Computers & Operations Research 37 (2010) 1641–1647

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A strengthened formulation and cutting planes for the open pit mine production scheduling problem

Andreas Bley [a,*], Natashia Boland [b], Christopher Fricke [c], Gary Froyland [d]

[a] Technische Universität Berlin, Straße des 17. Juni 136, 10623 Berlin, Germany
[b] School of Mathematical and Physical Sciences, University of Newcastle, Callaghan NSW 2308, Australia
[c] TSG Consulting, Level 11, 350 Collins Street, Melbourne, VIC 3000, Australia
[d] School of Mathematics and Statistics, University of New South Wales, Sydney, NSW 2052, Australia

ARTICLE INFO

Available online 4 January 2010

Keywords:
Integer programming
Mine optimisation
Scheduling

ABSTRACT

We present an integer programming formulation for the open pit mine production scheduling problem. We strengthen this formulation by adding inequalities derived by combining the precedence and production constraints. The addition of these inequalities decreases the computational requirements to obtain the optimal integer solution, in many cases by a significant margin.

1. Introduction

The design and scheduling of open pit mines is a significant and complex problem in mine planning. The most commonly used representation of an ore body for the purposes of mathematical modelling is as a *block model*, see e.g. Hustrulid and Kuchta [11]. The ore body is divided into a regular three dimensional array of blocks, each of which has individual *attributes* such as the tonnage of rock and ore contained within the block, assigned using geological techniques, and the expected in-ground value, based on current costs and forecasted commodity prices. The principal aim of a mining operation is to ensure that an ore body is exploited so as to maximise the value realised from the mine. A well-known early contribution to this field was made by Lerchs and Grossmann [14], who presented a graph-theoretic algorithm for determining the final contour of the open pit, known as the *ultimate pit*, such that the total profit from the mine is maximised. Since the life of an open pit mine can be as long as 40 years, recent developments have adopted the net present value of the operation as the criterion for mine project evaluation. The open pit mine production scheduling problem seeks to determine the sequence in which material should be removed over the lifetime of the mine in order to maximise net present value. The removal of material is contingent upon the removal of a cone of material situated above it, the size and shape of which is dictated by the requirement of safe wall slopes for the pit. This is modelled in the *precedence constraints* for the mine. An additional class of constraints are the *production constraints*, imposed by the availability of extraction and processing capacity in each year. Techniques applied to solve the mine production scheduling problem include heuristics [10], parametric methods [21], dynamic programming [15,20,18] and integer linear programming [9,17,6]. The major limitation with these approaches is that they encounter significant computational difficulties when trying to solve problems of realistic size.

In this paper, we consider the integer linear programming formulation presented by Caccetta and Hill [6]. For each time period and each attribute, the corresponding production constraint and the precedence constraints among the blocks combine to form a so-called precedence constrained knapsack problem (PCKP); see e.g. [7,16]. Exploiting the special structure of these subproblems, we derive several types of additional constraints, whose addition to the standard integer programming formulation leads to a tighter linear relaxation and a substantial reduction of the computation times required to solve open pit mine production scheduling problems.

In Section 2 we present the integer programming formulation of the open pit mine production scheduling problem. In Section 3 we derive constraints that involve only a single variable from the precedence constrained knapsack substructure. We illustrate how the production and precedence constraints can be combined to identify the earliest time period in which each single block of material can be extracted from the pit. The computation of these earliest extraction times can be performed very efficiently in polynomial time. Adding the corresponding constraints to the integer programming formulation eliminates a number of decision variables from the model prior to optimisation.

In the following sections we explain how the precedence constrained knapsack substructure can be used to derive additional inequalities that involve multiple variables. In Section 4, we introduce general *induced cover inequalities* and give a brief overview of the literature related to these inequalities.

* Corresponding author.
E-mail addresses: bley@math.tu-berlin.de (A. Bley), natashia.boland@newcastle.edu.au (N. Boland), g.froyland@unsw.edu.au (G. Froyland).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2009.12.008

In Section 5 we consider the special case of induced covers of size two, which can be regarded as pairwise conflicts between the variables in our model. Considering cliques in the graph defined by all these pairwise conflicts, we derive stronger *clique inequalities* for the precedence constrained knapsack subproblems and, as a consequence, for the open pit mine production scheduling problem. We also describe a heuristic separation algorithm for these inequalities, which is very efficient in practice. In Section 6 we return to the general induced cover inequalities involving more than two variables, describing an exact model for the corresponding separation problem and explaining how we perform this separation in practice.

The computational effects of the variable elimination and the addition of clique and induced cover inequalities at the root node of the branch and bound tree are discussed in Section 8. The numerical results demonstrate that the variable elimination is extremely effective in terms of both run time and LP gap reduction, and that the addition of clique and cover constraints can further reduce the root node gap.

Finally, in Section 8.4, we tested the effectiveness of the variable elimination and the additional inequalities derived by the precedence and production considerations when applied at nodes deeper in the branch and bound tree. While most of the computational benefits are already gained by applying these constraints at the root node, we find some modest additional improvements from variable elimination and the application of clique inequalities within the branch and bound tree.

