DIRECT OPTIMIZATION OF AN OPEN CUT SCHEDULING POLICY

Marcos Goycoolea, Associate Professor, School of Business, Universidad Adolfo Ibañez, Santiago, Chile, 7941169 (corresponding author).

Eduardo Moreno, Associate Professor, Faculty of Engineering and Sciences, Universidad Adolfo Ibañez, Santiago, Chile, 7941169.

Orlando Rivera, Research Associate, Center for Mining Innovation, Universidad Adolfo IbañezSantiago, Chile, 7941169.

ABSTRACT

Given a block-model of an open cut mine, a production scheduling policy defines which blocks should be extracted, when they should be extracted, and what should be done with them once extracted (i.e., sent to a mill, waste-dump, stockpile, etc.). The conventional three-step heuristic for constructing such a scheduling is as follows: First, compute an ultimate-pit. Second, subdivide the ultimate-pit into phases. Third, schedule the blocks in each bench-phase, taking into account the mining, milling and market/refining capacities. Though each of these steps is in itself an optimization problem, the three steps, when put together, constitute a piece-meal approach to the full problem. Recent developments make it possible to implement a direct optimization methodology using integer programming (IP) that can tackle real-sized problems. In this article we first compare such a direct approach to a commercial implementation of the traditional methodology (Gemcom Whittle) and find that the direct approach yields solutions with significantly higher value. In an attempt to explain this difference, we modify the IP approach so as to generate solutions more similar to those obtained by the traditional approach. We find that this modified approach still outperforms the commercial solver. We conclude that the direct approach attains higher value primarily because the commercial solver's heuristics are not effective in the problems considered. In fact, we find that the traditional methodology, if executed exactly, yields near-optimal solutions on all our benchmark problems. We do not consider blending, stockpiles, or operational spatial constraints (e.g., only "best-case" solutions).

INTRODUCTION

Before carrying out an open pit mining operation, analysts must first prepare what is known as a strategic mine plan. Given a mining reserve, this consists of a tentative production schedule, outlining which part of the reserve will be extracted, when and how it will be extracted, and the capital investments that will be required to do so. Such a plan, in practice, serves as a feasibility report with which investors can decide whether or not to pursue the mining endeavour, and as an approximate life-of-mine plan, detailing the cash flows over time from the moment the project is initiated until reconciliation.

In order to produce a strategic mine plan, analysts must first count with a block model. This is a discretized three-dimensional representation of the reserve that is made up of equal sized units known as blocks. In a block model, each block has a number of geologic and economic attributes, including location, tonnage, mineral grades, hardness, wall-slope requirements, and others (Hustrulid and Kuchta, 2006).

A strategic mine plan, or production schedule, consists in specifying which blocks in the reserve will be extracted, and, for those blocks that are to be extracted, the time of extraction and the destination (e.g., mill, waste-dump, leech pad, etc.). This is done in such a way as to maximize NPV and meet capacity constraints. Capacity constraints are primarily of two types: mining constraints and processing constraints. Mining constraints impose a limit on the total tonnage of blocks that can be extracted from the reserve in a time period. Processing constraints impose a limit on the total tonnage of blocks that can be sent to each destination in a time period.

The traditional mine planning methodology

There are many different ways in which a production schedule can be produced. We begin by introducing some notation, and then describe the key steps of the traditional methodology.

Consider a mining reserve with blocks $\mathcal{B}$ and destinations $\mathfrak{D}$ (e.g., mill, leech pad, waste-dump, etc.). Let value p_{bd} represent the net profit obtained from sending a block $b \in \mathcal{B}$ to destination $d \in \mathfrak{D}$. These values are calculated on the assumption that each block b "HAS been uncovered and that it WILL be mined" (Whittle, 1990). That is, these profits do not consider any fixed mining costs or variable costs associated to other blocks and are independent of the order in which blocks are mined and processed. Let $\mathcal{A}$ represent the set of all block-pairs (a, b) such that block a must be extracted before b in order to comply with pit-slope requirements. For each block $b \in \mathcal{B}$, let q_b be its tonnage. For each time period t let M_t be the mining capacity (in tons) and let U_t^d be the processing capacity for destination d (tons).

