore fines, which is a small granularity product. While the iron content is clearly important, there are contaminants in the ore that affect the efficacy of smelting and metal quality once the ore is processed. It is therefore important to our industry partner to create a blended product that meets the contaminant requirements of its customers. Ore that is low in silicon dioxide (SiO_2) and aluminium oxide (Al_2O_3) is considered "high grade" and can be blended with lower grade ore to obtain a final product within expected contaminant grade bounds.

In our case study, we analyse data from August/September 2013, where there were two primary active pits. We consider planning periods of one week. In order to demonstrate the computational effectiveness of using a combined MIP and planning approach, we select the most computationally complex, and therefore interesting, of the available week data for our case study.

Within this time-frame, only pre-blasted ore may be mined. Therefore, we do not consider vertical precedences. Multiple levels can be trivially projected to a plane. We are provided with an *a priori* short-term plan, which dictates which blocks should be mined in that period and the goal blends to be achieved in the stockpiles. Blocks vary from 5kT to 155kT in size, and are not uniformly shaped. We infer precedences, travel and processing times from the mine status, as provided by our industry partner.

The problem that we address in this paper can be formally expressed as follows.

Definition 1 (Operational mine planning) *Given a short-term plan, goal blends and a set of blocks to be mined, determine feasible extraction sequences for the available excavators to perform.*

In addition to the problem definition, there are several assumptions provided by our industry partner, as follows.

Assumption 1 *Blocks may be split across several stockpiles.*

While the potential outcome of this assumption is not ideal—as it could clearly lead to inefficient use of equipment—it may be necessary to, for example, split a high grade block across two stockpiles in order to meet the blending requirements. Such splitting actually occurs regularly in the minesite, but is undesirable from a planning perspective.

Assumption 2 *Only one excavator may mine a block at any one time.*

The accessibility of the blocks is also important—we require sufficient space for the excavator to swing 180° and for trucks to be able to complete a full turn. Let $\Gamma(b)$ contain the sets of precedences for a given block b. Then we define $\mathcal{B}_k(b) \in \Gamma(b)$ to be any of these sets, which are contiguous blocks that must be removed together in order to gain access to block b. For some blocks, such as those which are already accessible, $\Gamma(b) = \emptyset$. There may be $|\Gamma(b)|$ such sets—any of which may be removed to gain access, creating disjunctive precedences. We provide an illustration of such a set in Figure 1.

Assumption 3 *A block is accessible if the set $\mathcal{B}_k(b)$ of adjacent blocks is cleared for some k.*

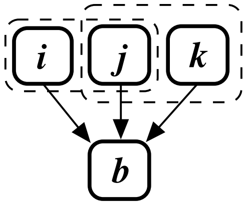

Figure 1: An illustration of the disjunctive precedence sets. Suppose we want to mine block b, and that this is possible if two adjacent blocks are already mined. Let blocks i, j, k be the blocks in the precedence set, while b is inaccessible from any other route. Then, the precedence set $\Gamma(b) = \{\{i, j\}, \{j, k\}\}$, where $\mathcal{B}_1(b) = \{i, j\}$ and $\mathcal{B}_2(b) = \{j, k\}$.

A further restriction relates to controlling the grade of each stockpile. This control is easier to achieve if the stockpiles are not created simultaneously.

Assumption 4 *Stockpiles are created sequentially and a new stockpile may not be started until the previous stockpile is complete.*

The output to the problem is a set of feasible sequences for the excavators, which dictate their movement and mining actions.

Mixed-Integer Programming

Mixed-Integer Programming (MIP) is both a modelling and solving methodology for problems that can be described completely with linear constraints and objective function, and with both continuous and discrete variables. It relies on the assumption that, when the discrete nature of some variables is relaxed, the remaining problem is convex and the optimal solution to the relaxed problem occurs at an intersection point of its convex hull. Vossen et al. showed that planning models can be equivalently expressed as MIP models. However, this requires an index of each variable to represent every state in the planning sequence—the resulting MIP is therefore unlikely to be tractable using traditional MIP solving methods in the context of interesting planning problems.

Related Work

In the context of scheduling in mining, MIP technology has been extensively studied. However, the operational level planning, has not been well addressed. Smith discusses operational level planning, but does not consider the extension to include equipment movement. Other notable works in the MIP literature include Martinez and Newman, who consider fine-grain operational productivities in the presence of coarse-grain short term plans, modelled as a MIP; and, Demeulemeester and Herroelen, who devise a branch and bound procedure for resource-constrained project scheduling. The first attempts to exploit the strength of MIP algorithms to aid search for planners involved transformations of a planning problem to a MIP problem—i.e., creating linear programming or mixed-integer programming based planners (see, e.g., (Bylander 1997; Kautz and Walser 1999; Vossen et al. 2000).)

The natural motivation to incorporate the strengths of MIP and planning is to use MIP as a feasibility check for planning search—the intention here is to reduce the search space, as in (Van Den Briel et al. 2007).