2. Integer programming formulation

In this paper, we consider the integer program presented by Caccetta and Hill [6]; see also [7]. The pit contains N blocks, and is to be scheduled over T time periods. The extraction of block i in time period t results in a discounted cash flow of c_i^t units. A set of attributes A of concern to the mining operation is determined when generating the block model, for example the tonnes of ore and tonnes of impurities contained in each block. Hence each block i is assigned a value q_i^a for each attribute $a \in A$. A bound on the tonnage of each attribute able to be processed in time period t is given by u_a^t. The minimal set of blocks that must be removed for block i to be removed, including block i, is denoted by E_i. Frequently E_i has the three dimensional shape of an inverted cone. Note that for each block i it is sufficient to consider a smaller set of predecessor blocks S_i; see Fig. 1. The decision variables of the model take the form

$$x_i^t = \begin{cases} 1 & \text{if block } i \text{ is mined by the end of period } t, \\ 0 & \text{otherwise,} \end{cases} \qquad i = 1,\dots,N,\ t = 1,\dots,T.$$

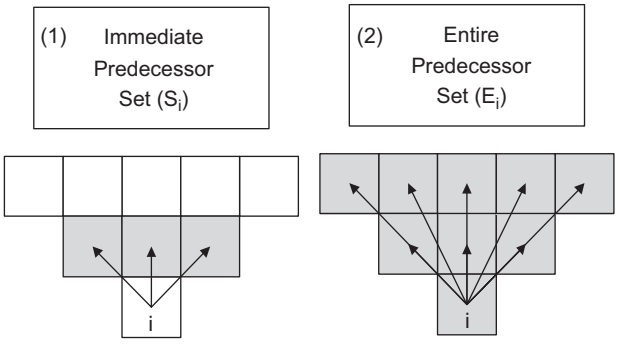

Fig. 1. Illustration of: (1) The immediate predecessor set. (2) The entire predecessor set.

For each block, these variables describe the period when the block is mined in an incremental way: if block i is mined in period t', we have $x_i^t = 0$ for all $t = 1,\dots,t'-1$ and $x_i^t = 1$ for all $t = t',\dots,T$. If a block is not mined at all, we have $x_i^t = 0$ for all $t = 1,\dots,T$. For ease of notation throughout this paper, we introduce a set of dummy decision variables $x_i^0, i = 1,\dots,N$, which are assigned the value 0. The generalised integer program for the open pit mine production scheduling problem (OPMPSP) is then given by

$$\max \quad \sum_{t=1}^{T}\sum_{i=1}^{N} c_i^t (x_i^t - x_i^{t-1}) \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} q_i^a (x_i^t - x_i^{t-1}) \le u_a^t, \quad t = 1,\dots,T,\ a \in A \tag{2}$$

$$x_i^{t-1} \le x_i^t, \quad t = 1,\dots,T,\ i = 1,\dots,N \tag{3}$$

$$x_i^t \le x_j^t, \quad t = 1,\dots,T,\ i = 1,\dots,N,\ j \in S_i \tag{4}$$

$$x_i^0 = 0, \quad i = 1,\dots,N \tag{5}$$

$$x_i^t \in \{0,1\}, \quad t = 1,\dots,T,\ i = 1,\dots,N. \tag{6}$$

The objective (1) is to maximise the net present value of the mining operation. Constraints on the amount of each attribute $a \in A$ able to be extracted or processed in each time period are enforced by (2). These constraints are generally referred to as production constraints. There are also constraints enforcing the rule that each block can be mined at most once (3), and the block precedence constraints (4). The formulation (1)–(6) contains NT binary decision variables. A moderate open pit mine production scheduling problem consists of 100,000 blocks to be scheduled over 10 years, requiring 1,000,000 binary decision variables. Efficient solution of integer programs of this size is beyond the scope of most commercial solvers on current hardware. In Sections 3 and 4, we motivate and develop valid inequalities that, when added to the formulation, substantially reduce LP relaxation gap and the computation time required to find the optimal integer solution in almost all cases.

3. Strengthening the formulation: single block case

Our aim is to reduce the number of decision variables in the formulation (1)–(6) by identifying variables that can be fixed prior to optimisation. This is achieved by combining the block precedence constraints (4) with the production constraints (2), aggregated over a sequence of time periods for a particular attribute. We define the total set of immediate predecessor arcs $\mathcal{S} = \{(i,j) : i \in \{1,\dots,N\}, j \in S_i\}$. Let $\mathcal{E}$ be the transitive closure of $\mathcal{S}$. Then for each block i, the entire set of blocks including block i that must be extracted for block i to be mined is given by $E_i = \{i\} \cup \{j : (i,j) \in \mathcal{E}\}$. For the extraction of block i in time period t to be feasible, each of the blocks $j \in E_i$ must be removed in time period t or earlier. If, for any attribute, the production capacity required to achieve this exceeds the cumulative production capacity available up to and including time period t, block i cannot be extracted in time period t or earlier, that is, $x_i^s = 0, 1 \le s \le t$. Hence, we have that

$$x_i^t = 0 \text{ if } \sum_{j \in E_i} q_j^a > \sum_{s=1}^{t} u_a^s \text{ for some } a \in A. \tag{7}$$

Using this observation, a number of variables in the model (1)–(6) can be fixed to 0 prior to optimisation, resulting in a reduction in the number of decision variables in the model.