The three main steps of the traditional mine planning are:

1. Computing an ultimate pit or final contour. In this step the region of the reserve where the mining operations are to ensue is limited to a set of blocks P satisfying the required wall-slope requirements (i.e., a pit). Let u be any vector in $\mathbb{R}^{\mathfrak{B}}$. Define problem:

$$\text{UPL(u):}\quad \begin{aligned} &\max \sum_{b\in\mathcal{B}} u_b x_b \\ &\text{st. } x_b \le x_a \quad \forall\,(a,b) \in \mathcal{A} \\ &x_b \in \{0,1\}\ \forall b \in \mathcal{B} \end{aligned}$$

The ultimate pit limit or final contour delimits the sub-region of the mine in which extraction will take place. It is determined by solving UPL(p), where vector p is defined so that $p_b = \max\{p_{bd} : d \in \mathfrak{D}\}$ for each $b \in \mathcal{B}$. In problem UPL(p), variable x_b takes value one if and only if block b is included in the ultimate pit limit. That is, the ultimate pit limit, or final contour, is defined as the set of all blocks in set $P = \{b \in \mathcal{B} : x_b = 1\}$. This problem can be solved using the algorithm of Lerchs and Grossman (1965), though recent evidence suggests the Pseudoflow algorithm can be significantly faster (Hochbaum and Chen, 2000; Chandran and Hochbaum 2009).

2. Sequencing. In this step the blocks in the ultimate pit are subdivided into a sorted family of sets $S^1, S^2, \ldots, S^n$ (typically called phases) that prescribe an order of block extraction. The intuition is that the blocks in S_1 are the most immediately profitable, and should be extracted first; blocks in S_2 second, and so on. To compute phases it is customary to define a set of vectors $(p^1, p^2, \ldots, p)$ such that $p^1 \le p^2 \le p^3 \le \cdots \le p$, where p is the block value vector used to compute the ultimate pit limit, and where the inequalities are defined component-wise (i.e., $p_b^i \le p_b^{i+1}$ for all i and b). For each vector p^i, let x^i represent the optimal solution of $\text{UPL}(p^i)$, and let $P^i = \{b \in \mathcal{B} : x_b^i = 1\}$. It is well known that the sets P^i will be nested (Lerchs and Grossman, 1965). That is, $P^1 \subseteq P^2 \subseteq \cdots P$. Using these nested pits, the phases are defined using these pits, as $S^1 = P^{j_1}$ and $S^i = P^{j_i} - S^{i-1}$, where $j_i < j_{i+1}$ for $i > 1$.

3. Production Scheduling. In this step each block in the ultimate pit is assigned a time of extraction and a destination (e.g., mill, waste-dump, leach pad, etc.). Rather than scheduling blocks individually, blocks are scheduled in sets corresponding to the intersections of phases and benches. Formally, for each j, let $\mathcal{B}(j)$ represent the set of all blocks in bench j of the reserve (numbered from top to bottom), and consider a set of ordered phases $S^1, S^2, \ldots, S^n$. For each phase S^i, let $S_j^i = S^i \cap B(j)$. If a set S_j^i is non-empty, it is called an increment or a bench-phase. Increments are scheduled for extraction in such a way that if two increments are in a same phase, then the one on top is scheduled first for extraction, and if they are in different phases, the one in the lowest-indexed phase is scheduled first. Formally, let $\mathcal{I} = \{I_1, I_2, \ldots\}$ represent the set of all increments defined from a set of phases, sorted in extraction order. If $I_u = S_j^i$ and $I_v = S_l^k$ correspond to two distinct increments, then $u < v$ if and only if $i<k$, or, if $i = k$ and $j < l$. See Figure 1 for a depiction of phases and increments in a hypothetical two-dimensional mine.