Another way to reduce the planning search space is by decomposing the problem, as we have in this paper. When the required structure is present in a problem, this is the obvious choice and we do not claim that we are the first to implement it. Fernández and Borrajo use a linear program to solve part of a clustered-knapsack problem, and leave the remaining causal constraints for the planner to solve. In another example, Flórez et al. use a MIP to make a partial assignment in an intermodal transportation application. This has the effect of reducing the search space for the planner, such that the problem becomes easier to solve in a reasonable time-frame. In our work, we strengthen this notion to not only reduce the state space for the planner, but to choose a part of the state space that produces consistently good solutions.

Method

In order to decompose the problem into the respective strengths of the two solving technologies, we use a projection method as in Benders Decomposition (e.g., see (Hooker 2005).) To begin, we approximate the projection of the problem onto the variables relating to resource allocation, i.e., we eliminate the cumbersome time-related variables. This problem can be solved effectively by MIP technology, yielding an allocation of blocks to stockpiles, such that the blending constraints are satisfied. Since there may be many solutions meeting this criteria, we attempt to guide the MIP toward ‘better’ solutions using a specially formulated objective function and constraints. We then pass this block allocation to the planner, which searches for good excavation sequences for the available equipment. We note that even if the MIP and planning decomposed problems are solved to optimality, the final solution cannot be guaranteed to be optimal. This is because the MIP model is a relaxation of the overall problem, and can yield solutions that are not feasible with respect to a time horizon. We therefore lift the time horizon restriction—that the plans must be enacted within a week—as our solutions are approximate by design.

An Approximation of the Projection (as a MIP)

Although we project away the ‘movement’ or time-related variables, the stockpiles themselves are created sequentially: this ordering gives rise to an implicit time ordering in the projection. That is, in the projection we can consider the stockpiles to partition time into β periods, where β is the number of stockpiles inferred from the summation of block tonnes and the minimum allowable size of stockpiles. Every block b is contained in the set of blocks $\mathcal{B}$. We adopt the following variables in this model:

$x_{b,d}$ [continuous] gives the proportion of block $b \in \mathcal{B}$ mined in period $d \in \beta$,

$y_{b,d}$ [binary] indicates if block $b \in \mathcal{B}$ has been completely mined in period $d \in \beta$,

$\gamma_{b,d,k}$ [binary] indicates if an adjacent subset of blocks $\mathcal{B}_k(b) \subseteq \Gamma(b), k \in \{1, \ldots, |K(b)|\}$, to block b are cleared by period d.

Other important notation includes:

c is an index referring to a particular contaminant,

C_b is the capacity (tonnes) of block b,

$G_{b,c}$ is the grade of contaminant c in block b,

G_c^i is the lower (upper) bound for contaminant c for all periods, when $i = \mathcal{L}$ ($i = \mathcal{U}$),

D^i is the lower (upper) bound for the size of the stockpiles, when $i = \mathcal{L}$ ($i = \mathcal{U}$),

$K(b)$ is the set of indexes for the disjunctive precedences, such that $k \in \{1, \ldots, |K(b)|\}$.

The following model, *feasibility* MIP F, describes a mixed-integer programming formulation where the objective function, (1), is zero for all variables. Therefore, as it is stated here it is a feasibility problem. Later, we will alter the coefficients to be non-zero for some variables, thereby obtaining a way to drive the solutions into a better solution space.

$$F: \quad \min \ \mathbf{0}^\top (x_{b,d}, y_{b,d}, \gamma_{b,d,k}) \tag{1}$$

$$\text{s.t.} \sum_d x_{b,d} = 1 \qquad \forall\, b, \tag{2}$$

$$\sum_d y_{b,d} = 1 \qquad \forall\, b, \tag{3}$$

$$\sum_b C_b x_{b,d} \geq D^{\mathcal{L}} \qquad \forall\, d, \tag{4}$$

$$\sum_b C_b x_{b,d} \leq D^{\mathcal{U}} \qquad \forall\, d, \tag{5}$$

$$\sum_b G_{b,c} C_b x_{b,d} \geq G_c^{\mathcal{L}} \sum_b C_b x_{b,d} \qquad \forall\, c, d, \tag{6}$$

$$\sum_b G_{b,c} C_b x_{b,d} \leq G_c^{\mathcal{U}} \sum_b C_b x_{b,d} \qquad \forall\, c, d, \tag{7}$$

$$\gamma_{b,d,k} \leq \frac{\sum_{d' \leq d, i \in \mathcal{B}_k(b)} y_{i,d'}}{|\mathcal{B}_k(b)|} \qquad \forall\, b, d, k, \tag{8}$$

$$\gamma_{b,d,k} \geq \sum_{d' \leq d, i \in \mathcal{B}_k(b)} y_{i,d'} - (|\mathcal{B}_k(b)| - 1) \qquad \forall\, b, d, k, \tag{9}$$

$$\sum_{d' \leq d} x_{b,d'} \leq \sum_k \gamma_{b,d,k} \qquad \forall\, b, d, \tag{10}$$