4. Strengthening the formulation: multiple block case

We define the union of the entire precedence sets for the blocks in the set $X \subset \{1,\dots,N\}$ to be $E(X) = \cup_{i \in X} E_i$. We extend the idea developed in the single block case by considering the blocks in $E(X)$ and the cumulative production capacity available up to and including time period t. This relationship can be described by aggregated production constraints for the blocks in the set X, again derived by summing (2) over time periods $1,\dots,t$. It is clear that the inequality

$$\sum_{i \in E(X)} q_i^a x_i^t \le \sum_{s=1}^{t} u_a^s \tag{8}$$

is a valid knapsack inequality for the mine scheduling polytope; see Fig. 2. The aggregated production constraint for attribute a and time period t, given by (8), combined with the precedence constraints (4) on the time period t decision variables describe a polytope known as the *precedence constrained knapsack* (PCK) polytope. Let us denote such an instance as $\mathrm{PCK}_{t,a}$. We call $i,j \in \{1,\dots,N\}$ *incomparable* if $(i,j) \notin \mathcal{E}$ and $(j,i) \notin \mathcal{E}$. Following [16], we call a collection $X \subset \{1,\dots,N\}$ a *minimal induced cover* for attribute a and period t if (i) elements of X are pairwise incomparable, (ii) $\sum_{i \in E(X)} q_i^a > \sum_{s=1}^{t} u_a^s$, and (iii) $\sum_{i \in E(X \setminus j)} q_i^a \le \sum_{s=1}^{t} u_a^s$ for all $j \in X$. If condition (iii) is not satisfied, we call the collection X an *induced cover*. When X is an induced cover or minimal induced cover, the *induced cover inequality*

$$\sum_{i \in X} x_i^t \le |X| - 1 \tag{9}$$

is clearly valid for both $\mathrm{PCK}_{t,a}$ and the mine scheduling polytope. As noted in [16], given X with pairwise incomparable elements, if one were to augment X with predecessors of elements of X, this would only weaken (9), and thus there is no loss in the strength of (9) by enforcing incomparability on elements of X. Minimal induced covers are also discussed in [5,19] with condition (iii) replaced by the stronger (iii′) $\sum_{i \in E(X) \setminus j} q_i^a \le \sum_{s=1}^{t} u_a^s$ for all $j \in X$; in the sequel we assume that (iii) applies when using the term minimal.

Note that (7) from Section 3 is a special case of (9), since it is an inequality for minimal induced covers X with $|X| = 1$.

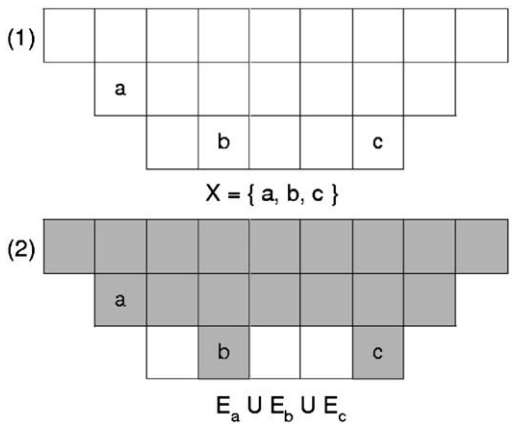

Fig. 2. Illustration of: (1) A possible induced cover set X (where each block's immediate predecessors are the block directly above and the two blocks located directly above at 45° to the left and right). (2) The union of the entire precedence sets $E(X)$.

4.1. Polyhedral structure of the precedence constrained knapsack polytope

Garey and Johnson [8] showed that the PCKP is NP-complete. Pseudo-polynomial time algorithms for PCK problems have been developed for simple classes of precedence graphs [12,13,1,22], however, the precedence graphs arising in the mining context generally do not fall into these classes. The polyhedral structure of the problem was first investigated by Boyd [5], who generalised results corresponding to finding cover inequalities for the standard knapsack polytope. In particular, Boyd derived conditions under which inequalities of the form (9) are facet-defining for the convex hull of the feasible solutions to $\mathrm{PCK}_{t,a}$ restricted to the variables in $E(X)$. Further investigation of the PCK polytope is presented by both Park and Park [16] and van de Leensel et al. [19], where lifting orders and general sequential lifting procedures are derived to lift valid inequalities from lower dimensional faces of the PCK polytope into facets of the PCK polytope. Note that the polytope of the open pit mine production scheduling problem lies in the Cartesian product of the $\mathrm{PCK}_{t,a}$ polytopes for all $t \in \{1,\dots,T\}$ and $a \in A$. Hence inequalities that are facet-defining for $\mathrm{PCK}_{t,a}$ will not necessarily be facet-defining for the mine production scheduling problem.

