A strengthened formulation for the open pit mine production scheduling problem

Natashia Boland[*], Christopher Fricke[*†], Gary Froyland[‡§]

June 27, 2006

Abstract

We present a strengthened integer programming formulation for the open pit mine production scheduling problem, where the precedence and production constraints are combined to form 0-1 knapsack inequalities. Addition of corresponding knapsack cover inequalities decreases the computational requirements to obtain the optimal integer solution, in many cases by a significant margin.

Keywords: Integer programming; Mine optimization; Scheduling.

1 Introduction

The design and scheduling of open pit mines is a significant and complex problem in mine planning. The principal aim of a mining operation is to ensure that an ore body is exploited in a way such that the value realized from the mine is maximized. A well-known early contribution to this field was made by Lerchs and Grossmann [8], 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 maximized. 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 criteria 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 maximize 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 (Gershon [5]), parametric methods (Whittle [15]), dynamic programming (Onur and Dowd [11], Wang [14], Tolwinski and Underwood [12]) and integer linear programming (Gershon [4], Smith [10], Caccetta and Hill [2]). 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 [2]. We combine the precedence and production constraints to form constraints that possess the structure of 0-1 knapsack inequalities. Corresponding knapsack cover inequalities

[*]Department of Mathematics and Statistics, University of Melbourne, Victoria, Australia, 3010

[†]Corresponding author. Email address: fricke@ms.unimelb.edu.au

[‡]School of Mathematics, University of New South Wales, New South Wales, Australia, 2052

[§]The authors are grateful to Merab Menabde (BHP Billiton Technology) for assistance in providing data, and for discussion regarding creation of the minimal precedence sets.

are then generated, which, in the case of single units of material, identify the earliest time period in which each unit of material can be extracted from the pit, as shown in Section 3. Addition of these constraints to the standard integer programming formulation eliminates a number of decision variables from the model prior to optimization. Consequently, a tighter root node relaxation is obtained, and a dramatic decrease is seen in the computation time required to obtain the optimal integer solution for certain pits.

Generation of knapsack cover inequalities in the case of multiple units of material is considered in Section 4. We test the effectiveness of the inequalities, first by adding constraints for all covers of size two, a priori, at the root node (Section 4.3). To see if these inequalities can further tighten the LP relaxation, we implement a cutting plane approach, with straightforward exact separation, and demonstrate computationally that further improvements in gap are possible (Section 4.4).

2 Integer programming formulation

The most commonly used representation of an ore body for the purposes of mathematical modelling is as a *block model*, described by Hustrulid and Kuchta [6]. 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. These attributes of the block model are used as input parameters for the integer linear programming formulation for the mine production scheduling problem.

In this paper, we consider the following generalized version of the integer program presented by Caccetta and Hill [2]. 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. E_i is the minimal set of blocks that must be removed for block i to be removed, including block i. Frequently E_i has the three dimensional shape of an inverted cone. Note that for computer memory management, for each block i it is sufficient to consider a smaller set of predecessor blocks S_i. Depending on the required angle for safe wall slopes, S_i could be as small as the set of immediate predecessors of block i; see Figure 1.

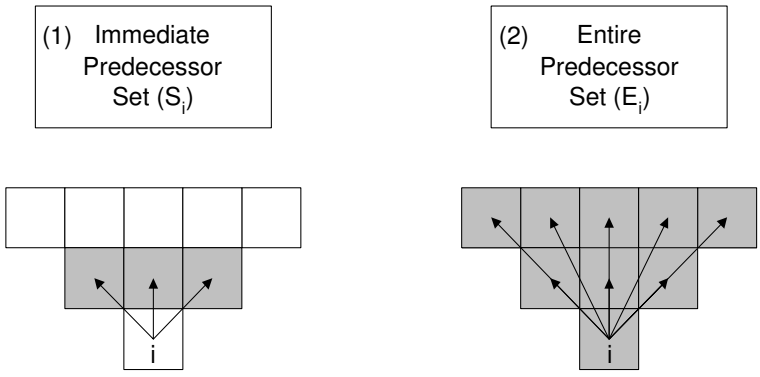

Figure 1: Illustration of (1) The immediate predecessor set (2) The entire predecessor set.