$$y_{b,d} \leq \sum_{d' \leq d} x_{b,d'} \qquad \forall\, b, d, \tag{11}$$

$$x_{b,d} \in [0, 1], \gamma_{b,d,k}, y_{b,d} \in \{0, 1\}. \tag{12}$$

Constraint (2) ensures all blocks in the given set are mined. Constraint (3) ensures that block completion occurs only once. The correct size of the stockpiles is ensured by demand constraints (4)–(5), while the blend constraints are enforced by constraints (6)–(7). The disjunctive precedence constraints (8)–(10) prevent $x_{b,d}$ from being mined unless any precedences in a set $\mathcal{B}_k(b) \in \Gamma(b)$ for k have been mined. Constraint (11) links variables $x_{b,d}$ and $y_{b,d}$ together.

The precedence constraints (8)–(10) in this form are not sufficient to prevent cliques of blocks that each satisfy one anothers precedences, yet together are not reachable. In the case where precedence cliques occur, we introduce an additional constraint, which considers the precedences for every set $\mathcal{A}_{b,k} = b \cup \mathcal{B}_k(b)$, which is the union of b with a subset of its precedences. $\Gamma(\mathcal{A}_{b,k})$ now defines the precedences for that union set, which necessarily excludes every block from $\mathcal{A}_{b,k}$. Then the required constraint to ensure these sets of blocks are themselves accessible is:

$$\frac{\sum_{i \in \mathcal{A}_{b,k}} x_{i,d}}{|\mathcal{A}_{b,k}|} \leq \sum_{b' \in \Gamma(\mathcal{A}_{b,k}), d' \leq d} y_{b',d'} \qquad \forall\, \mathcal{A}_{b,k}, d.$$

While this type of constraint should be implemented in a separation algorithm (see, for example, Geoffrion and Marsten), the number of such precedence cliques arising in a one week schedule is very small. In these cases, the number of constraints can easily be determined *a priori* and can be added to the model from the outset.

The key output from the MIP F are the values of variable $x_{b,d}$, which provide the quantity of block b which should be mined for stockpile d. This model leaves us with a computationally efficient way to find a feasible block to stockpile allocation. However, we would like to generate allocations that are desirable for the planner. Consider Figure 2.

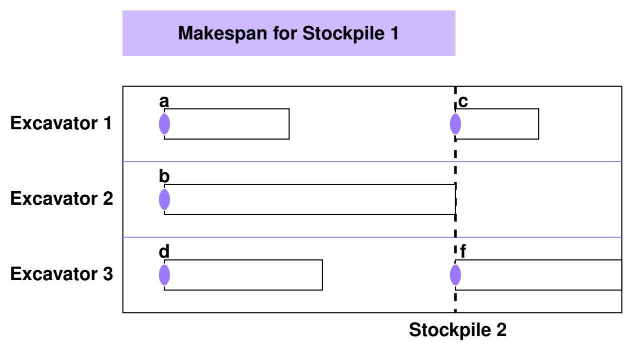

Figure 2: An illustration of a planning solution based on a feasible MIP F output. The excavator-to-block allocations are given, where the rectangles depict the time required to mine each block. Once an excavator has finished mining a block, it will move to the next block for that stockpile. If no such block exists, it will wait. Typically, the stockpile allocations elicit unbalanced block sizes, leading to long waiting times for the excavators.

Here, we illustrate a typical planning solution obtained from the allocation given by the feasible MIP F. Since the stockpiles must be generated in sequence, the imbalance in workload between the excavators leads to waiting times that have negative impact on the makespan. Ideally, the MIP solution would produce a more balanced allocation, which is illustrated in Figure 3. This is discussed in the remainder of this section.

Desirable properties of the stockpile allocation

We would like to find the best partitioning of the blocks from the MIP such that the planner can find efficient sequences for the excavators in order to reduce the makespan. That is, we wish to manipulate the MIP output such that the

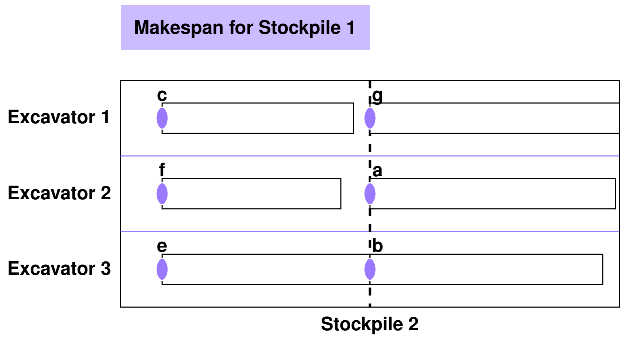

Figure 3: An illustration of a planning solution based on the heuristic MIP H output. The stockpile allocations are now balanced with respect to workload, leading to shortened waiting times for excavators.

allocation is as close as possible to the optimal allocation obtained for the minimal makespan solution. Recall that the MIP F does not have any variables associated with time, excavators or mine topology, and therefore cannot explicitly encode the allocations of blocks to excavators. However, we can motivate the allocations using an objective function. The desirable *properties* of the solution from F are:

1. *Each excavator makes a contribution to each stockpile*. Waiting leads to under-utilisation of equipment, and may contribute to longer makespans.
2. *The workload for each stockpile is balanced among excavators*. Clearly, the makespan will be minimised if the workload is shared.
3. *Partitions preferably occur in neighbouring stockpiles*. This property allows excavators to stay within the partitioned block and begin mining it immediately in the period.