5. Clique constraints based upon two block conflicts

To test the effectiveness of the induced cover inequalities, we apply them in a cutting plane approach at the root node and, later, within the branch and bound tree. First, we consider only minimal induced covers X with $|X| = 2$.

Let $t \in \{1,\dots,T\}$ be a time period, $i,j \in \{1,\dots,N\}$ be two distinct incomparable blocks, and suppose $X = \{i,j\}$ is a minimal induced cover for some attribute in period t. The corresponding induced cover inequality for this pairwise conflict

$$x_i^t + x_j^t \le 1 \tag{10}$$

is a valid inequality for the formulation (1)–(6). Clearly, there are at most $TN(N-1)/2$ such pairwise conflicts, and they can be computed easily in polynomial time.

Instead of adding these inequalities directly to the OPMPSP formulation, we take a cutting plane approach and add violated maximal clique inequalities based on these pairwise conflicts. Let $C \subseteq \{1,\dots,N\}$ be a set of blocks such that each pair $\{i,j\} \subseteq C$ is a minimal induced cover for some attribute in time period t. We call such a set a *clique of conflicting blocks* for period t. If there is no block $k \notin C$ such that $C \cup \{k\}$ forms a larger clique, we say that C is a maximal clique. Since at most one of the blocks in any such clique C can be mined by period t, the corresponding (maximal) clique inequality

$$\sum_{i \in C} x_i^t \le 1 \tag{11}$$

is valid for (1)–(6). It is not difficult to see that the clique inequality (11) implies all pairwise induced cover inequalities (10) for all $\{i,j\} \subseteq C$, but the linear programming relaxation obtained by adding (11) to formulation (1)–(6) is stronger than the one obtained by adding inequalities (10); see [23] for example. One can also lift in a common predecessor j of all blocks in the clique C to obtain a stronger inequality of the form

$$\sum_{i \in C} x_i^t \le x_j^t. \tag{12}$$

The conditions when the clique inequalities (11) and the lifted clique inequalities (12) are facet defining are discussed in [2,7].

Algorithm 1 summarises the construction of these clique inequalities.

Algorithm 1. Generation of clique inequalities.

1. For each time period $t = 1, \ldots, T$, create a graph of pairwise conflicts. The N vertices of the graph correspond to the N variables x_i^t, $i = 1, \ldots, N$. Each edge in the graph corresponds to a pairwise conflict that gives rise to a two block induced cover inequality (10).
2. Set vertex weights to the current primal LP solution x_i^t.
3. Compute an (approximate) maximum weight clique C.
4. If there is a common predecessor of all blocks in C, lift in the minimum weight predecessor block j to replace the RHS of (11) with x_j^t; see [2,7] for further details.
5. If the resulting lifted inequality is violated (by more than the minimum violation threshold of 0.001), add the lifted inequality as a globally valid cut to the model.

In general, the problem of finding a maximum clique in Step 3 is NP-hard. However, there are numerous efficient heuristics and exact solution methods for finding maximum weight cliques in general graphs that can be applied. Bomze et al. [3] provide a comprehensive overview of such methods. In our implementation we use the Sequential Greedy heuristic and the combinatorial branch and bound algorithm proposed in [4], which is based on this heuristic, with a limit of at most 100 branch and bound nodes to explore. We remark that Algorithm 1 does not solve the separation problem exactly (which is NP hard), but it does however find a violated clique inequality (11) or lifted clique inequalities (12) if there is a violated two block induced cover inequality (10).

6. Cover constraints based upon multiple block conflicts

To test the effectiveness of the induced cover inequalities for minimal induced covers of size greater than two, we apply them in a cutting plane approach, using a straightforward integer program to solve the separation problem exactly.

Applied at the root node only, the separation problem for the minimal induced cover inequality can be formulated as follows. Given a fractional solution $\hat{x}$ to the root node relaxation, we seek a time period t, an attribute a, and a minimal induced cover X for t and a, such that (9) is violated by $\hat{x}$, i.e. such that

$$\sum_{i \in X} \hat{x}_i^t > |X| - 1. \tag{13}$$

If a violated minimal induced cover inequality is found, the corresponding constraint is added to the formulation. We provide an integer programming formulation of this separation problem below.