The decision variables of the model take the form

$$x_i^t = \begin{cases} 1, & \text{if block } i \text{ is mined in periods } 1,\ldots,t \\ 0, & \text{otherwise} \end{cases} \qquad i = 1,\ldots,N, \;\; t = 1,\ldots,T.$$

For ease of notation throughout this paper, we introduce a set of dummy decision variables $x_i^0, \;\; i = 1,\ldots,N$, which are assigned the value 0. The generalized integer program for the mine production scheduling problem is then given by (1)-(6):

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

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

$$x_i^{t-1} \leq x_i^t \quad t = 1,\ldots,T, \; i = 1,\ldots,N \tag{3}$$

$$x_i^t \leq x_j^t \quad t = 1,\ldots,T, \; i = 1,\ldots,N, \; \forall \, j \in S_i \tag{4}$$

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

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

The objective (1) is to maximize 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.

Caccetta and Hill [2] developed a customized Branch-and-Cut method to solve their formulation of the mine production scheduling problem, where two attributes were considered. The algorithm was implemented in C++ and contained about 17,000 lines of code [2]. We have implemented the formulation (1)-(6) in CPLEX version 8.0 by ILOG Inc., a commercially available mixed integer optimization software package. In Sections 3 and 4, we motivate and develop valid inequalities that, when added to the formulation, reduce 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 eliminated prior to optimization. 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. To achieve this, we define the total set of predecessors for all blocks $\mathcal{S} = \{(i,j) : i \in \{1,\ldots,N\}, j \in S_i\}$. 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, intuitively we have that

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

Although this relationship is intuitively clear, it is possible to deduce it analytically from the original constraints. The cumulative available production capacity for a particular attribute $a \in A$, up to and including time period t, is given by the *aggregated production constraints*, where (2) is summed over times $1, \ldots, t$:

$$\sum_{s=1}^{t} \sum_{i=1}^{N} q_i^a \ (x_i^s - x_i^{s-1}) \ \ \le \ \ \sum_{s=1}^{t} u_a^s. \tag{8}$$

These can be reduced as follows. For given $t \in \{1, \ldots, T\}$ and $a \in A$, (8) is equivalent to

$$\sum_{i=1}^{N} q_i^a \left(\sum_{s=1}^{t} (x_i^s - x_i^{s-1}) \right) \ \le \ \sum_{s=1}^{t} u_a^s \tag{9}$$

$$\Rightarrow \quad \sum_{i=1}^{N} q_i^a \ (x_i^t - x_i^0) \ \le \ \sum_{s=1}^{t} u_a^s \tag{10}$$

$$\Rightarrow \quad \sum_{i=1}^{N} q_i^a \ x_i^t \ \le \ \sum_{s=1}^{t} u_a^s \quad \text{since } x_i^0 = 0,\ i = 1, \ldots, N. \tag{11}$$

Now for any $i \in \{1, \ldots, N\},\ E_i \subseteq \{1, \ldots, N\}$, so

$$\sum_{j \in E_i} q_j^a \ x_j^t \ \ \le \ \ \sum_{s=1}^{t} u_a^s \ \ \Rightarrow \ \ \left(\sum_{j \in E_i} q_j^a \right) x_i^t \ \le \ \sum_{s=1}^{t} u_a^s \quad \text{since } x_i^t \le x_j^t \text{ for all } j \in E_i. \tag{12}$$

Thus, since x is binary,

$$x_i^t \le \lfloor \sum_{s=1}^{t} u_a^s \ / \ \sum_{j \in E_i} q_j^a \rfloor,\ t = 1, \ldots, T,\ i = 1, \ldots, N,\ a \in A, \tag{13}$$

which is equivalent to our intuitive conclusion.

Inclusion of (13) in the model (1)-(6) allows a number of variables to be fixed to 0 prior to optimization, resulting in a reduction in the number of decision variables in the model. The formulation (1)-(6) with and without this variable fixing has been tested 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 are a greater number of precedence constraints per block in the formulation. All formulations were solved using CPLEX v8.0 via ILOG Concert Technology on a Professional Station XP1000

with 2GB RAM. The default settings for mixed integer optimization in CPLEX were used, and all integer programs were solved to an optimality gap of 1%. The base case for all models has a mine life of 15 years. Hence the cases tested in Table 1 correspond to five time periods of three years each, and ten time periods of eighteen 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. The computational advantage of the tighter formulation is shown by the results in Table 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). Clearly, constraints (13) are not automatically detected and applied in the preprocessing phase by CPLEX.