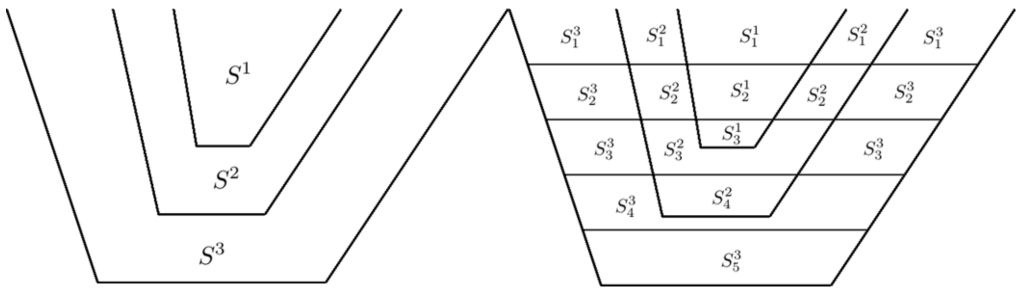

Figure 1 – Three phases (left) and its corresponding increments (right) of a two-dimensional mine with 5 benches.

Given a fixed set of increments corresponding to an open pit mine, it is possible to define many different production schedules. This is because there are many different ways of assigning destinations to the blocks in each increment. One way of obtaining a production schedule is by using Kenneth Lane's well-known "cut-off grade optimization" algorithm (Lane, 1988). This is an iterative dynamic programming algorithm that, under the assumptions that there is a single increment, and that there are only two destinations, guarantees to find an optimal production schedule. We are not aware of any study showing that this algorithm can be extended to multiple increments and destinations in an optimal manner. Presumably because of this, important mine planning software packages include unknown and proprietary heuristics (see for example, Whittle and Wharton, 1995).

Three remarks concerning the traditional mine planning methodology

First, the sequencing method described above is likely to produce phases with undesirable shapes and sizes. This problem can sometimes be overcome by changing the way vectors $(p^1, p^2, \ldots, p)$ are determined, or by modifying the resulting phases with a post-processing algorithm (Wharton, 1997; Khalokakaie, 1999). For example, Gemcom Whittle provides an algorithm known as "Width" for this purpose. Unfortunately, it is known that these techniques can significantly decrease the objective function value of the final solution (Wharton, 1997). Because of this, the solution obtained by Steps 1 through 3 choosing $S^1 = P^1$ and $S^i = P^{i+1} - P^i$ for $i > 1$ is typically referred to as a "best-case" solution, and is most often used as an upper-bound solution that is later modified to obtain a feasible one.

Second, the requirement that increments should be scheduled for extraction in sequential order (by phase and then by bench) is not strict. For example, Gemcom Whittle's Milawa algorithm (Wharton, 2000), allows different phases to be simultaneously worked on. Though in practice this tends to decrease the objective function value of a solution, it is helpful to meet different types of operational constraints.

Third, it is not clear how to extend the methodology described above so as to incorporate blending requirements, stockpiles, multi-mine projects, and other modelling issues. Different software vendors have adopted different approaches.

A direct optimization (integer programming) methodology

To our knowledge, the first effort to formulate a mathematical programming model for solving the strategic open-pit mine planning problem dates back to the work of Johnson (1968). An integer-programming version of the problem described in the introduction can be formulated as follows (Bienstock and Zuckerberg, 2010):

$$\max \sum_{b\in\mathcal{B}} \sum_{d\in\mathcal{D}} \sum_{t\in T} \frac{1}{(1+\alpha)^t} p_{b,d,t} y_{b,d,t} \qquad (1)$$

$$\text{s.t.}$$

$$x_{b,t} = \sum_{d\in\mathcal{D}} y_{b,d,t} \qquad \text{for all } b\in\mathcal{B},\ t\in T \qquad (2)$$

$$\sum_{t=1}^{\tau} x_{b,t} \le \sum_{t=1}^{\tau} x_{a,t} \qquad \text{for all } (a,b)\in\mathcal{A},\ \tau\in T \qquad (3)$$

$$\sum_{b\in\mathcal{B}} q_b x_{b,t} \le M_t \qquad \text{for all } t\in T \qquad (4)$$

$$\sum_{b\in\mathcal{B}} q_b y_{b,d,t} \le U_t^d \qquad \text{for all } d\in\mathcal{D},\ t\in T \qquad (5)$$

$$x_{b,t}, y_{b,d,t} \in \{0,1\} \qquad \text{for all } b\in\mathcal{B}, d\in\mathcal{D}, t\in T \qquad (6)$$