A model of the separation problem must count each block in the union of the entire precedence sets of the blocks in X exactly once, as in the development of the aggregated production constraints for multiple blocks. To derive an integer programming model of the separation problem, for each time period t, we let

$$z_i = \begin{cases} 1 & \text{if block } i \text{ is included in the set } X, \\ 0 & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, N,$$

$$w_i = \begin{cases} 1 & \text{if block } i \text{ is included in } E(X), \\ 0 & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, N.$$

Given a fractional solution $\hat{x}$ of (1)–(5), the problem of finding an induced cover X for attribute a and period t such that X violates the associated induced cover inequality (9) can be formulated as the following integer program (assuming that $q_i^a \in \mathbb{Z}$, $q_i^a \geq 0$ for all i and $u_a^s \in \mathbb{Z}$, $u_a^s \geq 0$ for all s):

$$\begin{aligned} \max \quad & \sum_{i=1}^{N} (\hat{x}_i^t - 1) z_i \\ \text{s.t.} \quad & \sum_{i=1}^{N} q_i^a w_i \geq \sum_{s=1}^{t} u_a^s + 1 \\ & \sum_{j : i \in E_j} z_j \geq w_i \quad \forall i = 1, \ldots, N \\ & w_i \leq 1 \quad \forall i = 1, \ldots, N \quad z_i \in \{0, 1\} \quad \forall i = 1, \ldots, N. \end{aligned}$$

If the optimal value of this integer program is strictly greater than -1, the set $X := \{i : z_i = 1\}$ is an induced cover for attribute a and period t, whose induced cover inequality (9) is violated. If the optimal value of this integer program is less or equal to -1, then (9) is satisfied for all induced covers for attribute a and period t. Finally, if the integer program is infeasible, then there is no induced cover for attribute a and period t at all.

From possibly several optimal solutions of the above integer program we select one which minimises $\sum_{i=1}^{N} z_i$, using an appropriate perturbation of the objective function in computation. As the cover inequalities resulting from minimal induced covers dominate those resulting from non-minimal induced covers, this is sufficient to ensure that the cover X corresponding to this solution is indeed a maximally violated minimal induced cover.

The resulting inequalities are applied if they are violated by an amount greater than 10^{-3}.

7. Locally valid constraints

If the OPMPSP formulation (1)–(6) is solved using an algorithm that is based on branch and bound, the constraints described in Sections 3, 5, and 6 can also be applied within the branch and bound tree. At each node of the tree, several binary variables will already have been fixed. By considering these variable fixings in the variable elimination and separation procedures, we may derive stronger constraints that are locally valid for a particular node.

7.1. Variable fixing based upon single blocks

Consider a node of the branch and bound tree, where for some period t the variables x_i^t, $i \in F \subset \{1, \ldots, N\}$, have been fixed to 1 by the branching decisions. The precedence relations among the blocks then imply that all variables x_i^t, $i \in E(F)$ are fixed to 1, which only leaves a residual capacity of $\sum_{s=1}^{t} u_a^s - \sum_{i \in E(F)} q_i^a$ of resource $a \in A$ for the remaining blocks. In the spirit of (7), in the current subproblem we therefore can fix

$$x_i^t = 0 \text{ for all } i \notin E(F) \text{ with } \sum_{j \in E(i) \setminus E(F)} q_j^a > \sum_{s=1}^{t} u_a^s - \sum_{i \in E(F)} q_i^a \text{ for some } a \in A.$$

Of course, these fixings are valid only for the subproblem corresponding to the current branch and bound node.

7.2. Clique and cover constraints

To apply separation of clique and cover constraints within the branch and bound tree, the variable fixings corresponding to the current node of the branch and bound tree are additionally imposed upon the corresponding separation problems.

At each node of the branch and bound tree, we consider the residual problem obtained by removing all blocks i with x_i^t fixed to 0 or 1 and reducing the residual capacities accordingly. Applying Algorithm 1 for the conflict graph corresponding to the residual problem at each node, locally valid clique constraints are generated. Locally valid minimal induced cover inequalities are generated by solving the auxiliary integer program presented in Section 6 for the residual problem at the nodes.

8. Numerical results

We evaluated the impact of the techniques developed in Sections 3, 5, and 6 on two pits provided by our research partner BHP Billiton. The block models for each pit were tested at four different resolutions, to investigate the effect of our inequalities on models with different block sizes. Note that we only store the average data values for each block, so a larger number of blocks represents greater data resolution.

Pit 1 is a shallow pit, where the height of the blocks is constant at each of the four resolutions. Hence the lower resolution instances correspond to flat block shapes, which become more cubic as the resolution is increased. So for Pit 1 all blocks can be accessed early in the mine's life, and there are relatively few precedence constraints in the formulation.

Pit 2 is a deeper pit, where the number of layers of blocks is also increased as the resolution is increased. Hence for Pit 2 the block shapes are approximately cubic at all resolutions, and there is a greater number of precedence constraints per block in the formulation. Moreover, there are many deep blocks that cannot be accessed until later in the mine's life.

All models have a mine life of 15 years. Hence the cases tested in Table 1 correspond to five time periods of three years each, and 10 time periods of 18 months each. In all cases the block models contain two attributes, namely the tonnes of rock and the tonnes of ore in each block. The capacity constraints are such that the entire ore body can be extracted over the life of the mine, with the production capacity for each attribute identical in all time periods.