Data Set	No. Time Pers	Standard Formulation				Strengthened Formulation			
		No. Dec Vars	No. B & B Nodes	Root Node Gap (%)	CPU Time (secs)	No. Dec Vars	No. B & B Nodes	Root Node Gap (%)	CPU Time (secs)
Pit 1									
N = 67	5	335	2700	9.5	24.6	289	900	3.3	9.4
$\|\mathcal{E}\|$ = 190	10	670	2132700$^+$	11.3	40300$^+$	567	3768100$^+$	4.7	50000$^+$
N = 115	5	575	3840	6.0	103.1	535	610	2.3	22.6
$\|\mathcal{E}\|$ = 368	10	1150	1935300$^+$	7.0	100000$^+$	1044	456000	3.2	17479.6
N = 182	5	910	1100	3.9	101.3	870	100	1.8	28.5
$\|\mathcal{E}\|$ = 615	10	1820	143400	4.4	15843.7	1719	27210	2.3	2469.5
N = 354	5	1770	500	1.5	159.6	1749	290	1.1	189.9
$\|\mathcal{E}\|$ = 1670	10	3540	28900	1.7	13745.2	3450	29970	1.3	12753.0
Pit 2									
N = 66	5	330	800	5.2	15.3	292	60	3.4	2.3
$\|\mathcal{E}\|$ = 312	10	660	15700	8.9	546.1	566	13030	5.0	251.9
N = 90	5	450	4700	10.3	75.0	396	300	4.8	9.0
$\|\mathcal{E}\|$ = 544	10	900	1016200	11.9	64183.3	760	172700	5.4	5410.8
N = 166	5	830	3300	39.3	434.7	621	2290	5.5	103.7
$\|\mathcal{E}\|$ = 1551	10	1660	1520500$^+$	45.2	360000$^+$	1185	164320	7.2	13035.4
N = 420	5	2100	19800	68.5	10232.8	1481	1200	9.7	652.6
$\|\mathcal{E}\|$ = 5900	10	4200	119000$^+$	70.2	360000$^+$	2816	101500	9.3	60463.3

Table 1: Computational Results for Single Block Case

Note that a $^+$ in the table indicates an instance where the formulation was unable to find the optimal integer solution, either before the branch-and-bound tree reached an imposed memory limit of 1GB, or the CPU time reached 100 hours. 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. In each of the sixteen instances tested, the strengthened formulation gave a significantly tighter root node relaxation than the standard formulation. In the eleven instances where both formulations found the

optimal integer solution and the strengthened formulation computationally outperformed the standard formulation, the strengthened formulation was an average of 4.8 times faster and required an average of 6.4 times less branch-and-bound nodes.

4 Strengthening the formulation: multiple block case

4.1 Valid knapsack inequalities and covers

We define the union of the entire precedence sets for the blocks in the set X 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 $1, \ldots, t$.

However, when considering a set of blocks, it is possible for the union of the entire precedence sets of the blocks to overlap. To ensure that each block is only counted in the aggregated production constraints once, we let $\mathcal{P}_X = \{P_i \ : \ i \in X\}$ define a partition of $E(X)$ such that $\cup_{i \in X} P_i = E(X)$, $P_i \subseteq E_i$ for all $i \in X$, $P_i \cap P_j = \emptyset$ for all $i \neq j$. We call such a partition *precedence aligned*. An example of a set of blocks X, the union of the entire precedence sets $E(X)$, and a corresponding precedence aligned partition in a two-dimensional case are given in Figure 2, where we use a slope angle of 45^o, so that the minimal immediate predecessor set S_i for each block i contains the block immediately above it, and the block either side of that block.