One way to achieve properties (1) and (2) is to minimise the deviation between excavator workload. Since the MIP has no variables relating to excavators, we estimate possible excavator allocations by introducing continuous variables, $\delta^i_{l,l',d} \in \mathcal{R}^+$, to infer the minimum positive (negative) deviation between the material moved in any period when $i = +$ ($i = -$) for any excavator pair (l, l'). These deviations are sound if we know the excavator allocations. So, we first need to make a best-guess as to which block sequences the planner will give to the excavators. To do this, we partition the blocks into preferential sets, $\mathcal{P}(l)$, for each excavator using a fair division scheme as follows: each excavator has an equal opportunity to nominate its preferred block (which we choose based on the shortest path from each excavator to each block.) We then introduce new constraints:

$$\delta^+_{l,l',d} \geq \sum_{b \in \mathcal{P}(l)} C_b x_{b,d} - \sum_{b' \in \mathcal{P}(l')} C_{b'} x_{b',d} \quad \forall\, l, l', d, \quad (13)$$

$$\delta^-_{l,l',d} \geq \sum_{b' \in \mathcal{P}(l')} C_{b'} x_{b',d} - \sum_{b \in \mathcal{P}(l)} C_b x_{b,d} \quad \forall\, l, l', d. \quad (14)$$

In order to create balance, it is not sufficient to minimise the sum of the deviations. This is because a summation amortises across all deviations and can still lead to some large differences between workloads. We must, instead, minimise the bottleneck deviation. Since we wish to obtain 'uniform' workload in each stockpile, we only need to minimise the maximum deviation within each stockpile. To do this, we introduce further continuous variables, $\rho_d \in \mathcal{R}^+$, to represent this bottleneck, and link it to the deviations in the following way:

$$\rho_d \geq \delta^i_{l,l',d} \qquad \forall\, l, l', d, i \in \{+, -\}. \quad (15)$$

Thus, we obtain the following objective function to minimise the bottleneck in each stockpile:

$$\min \quad \sum_d \rho_d.$$

These extensions to the feasible MIP F take care of properties (1) and (2). We shall differentiate this MIP from the feasibility version by referring to it as the *heuristic MIP H*.

For property (3), we introduce a constraint that ensures that at most two adjacent variables may be non-zero:

$$SOS2(x_{b,d} \;\; \forall\, d), \forall\, b. \quad (16)$$

This is a complex constraint expressed as a Special-Ordered Set (type 2) branching rule (Ryan and Foster 1981). It elicits a structure suitable for branching on the constraint itself in the branch-and-bound schema for solving MIP. Including this constraint ensures that, if blocks are partitioned across multiple stockpiles, this is restricted to at most two stockpiles and the stockpiles must be adjacent.

To demonstrate the computational effect of this constraint, we perform separate experiments on problem H with the $SOS2$ encoded, which we indicate by MIP H_s.

Planning Model

The planning problem is formulated in PDDL2.1 (Fox and Long 2003) such that its solution is a concurrent temporal plan realising a partition from blocks to stockpiles, computed previously by the MIP solver. The purpose of the partition is to reduce the planning search space, simplify the resulting model, and thus improve the scalability of the planning solvers.

If no partition is given, the planning model has to account for the blending constraints over each stockpile. Note that these constraints do not apply for the complete state trajectory, but rather intermediate states as each stockpile is completed. Current temporal planning technology is not equipped with adequate tools to reason over these types of constraints. Conversely, modern planners act greedily to achieve the blend constraints for the first stockpiles, without a mechanism for detecting poor choices that lead to infeasibility. If a partition of blocks to stockpiles is taken as a sorted sequence of goal sets, it is sufficient to reduce the search space to only finding concurrent paths that respect this order. Indeed, all the paths that do not violate the order provided by the partition achieve the blending constraints, without the need to model blending in the planner.

The partition not only puts the planner into a feasible space, but also avoids the need for time windows, which, if too coarse, may render some blending requirements infeasible; and, if too fine, may incur a branching factor explosion. Given the practical assumption 2, and the MIP solution, a single *dig* action per excavator for each partitioned block per stockpile is sufficient. Thus, the total number of possible *digging* actions is

$$\Big|L \times \sum_{b,d} \mathbb{1}_{x_{b,d}>0}\Big|,$$

where L is the number of excavators and the indicator function is 1 if some ore is removed from block $b \in \mathcal{B}$ in period $d \in \beta$. A fixed time window is assigned to each action depending on the extraction rate of an excavator and the tonnes to be mined per block.