All formulations were solved using CPLEX v11.0 on an Intel Pentium 4 machine with 2.66 GHz and 4 GB RAM running Linux 2.6. Apart from applying CPLEX's built-in heuristics more aggressively, the default settings for mixed integer optimisation in CPLEX were used; all integer programs were solved to an optimality gap of 1%. In all tables in this paper, we calculate the root node gap using the LP relaxation after CPLEX has performed its automatic preprocessing algorithms (node 0^+ in CPLEX terminology). As expected, it is evident from Table 1 that dividing the life of the mine into smaller time periods significantly increases the difficulty of the integer program, in terms of both the CPU time and the number of branch and bound nodes required to find the optimal integer solution. In addition, the resolution of the block model has a significant effect on the difficulty of solving the integer program to optimality. All CPU times reported include the CPLEX solve time and the time taken by auxiliary algorithms such as calculating the weights of the predecessor sets E_i, forming conflict graphs, and finding maximum weight cliques and induced covers.

8.1. Variable fixing based upon single blocks

The computational advantage of fixing variables to zero according to (7) is shown in the columns labelled "with variable elimination" in Table 1. In each of the 16 instances tested, the strengthened formulation gave a significantly tighter root node relaxation than the standard formulation. The reduction in the gap of the root node relaxation is most pronounced for the higher resolution versions of Pit 2. This is to be expected as there is a higher proportion of deep blocks i for which x_i^t may be fixed to 0 for several early periods t. The strengthened formulation outperformed the standard formulation in terms of solution time in all instances, with the greatest reductions occurring for the larger instances of Pit 2. We remark that—even with the most aggressive setting for variable probing—constraints (7) are not automatically detected and applied in the preprocessing phase by CPLEX.

8.2. Cliques based upon two block conflicts

After fixing variables as in the previous section, we applied Algorithm 1 in a cutting plane framework at the root node of the branch and bound tree until no further clique inequalities could be found to cut off the current fractional solution. The results are

Table 1
Computational results with fixing and cuts added at the root node.

Data set			Standard formulation				With variable elimination				With clique constraints				With cover constraints				
N	$\lvert\mathcal{E}\rvert$	T	Vars	B&B nodes	Root node gap (%)	Time (s)	Fixed	B&B nodes	Root node gap (%)	Time (s)	Clique cuts	B&B nodes	Root node gap (%)	Time (s)	Clique cuts	Cover cuts	B&B nodes	Root node gap (%)	Time (s)
Pit 1																			
67	190	5	335	407	5.2	4.8	46	432	2.8	2.8	14	142	1.5	2.6	17	23	50	1.1	4.4
67	190	10	670	41,723	9.6	531.4	103	59,319	4.6	444.3	38	102,700	2.9	1589.7	43	45	66,442	2.4	728.8
115	368	5	575	225	4.1	11.2	40	97	2.3	8.6	18	50	1.2	9.3	21	23	1	1.0	14.3
115	368	10	1150	5925	5.1	224.0	106	2882	2.6	88.5	34	1996	1.5	73.6	37	40	950	1.4	79.6
182	615	5	910	100	2.0	22.9	45	214	1.8	17.8	14	93	1.2	10.8	14	24	3	1.0	31.4
182	615	10	1820	4136	3.4	505.6	101	954	2.0	119.5	29	1356	1.3	143.9	34	55	1250	1.1	181.5
354	1670	5	1770	3	1.0	12.2	21	1	0.9	9.1	8	1	0.8	9.0	10	23	1	0.8	42.4
354	1670	10	3540	239	1.1	120.5	93	300	1.0	92.9	10	431	1.0	123.5	12	46	50	0.9	154.5
Pit 2																			
66	312	5	330	62	3.2	1.1	38	50	2.4	0.9	6	3	1.1	0.4	6	7	1	0.9	0.5
66	312	10	660	1567	7.2	42.0	94	1750	5.1	35.0	13	539	1.9	11.5	18	52	900	1.0	20.8
90	544	5	450	2102	6.2	9.2	54	75	4.5	1.6	9	1	0.9	0.2	11	30	1	0.4	3.4
90	544	10	900	10,473	9.1	213.4	140	1387	5.2	34.2	25	599	2.2	21.8	27	41	72	1.9	13.3
166	1551	5	830	460	25.6	34.7	209	312	5.6	11.9	21	187	4.1	11.6	22	22	26	2.7	12.2
166	1551	10	1660	44,359	37.0	2477.8	475	13,545	6.8	317.9	21	3663	4.7	117.9	34	53	1185	2.5	113.0
420	5900	5	2100	2514	37.2	487.2	620	445	9.4	112.3	11	462	5.3	153.8	23	47	323	3.4	491.8
420	5900	10	4200	71,268	55.2	61,224.5	1385	3739	9.2	1023.8	19	6417	5.6	3114.7	41	115	1887	4.2	5032.2

presented in the columns labelled "with clique constraints" in Table 1. The addition of clique inequalities made further significant reductions to the root node gap across all instances. With some exceptions, adding the clique inequalities also produced fewer branch and bound nodes and faster solution times than the variable fixing based on single blocks alone.