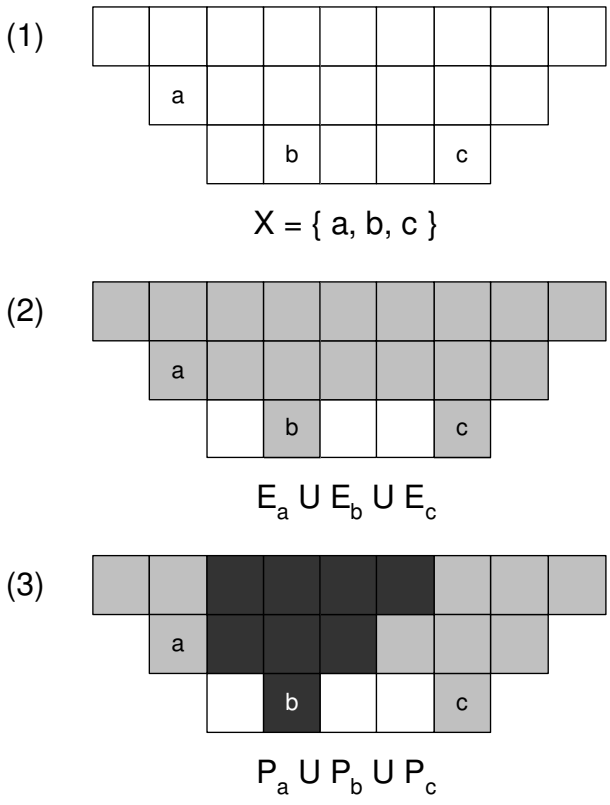

Figure 2: Illustration of (1) A possible cover set X (2) The union of the entire precedence sets $E(X)$ (3) A precedence aligned partition $\mathcal{P}_X$ of $E(X)$.

Theorem 1. *For given attribute $a \in A$, period $t \in \{1, \ldots, T\}$, set $X \subseteq \{1, \ldots, N\}$ and precedence aligned partition $\mathcal{P}_X$*

$$\sum_{i \in X} \left(\sum_{j \in P_i} q_j^a \right) x_i^t \leq \sum_{s=1}^{t} u_a^s \tag{14}$$

is a knapsack inequality valid for the mine scheduling polytope.

Proof: By (11), and since $E(X) \subseteq \{1, \ldots, N\}$, the following inequality is valid: $\sum_{j \in E(X)} q_j^a \, x_j^t \leq \sum_{s=1}^{t} u_a^s$. Hence, and since $\mathcal{P}_X$ partitions $E(X)$, it must be that the following inequality is also valid: $\sum_{i \in X} \left(\sum_{j \in P_i} q_j^a x_j^t \right) \leq \sum_{s=1}^{t} u_a^s$. Furthermore, for all $i \in X$, $P_i \subseteq E_i$ and $x_i^t \leq x_j^t$ for all $j \in E_i$; the result follows. □

In what follows, we will focus on knapsack covers for (14) that include variables x_i^t for all $i \in X$. We will call such a cover a *complete cover*. We now show that for complete covers the choice of partition becomes irrelevant. Suppose X is a complete cover for (14), i.e. suppose that $\sum_{i \in X} (\sum_{j \in P_i} q_j^a) > \sum_{s=1}^{t} u_a^s$. But $\{P_i : i \in X\}$ partitions $E(X)$, so $\sum_{i \in X} (\sum_{j \in P_i} q_j^a) = \sum_{i \in E(X)} q_i^a$. Thus we may equivalently define X to be a *complete cover for attribute a and period t* if

$$\sum_{i \in E(X)} q_i^a > \sum_{s=1}^{t} u_a^s. \tag{15}$$

Clearly (15) does not depend at all on the choice of precedence aligned partition.

For X a complete cover for some attribute a and period t, the corresponding cover inequality

$$\sum_{i \in X} x_i^t \leq |X| - 1 \tag{16}$$

is valid for the mine scheduling polytope; as the above discussion shows, it is simply a cover inequality for a valid knapsack inequality. This significantly simplifies the problem of discovering covers. Note that (13) from Section 3 is a special case of (16), since it is an inequality for complete covers X with $|X| = 1$.

In closing this section, we note that it is not hard to show that we can always derive a knapsack constraint and corresponding complete cover from any knapsack constraint of the form of (14). (If $X' \subset X$ gives rise to a cover for (14), choose any partition $\{Q_i : i \in X'\}$ of $E(X') \setminus \cup_{i \in X'} P_i$ with $Q_i \subseteq E_i$, and set $\mathcal{P}_{X'} = \{P_i \cup Q_i : i \in X'\}$. Then $\mathcal{P}_{X'}$ is a precedence aligned partition of $E(X')$, and X' is a cover for (14) with X' and $\mathcal{P}_{X'}$ in place of X and $\mathcal{P}_X$ respectively.)