Moreover, given Assumption 4, we can further exploit the MIP solution to decompose the planning problem itself. If the blocks are partitioned into β stockpiles, it creates β planning problems, the solution of which provides the movements and assignments of the excavators that achieve the mining of each block just for the current stockpile $d \in 1..\beta$. Once the first stockpile, $d = 1$, is solved, the initial positions of the excavators for the next stockpile, $d+1$, are defined as the final positions in previous subproblem d. The solution for each stockpile, d, concatenated together form a complete solution for mining the sequence of stockpiles. The precedences ensure that the MIP makes good decisions regarding ordering the stockpiles, and the planner takes this as input. Note that no digging action for stockpile $d+1$ can start before the last digging action for stockpile d ends, but movements are still allowed to position the excavators in the best position for the blocks in the next subproblem.

We define the topology of the mine as an undirected graph $G_T = \langle V_T, E_T \rangle$. Vertexes, $v \in V_T$, are the initial location of excavators and scheduled blocks, and edges $(v, v') \in E_T$ are roads connecting the blocks. A cost function $dist(v, v')$, for all (v, v') defines the distance to traverse an edge. Note that the cost function associated with indirect paths is state-dependent, as the traversal time may update as blocks are mined and new pathways are created.

We model the temporal planning subproblems without the need for any numerical variable as follows.

Definition 2 (planning problem Π_d) *Given the tonnes $x_{b,d} \times C_b$ mined from each block b for stockpile d, the topology graph G_T and road distances $dist(v, v')$, the planning problem Π_d for stockpile d is characterised by a tuple $\langle F, I, O, G \rangle$, where*

- $F = \{at(b, v), at(l, v), mined(b)\}$ *are the set of Boolean variables (fluents), $at(*, v)$ describing all possible locations of excavators $l \in L$, fixed location of blocks $b \in \mathcal{B}$, and $mined(b)$ indicating if a scheduled block is mined for a particular stockpile;*
- $I = \{at(b, v), at(l, v)\}$ *describes the initial locations of blocks and excavators;*
- $O = \{dig(l, b), move(l, v, v')\}$ *are the set of operators— $dig(l, b)$ defines the digging action for each excavator and scheduled block, and $move(l, v, v')$ defines the moving action for each excavator along each edge;*
- $G = \{mined(b)\}$ *is the goal situation, defined for all blocks b in stockpile d, whose value $x_{b,d} > 0$ in the MIP solution.*

A solution for Π_d is a plan, π_d, containing at most $|G|$ dig actions, as just one excavator at a time can mine a block, and a set of $move$ operators. Given the excavators dig rate, R_l, in tonnes/minute and velocity V_l in metres/minute, the duration of a dig action is defined as $dur(dig(l, b)) = C_b \times x_{b,d}$ / R_l, i.e., tonnes allocated from block b to stockpile d divided by the digging rate of excavator l; and, the duration of a $move$ action as $dur(move(l, v, v')) = dist(v, v')$ / V_l, i.e., the distance connecting vertexes $(v, v') \in E_T$ divided the velocity of excavator l. Note that digging actions last substantially longer than movement actions, as the average block takes 1000 minutes to dig, while average moving actions take only 80 minutes.

In order to apply a $dig(l, b)$ action, excavator l has to be at a location v where $at(b, v) = true$. Then, $mined(b)$ becomes true at the completion of the action, while $at(b, v)$ becomes false if $y_{b,d} = 1$, i.e., block b is finished at current subproblem d. A special fluent, $mining(l)$, is true throughout the duration of $dig(l, b)$ actions, and is necessarily false at the beginning of any moving action. The operator $move(l, v, v')$ also requires excavator l to initially be at v, location v' to be free by having $at(l, v') = false$ for all $l \in L$, and sets location v' true at the end. Furthermore, as an excavator has to be located in the same location of an available block in order to mine it, but cannot move through it, $move(l, v, v')$ requires that $at(b, v)$ is false. As a result, if an excavator chooses to move into a location containing a block to mine, it is forced to mine it, avoiding unnecessary branching in the search. A special move operator is introduced for moving excavators from partially mined blocks, to prevent them from getting stuck.

The plan, π_d, for each stockpile $d = 0, \ldots, \beta$ is concatenated ensuring that all $dig(l, d{+}1)$ actions for stockpile $d{+}1$ start after the last $dig(l, d)$ action for stockpile d has finished.

The quality of the global solution is given by the ability of planners to minimise the makespan of the solutions π_d. The only optimal temporal planner available is CPT (Vidal and Geffner 2006), while more alternatives are available as *satisficing* temporal planners. Surprisingly, not even the satisficing planners scale-up if all the stockpiles are solved at once, only finding quick solutions for the subproblems Π_d induced by the MIP partition. We compare the impact of the optimal planner CPT, and the suboptimal planner POPF (Coles et al. 2010) comparative results for the readers interest [1].