8.3. Covers based upon multiple block conflicts

After fixing variables and adding clique inequalities as in the previous two sections, we searched for separating induced cover inequalities using the integer programming formulation given in Section 6, and applied them in a standard cutting plane approach at the root node of the branch and bound tree. At each iteration of the cutting plane algorithm, we first search for violated cliques as in the previous section, and only if no clique inequality is found, do we apply the separation algorithm from Section 6. The results are shown in the columns labelled "with cover constraints" in Table 1.

The greatest benefit from adding induced cover inequalities for multiple blocks is that in all cases, the root node gap has decreased when compared to the gaps reported in the single and two block columns of Table 1. The decrease in root node gap is most pronounced for the larger instances of Pit 2, which are the most computationally demanding problems. This indicates that there is potential for further computational improvement if these inequalities can be derived efficiently.

Clearly the computation time for finding violated minimal induced cover inequalities by solving an integer program is prohibitive. Even though minimal induced cover inequalities have been generated only at the root node of the branch and bound tree, the proportion of time spent in the separation subroutine ranged from 1% to 88% of the overall solution time, with an average of 70% over all instances. However, the structure of the separation problem is quite nice; it is close to a precedence constrained knapsack problem. Hence finding efficient exact algorithms or effective heuristics may be possible; this is a direction for future research.

8.4. Locally valid constraints

As we show in Table 2, the impact of performing locally valid variable fixing and adding locally valid inequalities is mixed.

Variable elimination within the branch and bound tree resulted in a reduction (or no change) in the number of branch and bound nodes over performing variable elimination at the root node alone in all but two instances; see the "with variable elimination" columns of Table 2. In terms of solution time, further improvements were seen in some instances, although in other instances, the auxiliary calculations required overtook the modest gains in solution time.

The addition of clique inequalities based on two-block conflicts in the branch and bound tree produces similar improvements. In the "with clique constraints" columns of Table 2 we see a reduction in branch and bound nodes over the application of clique inequalities at the root node only in all but one instance. However, while some solution times improve, the results are mixed, due largely to the required auxiliary calculations.

The addition of multiple block induced cover inequalities within the branch and bound tree became too expensive in terms of computation time; see the "with cover constraints" columns of Table 2. The time required for solving the separation problem at every node of branch and bound tree dramatically increases the overall solution times. The proportion of time spent in solving the artificial integer programs in the separation subroutine for the multiple block induced cover inequalities ranged from 65% to 98% of the overall solution time, with an average of 97% over all instances. The time spent on variable elimination and the generation of clique inequalities together, on the other hand, ranged between 0.1% and 0.9% of the overall solution time only.

To avoid the expensive multiple block induced cover calculations, we also tried applying variable elimination and clique inequalities in the branch and bound tree, but applying multiple block induced cover inequalities at the root node only. While this was of course faster than applying multiple block induced cover inequalities within the tree, the results in terms of branch and bound nodes and solution times were not much different than applying all procedures at the root node only. We therefore have not included these results in Table 2.

9. Conclusions and future work

The single block variable elimination derived in Section 3 was found to be highly beneficial in the majority of cases tested, in

Table 2
Computational results with local fixing and local cuts in the branch and bound tree.

Data set			With variable elimination			With clique constraints				With cover constraints				
N	$\|\mathcal{E}\|$	T	Fixed	B&B nodes	Time (s)	Fixed	Clique cuts	B&B nodes	Time (s)	Fixed	Clique cuts	Cover cuts	B&B nodes	Time (s)
Pit 1														
67	190	5	46	432	3.2	46	27	107	3.0	46	20	151	50	32.0
67	190	10	405	55,877	488.4	135	4842	29,911	415.4	141	6372	167,404	69,350	52,074.2
115	368	5	40	97	9.3	40	25	31	9.6	40	21	23	1	15.5
115	368	10	116	4078	112.9	111	170	1340	64.9	107	199	4760	1050	3167.4
182	615	5	45	214	18.6	45	17	93	11.1	45	14	29	3	38.5
182	615	10	104	800	96.6	101	111	900	131.6	111	185	7117	1100	9047.5
354	1670	5	21	1	9.4	21	8	1	9.2	21	10	23	1	43.5
354	1670	10	93	300	104.2	93	12	431	141.3	93	25	420	50	1270.7
Pit 2														
66	312	5	40	63	1.0	38	6	3	0.4	38	6	7	1	0.6
66	312	10	132	1300	21.0	95	118	550	12.4	95	39	563	450	139.1
90	544	5	54	75	1.7	54	9	1	0.2	54	11	30	1	3.4
90	544	10	154	1136	33.7	140	69	385	16.8	140	46	223	67	88.5
166	1551	5	215	187	12.4	209	28	154	12.5	209	23	46	18	36.6
166	1551	10	510	1988	103.0	475	233	7950	276.8	476	58	2005	1470	2627.1
420	5900	5	628	347	126.8	621	42	456	162.9	621	63	413	275	5391.4
420	5900	10	1463	5721	2688.7	1404	530	2550	1890.2	1391	463	4658	2157	87,659.7

particular for reducing the CPU time required by the solver as the height of the blocks decreased. Addition of clique inequalities based on two-block conflicts at the root node was also found to be beneficial in terms of decreasing the root node gap, reducing the number of branch and bound nodes, and decreasing solution times.