4.2 Connections to the precedence constrained knapsack polytope

An alternative derivation of the valid inequalities (16) stems from the following observation. Consider a time period $t \in \{1, \ldots, T\}$ and an attribute $a \in A$. The aggregated production constraint for time period t, given by (11), 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 $\text{PCK}_{t,\,a}$. Johnson and Niemi [7] showed that the PCK problem is NP-complete. The polyhedral structure of the problem was first investigated by Boyd [1], who generalised results corresponding to finding cover inequalities for the standard knapsack polytope. In particular, Boyd derived conditions under

which inequalities of the form (16) are facet-defining for the convex hull of the feasible solutions to $\text{PCK}_{t,a}$ restricted to the variables in $E(X)$. Further investigation of the PCK polytope is presented by both Park and Park [9] and van de Leensel et. al. [13], where lifting orders and general sequential lifting procedures are derived to lift valid inequalities from lower dimensional polytopes 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 $\text{PCK}_{t,\,a}$ polytopes for all $t \in \{1, \ldots, T\}$ and $a \in A$. Hence inequalities that are facet-defining for $\text{PCK}_{t,\,a}$ will not necessarily be facet-defining for the mine production scheduling problem.

4.3 The two block case

To test the effectiveness of the complete cover inequalities we consider complete covers X with $|X| = 2$. For all pairs of distinct blocks $i, j \in \{1, \ldots, N\}$, we find the latest time period $t(i,j) \in \{1, \ldots, T\}$ such that both i and j cannot be mined by period $t(i,j)$ without violating the cumulative capacity constraint for some attribute. In other words, $t(i,j)$ is the maximizer of t over all $t = 1, \ldots, T$ such that $\sum_{k \in E_i \cup E_j} q_k^a > \sum_{s=1}^{t} u_a^s$, for some $a \in A$, if one exists. For each pair i, j for which $t(i,j)$ is found, provided that neither i nor j can be removed as single block complete covers, i.e. eliminated using the results of Section 3, a constraint of the form

$$x_i^{t(i,j)} \;+\; x_j^{t(i,j)} \;\leq\; 1 \tag{17}$$

is added to the formulation (1)-(6), (13) prior to optimization.

Rather than directly add all such block pair inequalities to the formulation, we instead add them to a pool of cuts in CPLEX known as UserCuts. Cuts from this pool are then added to the formulation by CPLEX during the optimization if they are violated by the incumbent solution. Adding the cuts in this way ensures that the solution of the linear programming relaxation is not slowed down due to the inclusion of a large number of redundant inequalities. Results following the addition of such two block inequalities to the strengthened formulation are presented in Table 2. Note that the IP CPU time column does not include determination of the constraints (17), while the overall CPU time column does.

In all instances tested, the root node gaps are less than or equal to those of the single block case reported in Table 1. In ten instances the CPU times were improved further (an average of 2.0 times faster), and in nine instances the number of branch-and-bound nodes was further reduced (an average of 3.0 times less nodes). Note that we also considered adding only two block inequalities that were facet-defining for $\text{PCK}_{t,a}$ according to Boyd's condition [1], as discussed in Section 4.2. However, in all cases adding all the two block inequalities as in Table 2 gave significantly better computational results.

4.4 The multiple block case and a cutting plane approach

Clearly, finding and adding all complete cover inequalities a priori is impractical in general. Thus to test the effectiveness of such inequalities for complete covers of size greater than two, we develop a cutting plane approach, using a straightforward integer program to solve the separation problem exactly, and applying constraints at the root node only.

The separation problem for the complete cover inequality can be formulated as follows. Given fractional solution $\hat{x}^t$ to the root node relaxation, we seek a time period t, an attribute a, and a complete cover X for t and a, such that (16) is violated by $\hat{x}^t$, i.e. such that