Experiments

We first test the scalability of our approach on benchmark test cases before demonstrating its effectiveness on a real case study. All experiments were performed single threaded on a 2.40GHz Intel Processor, with processes time-

[1] CPT ver. 4, and POPF ver. 2, from the 2011 IPC.

or memory-out after 2 hours or 2 GB. The MIPs were solved using Cplex (v.12.5) with tuned pre-processing and branch-and-cut settings for different difficulty classes. For the interested reader, these included switching off Gomory cuts for feasibility problems with many precedences, turning off cutting plane generation and branch on pseudo reduced costs for no precedences. For the model with an objective function, mixed-integer rounding cuts were generated moderately for many and no precedences, while for instances with moderately many precedences, Gomory cuts were generated aggressively. Since F is a feasibility problem, for this problem we set Cplex to emphasise feasibility, and the algorithm stops once any feasible solution is obtained. However, for H and H_s, we run Cplex to optimality. CPT is run with the conflict counting heuristic option, and POPF runs only the EHC fast but incomplete search, which, despite being incomplete, always finds a solution. POPF-BFS search is disabled, as it does not improve POPF-EHC solutions over 30 min.

Analysis on Benchmark Problems

We varied the following key parameters in our analysis:

- number of blocks (20–30, increments of 5);
- number of excavators ($3 \leq |L| \leq 5$, increments of 1);
- block capacities (sampled [normal distribution] with $\mu = 50000$ and $\sigma = 30000$);
- tightness of blend constraints (sampled [uniform distribution] outside bounds by 200%.)

These criteria give rise to 450 test cases with 50 random instances generated for the block and excavator variants. The topology of the test instances, are designed to be similar to real-life examples: few blocks are clustered. To create the topologies, we scatter blocks and excavators using a uniform distribution along the x and y axis in a grid. We then mapped the blocks from the grid to a graph, $G_T = (\mathcal{V}_T, \mathcal{E}_T)$, such that each edge defines the direct path between two nodes, and each node defines a block and excavator initial position.

In Table 1, we present the results from experiments with the three versions of the MIP: F, H, and H_s, each with CPT and POPF. Of the 450 generated instances, 113 were infeasible due to the high grade variance. These were all detected by the MIP solver in less than $0.05s$. POPF was able to solve all the benchmark instances given any of the partitions, while CPT timed out on most of the instances with 30 blocks . As expected, the Cplex run-time for MIP F is faster than for H, which uses an objective function; while H_s is the slowest, as it requires the blocks to be partitioned among adjacent stockpiles—that is, it has an additional constraint. Nevertheless, their performance is extremely fast, only taking 0.01, 0.02, and 0.04 seconds respectively. POPF is on average 2 orders of magnitude faster than CPT. Interestingly, both planners were able to realise the partitions computed by F faster than those computed by both heuristic MIPS, with H_s producing faster planning solutions than H. These results can be understood by looking at the plan length of the full solution: F plans are shorter than plans for H and H_s, which results from the heuristic MIPs trying to partition more blocks than F in order to share the workload among all excavators, thus resulting in more digging actions. Note that plans arising from H_s are shorter than those from H. This is due to partitions occurring in adjacent stockpiles, which results in less moving actions. This also has an important impact on computation time for the planner.

Minimising makespan is of extreme importance for our industrial partner, as each excavator, on average, can extract 3000 Tons in 60 minutes—shorter makespans translate directly to increased overall production. In our experiments, the heuristic MIPs improve makespan over F with either planner. While H_s marginally improves the makespan only for POPF, there is no impact for CPT. The key to understanding this is that more freedom is allowed in H, than H_s, to partition the blocks among distant stockpiles if it leads to a better workload balance. Since CPT is optimal, this restriction is clear in the increased makespan. However, for POPF, the additional freedom from H leads to much longer suboptimal movements. Overall, CPT yields a significantly shorter makespan than POPF, thereby evidencing the impact of computing optimal plans for each subproblem.

Solvers	I	S	T	#P	M
F/POPF	450	(113) 450	(0.01) 199	57.66	7491.08
H/POPF	450	(113) 450	(0.02) 2.56	68.60	7429.84
H_s/POPF	450	(113) 450	(0.04) 2.14	60.49	7373.48
F/CPT	450	(113) 375	(0.01) 294.00	54.30	6285.57
H/CPT	450	(113) 328	(0.02) 711.97	68.68	5992.51
H_s/CPT	450	(113) 363	(0.04) 582.68	61.48	6173.74

Table 1: Benchmark results. I is the total number of instances, S is the number of "solved" instances (including those proved unsolvable by MIP in parentheses), T is (MIP) planner computation time in seconds, #P is plan length, and M is makespan in minutes. T, #P, and M are averaged among instances solved by all solvers. F, H, and H_s stands for feasability, heuristic MIP, and heuristic MIP with $SOS2$ respectively.