The columns labelled "multiple blocks" in Tables 1 and 2 demonstrate that when generating minimal induced cover inequalities for multiple blocks we need to be careful. No variables are eliminated by such constraints, and generation of these inequalities is computationally demanding. This can lead to an increase in the overall CPU time. All of our strengthening approaches were also applied within the branch and bound tree, but we found that while some further improvements were possible, most of the gains were already obtained by applying these approaches at the root node.

We did not attempt to lift the constraints used in this paper (beyond the lifting used in Algorithm 1). As mentioned earlier, our constraints can all be viewed as valid inequalities for a particular precedence constrained knapsack polytope. Techniques for lifting such inequalities have been developed [16,19]; these techniques could be applied to further strengthen our constraints. But as the computation of the right lifting coefficients is expensive, the effect of lifting on the overall CPU time may be negative.

Finally, future work could include the development of further strong, or even facet defining inequalities for the OPMPSP formulation (1)–(6). One approach is to find inequalities that are facet-defining for the PCK polytope. Results in this direction will be reported in [2]. The development of efficient algorithms for solving the separation problem is another important area for future research.

Acknowledgements

The authors are very grateful to Merab Menabde, Peter Stone, and Mark Zuckerberg (BHP Billiton) for their ongoing support and guidance on a variety of practical mining-related issues and for numerous technical suggestions and insightful feedback that improved the content and exposition of this work. This research is supported by the Australian Research Council Linkage Project LP0348907 and by BHP Billiton Limited.

References

[1] Bley A. Routing and capacity optimization for IP networks. PhD thesis, Technische Universität, Berlin; 2007.
[2] Boland N, Bley A, Fricke C, Froyland G, Sotirov R. Clique-based facets for the precedence constrained knapsack problem. Optimization Online, October 2009.
[3] Bomze I, Budinich M, Pardalos P, Pelillo M. The maximum clique problem. In: Handbook of combinatorial optimization, vol. 4. Dordrecht: Kluwer Academic Press; 1999.
[4] Borndörfer R, Kormos Z. An algorithm for maximum cliques. Unpublished working paper, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1997.
[5] Andrew Boyd E. Polyhedral results for the precedence-constrained knapsack problem. Discrete Applied Mathematics 1993;41:185–201.
[6] Caccetta L, Hill SP. An application of branch and cut to open pit mine scheduling. Journal of Global Optimization 2003;27:349–65.
[7] Fricke C. Applications of Integer programming in open pit mining. PhD thesis, University of Melbourne; 2006.
[8] Garey MR, Johnson DS. Computers and intractability: a guide to the theory of *NP*- completeness. San Francisco: W.H. Freeman and Company; 1979.
[9] Gershon ME. Mine scheduling optimization with mixed integer programming. Mining Engineering 1983;35:351–4.
[10] Gershon ME. An open-pit production scheduler: algorithm and implementation. Mining Engineering 1987;39:793–6.
[11] Hustrulid W, Kuchta M. Open pit mine planning and design, Fundamentals, vol. 1. Rotterdam: A.A. Balkema; 1995.
[12] Johnson DS, Niemi KA. On knapsacks, partitions, and a new dynamic programming technique for trees. Mathematics of Operations Research 1983;8:1–14.
[13] Kolliopolous SG, Steiner G. Partially ordered knapsack and applications to scheduling. Discrete Applied Mathematics 2007;155:889–97.
[14] Lerchs H, Grossmann IF. Optimum design of open-pit mines. Transactions CIM 1965;LXVIII:17–24.
[15] Onur AH, Dowd PA. Open-pit optimization—part 2: production scheduling and inclusion of roadways. Transactions of the Institute of Mining and Metallurgy Section A 1993;102:A105–13.
[16] Park K, Park S. Lifting cover inequalities for the precedence-constrained knapsack problem. Discrete Applied Mathematics 1997;72:219–41.
[17] Martin Smith L. Optimizing inventory stockpiles and mine production: an application of separable and goal programming to phosphate mining using AMPL/CPLEX. CIM Bulletin 1999;92(1030):61–4.
[18] Tolwinski B, Underwood R. A scheduling algorithm for open pit mines. IMA Journal of Mathematics Applied in Business and Industry 1996;7:247–70.
[19] van de Leensel RLMJ, van Hoesel CPM, van de Klundert JJ. Lifting valid inequalities for the precedence constrained knapsack problem. Mathematical Programming Series A 1999;86:161–85.
[20] Wang Q. Long-term open-pit production scheduling through dynamic phase-bench sequencing. Transactions of the Institute of Mining and Metallurgy Section A 1996;105:A99–104.
[21] Jeff Whittle, Four-X user manual. Melbourne: Whittle Programming Pty Ltd; 1998.
[22] Woeginger GJ. On the approximability of average completion time scheduling under precedence constraints. In: Proceedings of the 28 ICALP, 2001. p. 887–897.
[23] Wolsey LA. Integer programming. New York: Wiley-Interscience; 1998.