In this formulation variable $x_{b,t}$ is one if and only if block *b* is extracted in time period *t*, and variable $y_{b,d,t}$ is one if and only if block *b* is extracted and sent to destination *d* in time period *t.* The objective function (1) consists in maximizing profits, constraints (2) impose that each block should be sent to exactly one destination if it's extracted, constraints (3) impose the precedence relationships between blocks, constraints (4) impose the mining capacity constraint for each time period, constraints (5) impose the processing capacity constraints for each period and each destination, and constraints (6) impose integrality of the solution. We henceforth refer to this formulation as DIRECT-IP. Note that DIRECT-IP solves a problem that is slightly different than that solved by the traditional methodology, in that DIRECT-IP does not allow a fraction of a block to be extracted in a time period.

Like the traditional strategic open-pit mine planning methodology, DIRECT-IP suffers from some important limitations. First, its difficulty greatly increases with the size of the instance. Second, as in the case of the "best-possible" solution in the traditional approach, the solution of DIRECT-IP can be very fragmented, resulting in a schedule that is very difficult to implement in practice. DIRECT-IP does, however, have important advantages: First, it has the potential to yield significantly better solutions than the traditional approach. Second, it is easier to extend to more complex modelling requirements such as blending-requirements, multi-mine systems, and stockpile management.

METHODOLOGY AND RESULTS

Comparing the traditional methodology to the direct optimization methodology

For the purpose of this study we will ignore operational / spatial constraints (shape, size and fragmentation of extracted regions over time), and compare the direct approach to the traditional approach only in terms of net-present-value of the solution obtained.

Our comparison comprises five different instances block models. Three of them are publicly available in the MineLib website (http://mansci.uai.cl/minelib, Espinoza et al 2012): these are the McLaughlin, Marvin and W23 instances. The other two are Anonymous1 and Anonymous2. Information summarizing each of these instances is presented in Table 1. In all of these instances we ignore blending and stockpiling constraints if they exist.

Table 1 – Description of test block models.

Name	Number of Blocks	Number of Destinations	Number of Minerals	Discount Rate per Time Period
Marvin	8516	2	2	10%
W23	62300	4	1	10%
McLaughlin	180749	2	1	15%
Anonymous1	106409	2	1	10%
Anonymous2	129700	3	2	5%

We solve each instance twice; once with the Gemcom Whittle Nested Pit (GWNP) optimizer (traditional approach) and once with the DIRECT-IP integer programming formulation (direct optimization approach).

- Method 1: GWNP Optimizer. We solve the instance using the traditional approach as implemented by Gemcom Whittle version 4.2, a market leader in this class of software. We select the destination of each block using the "cash flow" method and we compute the pit shells for different revenue factors. We select the pit with revenue factor 1.0, and we schedule this pit using GWNP's "best case". Finally, we execute the cut-off optimization module in order to redefine the cut-off of each period, maximizing the discounted cash flow of the project.

- Method 2: DIRECT-IP. We begin by formulating the DIRECT-IP integer programming formulation, as described in the Introduction. We proceed by solving the Linear Programming relaxation of this formulation using the Bienstock-Zuckerberg (2010) lagrangian algorithm. In order to obtain an integer programming solution we use the TopoSort heuristics and Sliding Time Window heuristics described in Chicoisne et al (2012) and Munoz (2012).

The results of our experiment are summarized in Table 2. As can be seen from the "DIRECT-IP Improvement Over GWNP" column, DIRECT-IP outperforms GWNP in all the instances. In Marvin, the improvement is minimal (just 1.22%). However, in W23 and Anonymous2 the improvement is significant (over 12%). As can be seen from the "DIRECT-IP Optimality Gap" column, the values obtained by DIRECT-IP are not always optimal, but in general, they are very good. For example, in McLaughlin we know that the solution obtained by the DIRECT-IP method is within 0.28% of optimality, whereas for W23, the solution is only within 3.38%. In both of these cases there could exist a solution with higher objective function value than the one we found. If this were the case, it would indicate that value reported in the "Improvement of DIRECT-IP Over GWNP" column is an under-estimate of the improvement that can be obtained from direct optimization.