Analysis on Real Case Studies

From data provided by our industry partner, we selected the four most interesting (i.e., challenging) weeks, with at least 25 blocks across all pits. These schedules did not contain any complex precedences, so constraints (8)–(10) reduce to

$$\sum_{d' \leq d} x_{b,d'} \leq \sum_{d' \leq d, i \subset \Gamma(b)} y_{i,d'} \qquad \forall\, b, d,$$

thereby simplifying the difficulty of solving the MIP significantly. We generated a concatenated instance of the first three instances in order to build a more difficult instance with respect to precedences. We generated blend targets for four contaminants: iron, aluminium oxide, silicon dioxide and phosphorous. The permissible gaps of these grade bounds vary between contaminants from 0.006% to 0.4% of the final blend. Thus we have 5 instances ranging from 22–75 blocks with 1173–3704 kT, with an average block size of 50kT varying from 5–150kT, and contaminants grade ranging from within the bounds up to 30 times outside the min-

Data				(Plan Length) Makespan [minutes]					(MIP) Computation Time [seconds]				
Inst.	$\lvert B\rvert$	P	$\sum$	F/POPF	H/POPF	H_s/POPF	F/CPT	H/CPT	F/POPF	H/POPF	H_s/POPF	F/CPT	H/CPT
(1)	37	26	1614	(117) 12218	(140) 9704	(133) **8678.81**	—	—	(0.02) 10	(0.06) 12.69	(0.12) **9.67**	(0.02) —	(0.06) —
(2)	23	8	1173	(75) 8131	(109) **7130**	(95) 7209.87	(74) 7805	—	(0.01) 2.3	(0.04) 2.76	(2.20)**2.18**	(0.02) 249.65	(0.05) —
(3)	25	0	1449	(75) 14906	(110) 9475	(90) 8928.81	(75) 14659	(111) **7835**	(0.01) 2.65	(0.02) 3.44	(1.56)**2.48**	(0.01) 4715.69	(0.02) 6107.06
(4)	21	18	1375	(68) 12171	(94) 7645	(81) 7591.50	(78) 10438	(97) **6021**	(0.01) 1.16	(0.02) 33.78	(0.69)**1.05**	(0.01) 894.23	(0.02) 1694.99
(5)	66	68	3704	(255) 37405	(385) **26761**	—	—	—	(0.20) **74.49**	(0.37) 92.69	—	(0.20) —	(0.38) —

Table 2: The case study data and complete method results. In column (1) we list the instances. $|\mathcal{B}|$ refers to the number of blocks. P refers to the number of precedences. $\sum$ is the total ore in kilotonnes. F, H, and H_s stands for feasability, heuristic, and heuristic with SOS2 MIP respectively. Computation time is reported for (MIP) planner in seconds. A — indicates a timeout

imum and maximum blend targets. In all instances we permitted the average available equipment of 5 excavators.

The results of Table 2 over the real case scenarios are consistent with the results over the benchmark problems. $F/POPF$ and $H/POPF$ solve all instances, and $H_s/POPF$ solves all but instance 5, which represents 3 weeks of operations. Here, the $SOS2$ constraint renders the MIP too difficult and Cplex runs out of memory. F/CPT and H/CPT solve 3 and 2 instances, respectively. CPT times out in all instances trying to realise the partitions computed by H_s and is therefore not reported in the table. H and H_s partitions render longer plans than F with both planners, while H_s yields shorter plans than H with POPF, as both H and H_s try to partition more blocks to minimise the resulting makespan. As a result, the makespan of H and H_s partitions are significantly better than F with both planners. Remarkably, the heuristic MIP solutions computed with the suboptimal planner POPF yield a *much* smaller makespan than the solutions computed by the optimal planner CPT with just the feasible MIP F. While computing optimal plans improves the quality of the makespan, this result highlights the importance of finding 'good' partitions from blocks to stockpiles with the MIP solvers, which H and H_s are able to infer.

Discussion and Evaluation

Tables 1 and 2 present the computational results from our experiments using MIP F, H, and H_s with two planners, suboptimal POPF and optimal CPT. CPT obtains better solutions with respect to makespan, than POPF. We also see a sharp improvement in makespan when using the heuristic MIP compared with the feasibility MIP, even when the optimal planner CPT is used for the feasibility MIP and suboptimal POPF for H. The concatenated instance (5) presents significantly more precedences than the other instances, and therefore becomes more difficult to solve quickly. POPF is faster than CPT, so choice of planner becomes a choice between speed and quality. It is remarkable, also, that the addition of the SOS2 constraints (16) improved the allocation such that it was not only easier for the planner to find a solution, but also yielded better quality solutions.

We know, from experimentation, that it is not easy to outperform the feasibility problem F in terms of solution quality. For example, one heuristic we tried was to allocate the blocks according to distance from the excavators. This alone produced biased allocations that were worse than those provided by the feasible MIP. Contrary to intuition, only a small part of the differences in the makespan results are due to movement. The majority of the difference is due to a sharing of the workload amongst the excavators such that waiting time is minimised.

This work strongly emphasises both the practical importance of planning, and the importance of heuristics to drive the planner toward better solutions. On one hand, we have shown that planning can play a key role in solving a problem that has been left unaddressed in the literature in spite of its practical significance. On the other hand, our experiments have illustrated that good heuristics can be MIP-based, and can give rise to better results when combined with a suboptimal planner than simply using an uninformed decomposition MIP/planning approach.