$$\sum_{i \in X} \hat{x}_i^t \;>\; |X| - 1. \tag{18}$$

Data Set	No. Time Pers	No. User Cuts Applied	No. User Cuts Added	No. B & B Nodes	Root Node Gap (%)	IP CPU Time (secs)	Overall CPU Time (secs)
Pit 1							
N = 67	5	14	86	940	3.3	7.7	7.9
$\|\mathcal{E}\| = 190$	10	59	238	566600	4.3	9249.5	9249.8
N = 115	5	25	187	500	1.9	21.4	22.2
$\|\mathcal{E}\| = 368$	10	41	420	138700	2.8	5194.5	5195.4
N = 182	5	24	198	200	1.7	35.5	38.2
$\|\mathcal{E}\| = 615$	10	74	886	185740	2.2	16933.4	16936.5
N = 354	5	5	80	270	1.1	163.1	181.0
$\|\mathcal{E}\| = 1670$	10	24	620	36950	1.2	13331.4	13350.3
Pit 2							
N = 66	5	12	93	80	2.4	2.6	2.8
$\|\mathcal{E}\| = 312$	10	23	181	3800	3.9	87.5	87.8
N = 90	5	6	81	100	3.3	7.5	8.0
$\|\mathcal{E}\| = 544$	10	33	337	23100	3.8	818.9	819.5
N = 166	5	24	326	400	4.7	42.9	45.9
$\|\mathcal{E}\| = 1551$	10	41	1039	24900	5.7	4415.5	4419.2
N = 420	5	44	667	3000	8.0	1173.1	1221.5
$\|\mathcal{E}\| = 5900$	10	67	2346	73820	7.9	56614.5	56668.9

Table 2: Computational Results for Addition of Two Block Inequalities

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 a linear integer programming model of the separation problem, for each time period t, we let

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

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

We implemented a cutting plane algorithm to be applied at the root node of the branch-and-bound tree. At each iteration of the algorithm, the complete cover separation problem is solved by solving an integer program formulated using the binary variables z_i and w_i defined above. If a violated complete cover inequality is found, the complete cover is reduced, if necessary, to ensure that it is minimal (in the sense that $\sum_{j \in E(X \setminus \{i\})} q_j^a \leq \sum_{s=1}^{t} u_a^s$ for all $i \in X$), and the corresponding constraint is added to the formulation. Following completion of the cutting plane algorithm, the inequalities found are added to the pool of UserCuts in CPLEX, and the incumbent integer program is solved to determine the optimal integer solution. This cutting plane algorithm has been implemented for the data sets tested in Section 3, with $A = \{$rock tonnage, ore tonnage$\}$. The number of separation instances solved, along with the number of cover inequalities added to the formulation, is reported in Table 3. We also experimented with running the cutting plane algorithm for one of the attributes only. In general the results were equivalent or worse than those presented in Table 3.

Data Set	No. Time Pers	No. User Cuts Applied	No. Covers Added	No. Sep Probs Solved	Cut Plane CPU Time (secs)	No. B & B Nodes	Root Node Gap (%)	Final IP CPU Time (secs)
Pit 1								
N = 67	5	52	56	90	75.8	510	2.2	8.7
$\|\mathcal{E}\|$ = 190	10	94	110	360	168.0	871950	4.0	18920.1
N = 115	5	39	46	90	239.4	200	1.9	15.6
$\|\mathcal{E}\|$ = 368	10	99	111	560	806.5	137900	2.5	7796.0
N = 182	5	54	59	110	830.1	800	1.5	74.2
$\|\mathcal{E}\|$ = 615	10	123	140	640	2620.0	66950	2.0	6807.9
N = 354	5	55	62	270	2351.4	400	1.1	228.5
$\|\mathcal{E}\|$ = 1670	10	85	95	400	4549.0	2600	1.2	1784.4
Pit 2								
N = 66	5	16	22	60	8.9	50	2.3	3.1
$\|\mathcal{E}\|$ = 312	10	37	44	140	23.3	900	3.1	46.8
N = 90	5	27	30	140	63.6	120	3.0	8.1
$\|\mathcal{E}\|$ = 544	10	62	69	240	116.2	7050	2.6	445.0
N = 166	5	47	54	240	1306.6	360	2.5	67.8
$\|\mathcal{E}\|$ = 1551	10	85	96	380	2995.0	64400	2.9	12320.9
N = 420	5	114	117	520	110932.7	700	5.3	1016.0
$\|\mathcal{E}\|$ = 5900	10	240	253	1660	389083.9	67000	5.4	93665.7

Table 3: Computational Results for Exact Separation, Inequalities Added as User Cuts

The greatest benefit from the cutting plane algorithm reported in Table 3 is that in all cases, the root node gap has decreased when compared to the gaps reported in Tables 1 and 2. In nine of the sixteen instances tested, the number of branch-and-bound nodes required to find the optimal integer solution after implementation of the cutting plane algorithm is the lowest of the four approaches tested in this paper (an average of 2.5 times less nodes). In four instances the CPU time required to solve the final integer program is the fastest after the implementation of the cutting plane algorithm (an average of 2.5 times faster). This indicates that there is potential for further computational improvement if effective inequalities can be derived efficiently.