Table 2 – Computational results comparing DIRECT-IP and GWNP. For each instance, the value of "Improvement of DIRECT-IP Over GWNP" indicates the percentage improvement of the objective function of DIRECT-IP relative to GWNP. The "DIRECT-IP Optimality Gap" value indicates the integer programming optimality gap of the DIRECT-IP solution.

Instance	**Improvement of DIRECT-IP Over GWNP**	**DIRECT-IP Optimality Gap**
Marvin	1.22%	2.15%
W23	12.77%	3.38%
McLaughlin	5.91%	0.28%
Anonymous1	5.67%	0.75%
Anonymous2	12.06%	1.11%
Average	**7.53%**	**1.53%**

Analysis of the DIRECT-IP and GWNP solutions

Why does DIRECT-IP outperform GWNP? From the experiment just described this is not quite clear, and several possibilities arise. On the one hand it could mean that GWNP is a non-optimal implementation of the traditional method, in the sense that one of the three steps of the heuristic is not optimally carried out. On the other hand, it could be the case that GWNP is an optimal implementation of the traditional method, but yet, that the traditional method is limited by its three-step peace-meal approach to optimization. In this latter case, it could be that forcing the solution to extract blocks in the sequential order indicated by the phases leads to poor solutions. It might also be that forcing the solution to extract blocks in the order prescribed by the increments leads to the losses in value.

Is it fair to compare DIRECT-IP and GWNP in terms of the obtained objective function value? It might argued that it is indeed unfair, because GWNP produces very structured solutions that extract blocks by increments, whereas DIRECT-IP is allowed to generate much more scattered solutions where blocks are selected from all over the mine in a same time period. Consider, for example, a user interested in producing solutions that are more operationally tractable (i.e. better spatial properties). Such a user might get lucky, and find that the increments defined by the GWNP sequencing step have nice shapes. In this case, GWNP will likely produce more operationally practical solutions than those produced by DIRECT-IP. On the other hand, if the increments do not have desirable shapes, the user could use the Gemcom Whittle Width algorithm to obtain a new set of "nicer" increments, and then proceed to use the GWNP algorithm as before. It is not clear what option a DIRECT-IP user might have to produce solutions with more desirable shapes.

We now look to address these two concerns. On the one hand, we try to identify the reason that DIRECT-IP outperforms GWNP. On the other hand, we show how to adapt DIRECT-IP so as to generate solutions that are more comparable to those obtained by GWNP.

For this analysis we introduce two variants of the DIRECT-IP method. The first of these methods, which we call DIRECT-IP-Increments, adds constraints to the DIRECT-IP formulation so as to enforce that blocks are extracted in the order prescribed by the GWNP increments. The second method, which we call DIRECT-IP-Phases, adds constraint to the DIRECT-IP formulation so as to enforce that blocks are extracted in the order prescribed by the GWNP phases. The idea is that DIRECT-IP-Increments is a variant of the traditional approach where the production-scheduling step of the traditional three-step heuristic is done with integer programming. By comparing the DIRECT-IP-Increments solution to the GWNP solution we can assess how well GWNP is doing the production scheduling. DIRECT-IP-Phases is a variant of the traditional approach where the production-scheduling step of the traditional three-step heuristic is done with integer programming, but where it is not required that blocks be extracted in the order prescribed by the increments. By comparing the DIRECT-IP-Phases solution to the DIRECT-IP-Increments solution we can assess the cost that is incurred in the traditional approach by enforcing that phases should be extracted bench-by-bench. We now describe these methods in more detail:

- Method 3: DIRECT-IP-Increments. We solve a modified version of the instance as follows. We export the increments computed by GWNP, and we solve a modified version of DIRECT-IP in which the increments must be extracted in the order specified by the GWNP solution. That is, no block in increment I(i) can be extracted until all of the blocks in I(i-1) have been extracted. To achieve this, we add dummy blocks in between increments in order to impose the additional precedence constraints between blocks, just as we did in the DIRECT-IP-Phases approach. This method starts from the solution obtained by GWNP.