Ideally, we would like to compare the results of our approach with that of a human mine planner. Unfortunately, this is not possible as we take the block set, B, from the short-term plan, while the mine planner is influenced by the current state of the mine. This means there is a mismatch between the block sets considered in each week. In future incarnations of this solver, we will take the current mine status as an input. These are necessary, anyway, if the planner wishes to use our tool on-the-fly with the most up-to-date information possible, but also permit comparisons.

Conclusions and Outlook

In this paper, we have presented a comprehensive illustration of how MIP and planning technology can work together to efficiently solve a real-world mining problem. In particular, we demonstrated an effective method for solving problems with state-dependent edge costs (in a network), and resource constraints. The basic approach itself is simple and re-deployable for other applications with similar requirements. Our key contribution in this paper is the development of a 'guiding' objective function, which helps the MIP to select solutions that have desirable properties for the planner.

Our industry partner has also indicated interest in an extension of the problem which involves including, as an option, blocks that are scheduled to be mined in the future. Such a tool would allow the Operations Planners to validate their intuition regarding schedule 'fixes' or 're-optimising' on-the-fly. As it is, the presented approach is also useful for Operations Planners to check weekly schedules for feasibility and to quickly derive good mining sequences for the excavators. Thus, it is a valuable tool to aid in the complex and expensive decision-making that occurs on mine sites.

Acknowledgements

The authors wish to thank Jon Lapwood and John Usher from Rio Tinto for their extensive discussions throughout our research. This research was co-funded by the Australian Research Council linkage grant LP11010015 “Making the Pilbara Blend: Agile Mine Scheduling through Contingent Planning” and industry partner Rio Tinto.

References

Benton, J.; Coles, A. J.; and Coles, A. 2012. Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of ICAPS*, 2–10.

Blazewicz, J.; Lenstra, J.; and Rinnooykan, A. 1983. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics* 5:11–24.

Bylander, T. 1997. A linear programming heuristic for optimal planning. In *Proceedings of AAAI/IAAI*, 694–699.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *Proceedings of ICAPS*, 42–49.

Demeulemeester, E., and Herroelen, W. 1992. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science* 38(12):1803–1818.

Even, S.; Itai, A.; and Shamir, A. 1975. On the complexity of time table and multi-commodity flow problems. In *Proceedings of FCS*, 184–193.

Fernández, S., and Borrajo, D. 2009. Solving clustered oversubscription problems for planning e-courses. In *Proceedings of SPARK Workshop*, volume 9.

Flórez, J. E.; de Reyna, A. T. A.; García, J.; López, C. L.; Olaya, A. G.; and Borrajo, D. 2011. Planning multi-modal transportation problems. In *Proceedings of ICAPS*, 66–73.

Fox, M., and Long, D. 2003. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Geoffrion, A. M., and Marsten, R. 1972. Integer programming algorithms: A framework and state-of-the-art survey. *Management Science* 18(9):465–491.

Hooker, J. N. 2005. A hybrid method for the planning and scheduling. *Constraints* 10(4):385–401.

Kautz, H., and Walser, J. P. 1999. State-space planning by integer optimization. In *Proceedings of AAAI/IAAI*, 526–533.

Lenstra, J., and Rinnooykan, A. 1978. Complexity of scheduling under precedence constraints. *Operations Research* 26:22–35.

Martinez, M. A., and Newman, A. M. 2011. A solution approach for optimizing long-and short-term production scheduling at lkab’s kiruna mine. *European Journal of Operational Research* 211(1):184–197.

Nakhost, H.; Hoffmann, J.; Müller, M.; et al. 2012. Resource-constrained planning: A monte carlo random walk approach. In *Proceedings of ICAPS*, 181–189.

Ryan, D. M., and Foster, B. A. 1981. An integer programming approach to scheduling. *Computer scheduling of public transport urban passenger vehicle and crew scheduling* 269–280.

Sandeman, T.; Fricke, C.; Bodon, P.; and Stanford, C. 2010. Integrating optimization and simulation—a comparison of two case studies in mine planning. In *Proceedings of WSC*, 1898–1910.

Smith, M. L. 1998. Optimizing short-term production schedules in surface mining: Integrating mine modeling software with ampl/cplex. *International Journal of Mining, Reclamation, and Environment* 12:149–155.

Van Den Briel, M.; Benton, J.; Kambhampati, S.; and Vossen, T. 2007. An lp-based heuristic for optimal planning. In *Principles and Practice of Constraint Programming–CP 2007*. Springer. 651–665.

Vidal, V., and Geffner, H. 2006. Branching and pruning: An optimal temporal pocl planner based on constraint programming. *Artificial Intelligence* 170(3):298–335.

Vossen, T.; Ball, M.; Lotem, A.; and Nau, D. 2000. Applying integer programming to ai planning. *The Knowledge Engineering Review* 15(1):85–100.