Clearly the computation time for finding violated complete cover inequalities by solving an integer program is prohibitive. 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.

5 Conclusions and future work

The single block complete cover inequalities derived in Section 3 were found to be highly beneficial in the majority of cases tested, in particular for reducing the CPU time required by the solver as the height of the blocks decreased. A priori addition of two block complete cover inequalities was also found to be beneficial.

As seen in Section 4.4, when generating complete cover inequalities in the case of multiple blocks we need to be careful. No variables are eliminated by such constraints, and generation of these inequalities can be inefficient. This can lead to an increase in the overall CPU time. Hence the development of efficient algorithms for solving the separation problem is an important area for future research, as already mentioned. Nevertheless, the addition of such inequalities can lead to computational benefit in solving the integer program.

Other work for the future includes applying complete cover inequalities in a branch-and-cut framework. Particularly on branches for which variables have been set to one, thus reducing available production capacity, local single block elimination is likely to be particularly effective.

The constraints used in this paper were all unlifted, but can all be viewed as valid inequalities for a particular precedence constrained knapsack (PCK) polytope. As mentioned earlier, techniques for lifting such inequalities have been developed (Park and Park [9] and van de Leensel et. al. [13]); these could be applied to further strengthen the inequalities added.

Finally, future work would seek valid inequalities that are facet-defining, or at least strong, for the convex hull of the formulation. Inequalities that are facet-defining for the PCK polytope may be helpful. This approach could be particularly effective if an efficient scheme for deriving these inequalities is developed. This work is in progress.

References

[1] E. Andrew Boyd, Polyhedral results for the precedence-constrained knapsack problem, Discrete Applied Mathematics 41 (1993) 185-201.

[2] Louis Caccetta and Stephen P. Hill, An application of branch and cut to open pit mine scheduling, Journal of Global Optimization 27 (2003) 349-365.

[3] K.M.J. De Bontridder, B.V. Halldórsson, M.M Halldórsson, C.A.J. Hurkens, J.K Lenstra, R. Ravi and L. Stougie, Approximation algorithms for the test cover problem, Mathematical Programming Series B 98 (2003) 477-491.

[4] M. E. Gershon, Mine scheduling optimization with mixed integer programming, Mining Engineering 35 (1983) 351-354.

[5] M. E. Gershon, An open-pit production scheduler: algorithm and implementation, Mining Engineering 39 (1987) 793-796.

[6] William Hustrulid and Mark Kuchta, Open Pit Mine Planning and Design, Volume 1: Fundamentals, A.A. Balkerma, Rotterdam, 1995.

[7] D. S. Johnson and K. A. Niemi, On knapsacks, partitions, and a new dynamic programming technique for trees, Mathematics of Opertions Research 8 (1983) 1-14.

[8] Helmut Lerchs and Ingo F. Grossmann, Optimum design of open-pit mines, Transactions CIM, Vol. LXVIII (1965) 17-24.

[9] Kyungchul Park and Sungsoo Park, Lifting cover inequalities for the precedence-constrained knapsack problem, Discrete Applied Mathematics 72 (1997) 219-241.

[10] Martin L. Smith, Optimizing inventory stockpiles and mine production: An application of separable and goal programming to phosphate mining using AMPL/CPLEX, CIM Bulletin 92:1030 (1999) 61-64.

[11] A. H. Onur and P. A. Dowd, Open-pit optimization - part 2: production scheduling and inclusion of roadways, Transactions of the Institute of Mining and Metallurgy Section A 102 (1993) A105-A113.

[12] Boleslaw Tolwinski and Robert Underwood, A scheduling algorithm for open pit mines, IMA Journal of Mathematics Applied in Business and Industry 7 (1996) 247-270.

[13] R. L. M. J. van de Leensel, C. P. M. van Hoesel and J. J. van de Klundert, Lifting valid inequalities for the precedence constrained knapsack problem, Mathematical Programming Series A 86 (1999) 161-185.

[14] Qing Wang, Long-term open-pit production scheduling through dynamic phase-bench sequencing, Transactions of the Institute of Mining and Metallurgy Section A 105 (1996) A99-A104.

[15] Jeff Whittle, Four-X User Manual, Whittle Programming Pty Ltd, Melbourne, 1998.

[16] Laurence A. Wolsey, Integer Programming, Wiley-Interscience, New York, 1998.