- Method 4: DIRECT-IP-Phases. We solve a modified version of the DIRECT-IP instance as follows. We export the pit shells computed by GWNP, and we solve a modified version of DIRECT-IP in which the pits must be extracted in the order specified by these pit shells. That is, no block in pit S(i) can be extracted until all of the blocks in S(i-1) have been extracted. To achieve this, we add dummy blocks in between the different pits in order to impose the additional precedence constraints between blocks. See Figure 2 for an illustration depicting the use of a dummy block to impose the sequencing constraint. After adding the dummy block we obtain a new instance of DIRECT-IP that can be solved with the same method we used for Method 2. This method starts from the solution obtained by DIRECT-IP-Increments.

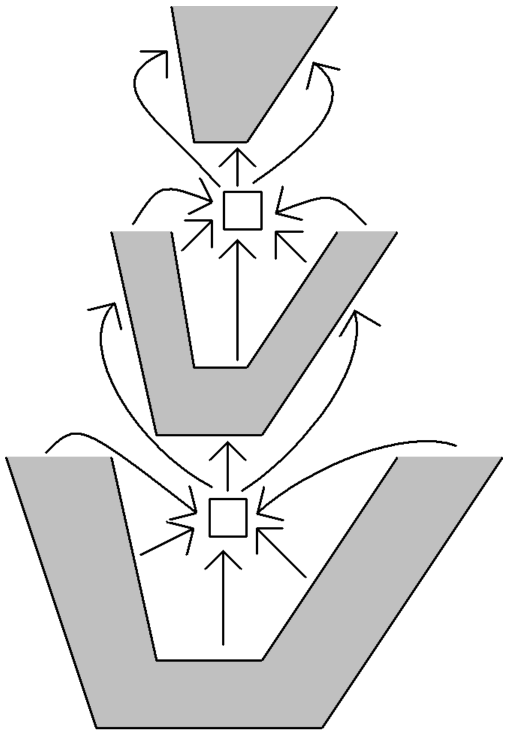

Figure 2 – Dummy blocks and its corresponding precedence constraints between blocks inside each phase, in order to impose the sequencing constraint.

The computational results obtained from solving the production scheduling problems with the new methodologies are described in Table 3.

Table 3 – Computational results comparing DIRECT-IP, DIRECT-IP-Phases, DIRECT-IP-Increments and GWNP. For each instance, the value of "DIRECT-IP Improvement Over GWNP" indicates the percentage improvement of the objective function of DIRECT-IP relative to GWNP. The values of the last two columns are analogous.

Instance	**Improvement of DIRECT-IP Over DIRECT-IP-**	**Improvement of DIRECT-IP-Phases Over DIRECT-IP-**	**Improvement of DIRECT-IP-Increments Over**

	Increments	**Increments**	**GWNP**
Marvin	1.22%	1.22%	0%
W23	5.32%	2.59%	7.07%
McLaughlin	5.91%	5.38%	0%
Anonymous1	4.86%	4.05%	0.77%
Anonymous2	1.69%	0.63%	10.20%
Average	**3.80%**	**2.77%**	**3.61%**

DISCUSSION AND CONCLUSIONS

As can be observed from Table 3:

- GWNP is not optimally solving the production-scheduling step of the traditional algorithm. In fact, in W23 and Anonymous2 there is a significant improvement obtained by DIRECT-IP-Increments over GWNP (7.07% and 10.20%, respectively). This might be explained by the fact that W23 and Anonymous2 are the only two data sets that consider more than two destinations.
- The cost of extracting phases bench-by-bench is negligible. This can be observed from the fact that the improvement of DIRECT-IP-Phases over DIRECT-IP-Increments is always below 5%.
- The traditional approach, when applied exactly, is surprisingly good, and achieves objective function values very near to those of the direct optimization approach. In fact, the improvement of DIRECT-IP over DIRECT-IP-Increments is always below 6%, and in average, is 3.8%. This is probably explained by the fact that the improvement afforded by DIRECT-IP over DIRECT-IP-Phases is very little, indicating that the practice of extracting blocks in the ordering indicated by the nested-pits (or phases) yields solutions of high value. In fact, for all instances except W23, the improvement of DIRECT-IP over DIRECT-IP-Phases is less than 1%.
- By modifying DIRECT-IP and solving the DIRECT-IP-Increments variant of the problem, it is possible to optimally solve the production-scheduling step of the traditional approach. This means that DIRECT-IP-Increments could easily be integrated into existing traditional software packages as an alternative algorithm to those currently employed. This has the advantage over simply using DIRECT-IP that the solutions that come out are more likely to have desirable shapes.

It should be noted that all five of the instances that we consider do not involve complicating issues such as blending and stockpiling. It is not clear how blending and stockpiling will change the results of our analysis. We expect that incorporating such constraints into DIRECT-IP should not be difficult, so we expect to conduct this experiment in the near future. It should also be noted that this entire analysis is not taking much into consideration important spatial constraints that ensure that the solutions obtained can be used in practice. This is a very important issue that should be considered in the future.

ACKNOWLEDGEMENTS

The authors would like to thank CONICYT grants ACT-88 and Basal Center CMM, Universidad de Chile. M.G. acknowledges FONDECYT grant #1110674. E.M. acknowledges FONDECYT grant #1130681. We would like to thank Redco Engineering for letting us use the Gemcom Whittle software.

REFERENCES

Bienstock, D., and Zuckerberg, M. (2010). Solving LP relaxations of large-scale precedence constrained problems. Proceedings of the 14th Conference on Integer Programming and Combinatorial Optimization (IPCO). Lecture Notes in Computer Science. Berlin. Edited by Friedrich Eisenbrand and Bruce Shepherd. Springer Heidelberg. pp. 1-14.

Chandran, B. G., and Hochbaum, D. S. (2009). A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem. Operations research. v. 57(2). pp. 358-376.

Chicoisne, R., Espinoza, D., Goycoolea, M., Moreno, E., and Rubio, E. (2012). A new algorithm for the open-pit mine production scheduling problem. Operations Research. v. 60(3). pp. 517-528.

Espinoza, D., Goycoolea, M., Moreno, E., and Newman, A. (2013). MineLib: a library of open pit mining problems. Annals of Operations Research. v. 206(1). pp. 93-114.

Hochbaum, D. S., and Chen, A. (2000). Performance analysis and best implementations of old and new algorithms for the open-pit mining problem. Operations Research. v. 48(6). pp. 894-914.

Hustrulid, W. A., and Kuchta, M. (2006). Open pit mine planning and design (Vol. 1). London, UK. Taylor and Francis.

Johnson, T. B. (1968). Optimum open pit mine production scheduling (No. ORC-68-11). Doctoral Thesis. Berkeley. Operations Research Center.

Khalokakaie, R. (1999). Computer-aided optimal open pit design with variable slope angles. Doctoral dissertation. Leeds. University of Leeds.

Lane, K. F. (1988). The economic definition of ore: cut-off grades in theory and practice. London, UK. Mining Journal Books.

Lerchs, H., and Grossmann, I. F. (1965). Optimum design of open-pit mines. CIM bulletin. v. 58(633). pp. 47-54.

Munoz, G. (2012). Modelos de optimizacion lineal entera y aplicaciones a la minería. Mathematical Engineering Thesis. Santiago. Dpt. Math. Engineering. Universidad de Chile.

Wharton, C. (1997). The Effect of Minimum Mining Width on NPV. In Proceedings of the 1997 Whittle Conference “Optimizing with Whittle”. Perth. WA. pp. 173-178.

Wharton, C. (2000). Add value to your mine through improved long term scheduling. In Proceedings Whittle North American Strategic Mine Planning Conference. Colorado. pp 1-13.

Whittle, J. (1990). Open Pit Optimization. SME and B. Kennedy. Surface Mining. pp. 470 – 475.

Whittle, J., and Wharton, C. (1995). Optimizing cut-offs over time. In Proceedings of the 25th International Symposium on the Application of Computers and Mathematics in the Mineral Industries. pp. 261-265.