A Branch and Bound Algorithm for Open Pit Grade Control Polygon Optimization

Matthew Deutsch[1*]
[1]*Maptek, United States*
*Corresponding author: matthew.deutsch@maptek.com

Grade control is a vitally important process in open pit mines. Misclassification of ore or waste due to suboptimal polygon design has an immediate and substantial impact on the mine's bottom line. Using an estimated block model with economic values, each block can easily be classified to maximize profit. However, this baseline classification of blocks does not necessarily satisfy mining width constraints. In this paper constructing a classification that both maximizes profit and satisfies the mining width is shown to be NP-Hard. A practical algorithm for generating near optimal, and often truly optimal, grade control polygons is introduced. This algorithm is based, in part, on branch-and-bound which is used to vastly reduce the search space.

Introduction

Open pit grade control is the process of sampling, estimating, classifying and mining material. Mines perform this process daily; drilling blast holes and taking samples, integrating those samples in some estimation framework to create a model, making tactical decisions on how material should be classified, and then excavating the material and hauling it to where it needs to go. This is a critical unit operation at mines today, in part because decisions made at this stage are irreversible. Depending on the operation, improvements to grade control can lead to a larger increase in profit than any other operational improvement [1].

The sampling, estimation, and mining aspects of grade control are extremely important, and often difficult to perform correctly due to many factors. However, this paper will not focus on those difficulties, and instead focus on the classification and selectivity aspects of grade control. We will assume that an appropriate grade control block model is given and the mining process is set. Then the best classification for each individual block is the classification which maximizes revenue. This best classification would be ideal, except most mines are not that selective, and do not mine a single block at a time. The concept of a selective mining unit (SMU), used in medium to long-range planning, does not apply directly in grade control and instead a larger collection of blocks is mined together. That is, a single block classified as waste surrounded by ore would be routed to the mill anyway. This selectivity constraint is often called the mining width.

Grade control engineers and geologists generally account for this mining width manually. They digitize grade control polygons on top of the grade control model, using some visual criteria to determine if a polygon is minable or not. However, this is a tedious and error prone process, and can lead to substantial economic loss due to unnecessary dilution and ore loss. In many cases the errors from heedlessly misclassifying material during manual polygon design outweigh the errors from the sampling and estimation process. Several techniques have been proposed to automate the process of designing minable grade control polygons, including some based on metaheuristics such as simulated annealing [2, 3], genetic algorithms [4], or hierarchical clustering techniques [5].

In this work a new method based on Branch and Bound and a thorough mathematical investigation of the problem is introduced. Section one will define the problem in more detail, and introduce a precise mathematical formulation. Then, in section two, an efficient algorithm is introduced and developed. The results of this algorithm on real mine datasets is shown in section three, followed by possible extensions and conclusions in section four.

Problem Definition

The grade control classification problem can be formulated as having two main inputs; the economic grade control model, and the mining width constraint. The model consists of a regular 2D grid of blocks, where each block contains variables which correspond to the revenue for each of the possible classifications. For example, the model

may have three variables for each block indicating the revenue if the block goes to the mill, the leach pad, or the dump. These values vary from block to block, and are both positive and negative. The revenue value is based on many different factors, some of which are local, such as gold content, and others which are global, such as gold price. This calculation is site and commodity specific, but boils down to revenues less costs. Figure 1 shows a grade control model colored by the best classification with a few callouts showing the revenue variables.

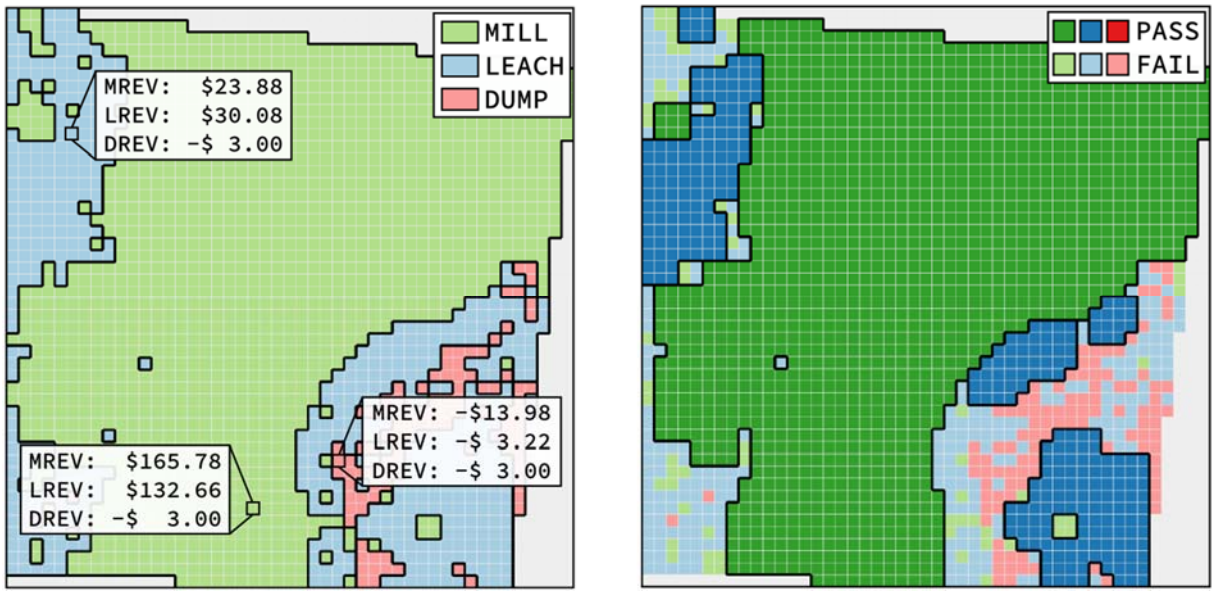

Figure 1: Economic grade control model colored by best classification (Left), Same model with regions that pass a 3x3 mining width element hatched (Right)

The second input is the definition of mining width. For this we use a structuring element, which is a small shape defined on a 2d grid where the values are either 1 (if the 'block' belongs to the structuring element) or 0 (otherwise). Typical structuring elements are rectangular, such as a 3x3 square, but they can also be irregular. Structuring elements are often used in mathematical morphology and image analysis [6].

The desired output of this problem is a classification on a block basis that maximizes value, and satisfies the mining width element such that every block 'fits' within at least one structuring element. The mining width constraint ensures that individual blocks, or small spurs are not constructed, because if a block is to have a classification at least a structuring elements worth of blocks around it must also have the same classification. In Figure 1, on the right, the original baseline classification is shown, and regions which satisfy a 3x3 mining width are shaded in.

Mathematical Formulation

We formulate the grade control classification problem as the following optimization problem. Given a universe of blocks U, a collection of possible classification C, a value function b, and many subsets $S = \{s_1, \dots, s_k\}$ derived from the mining width. The problem is to:

$$MAX \sum_{u \in U, c \in C} b_{u,c} \cdot x_{u,c} \tag{1}$$

subject to

$$0 \le x_{u,c} \le 1 \qquad integer,\ \forall u \in U, c \in C \tag{2}$$

$$\sum_{c \in C} x_{u,c} = 1 \qquad \forall u \in U \tag{3}$$

$$0 \le y_s \le 1 \qquad integer,\ \forall s \in S \tag{4}$$

$$\sum_{s:u \in S} y_s \ge 1 \qquad \forall u \in U \tag{5}$$

$$x_{s_{i,1}} = x_{s_{i,2}} = \cdots = x_{s_{i,l}} \qquad \forall s_i \in S | y_s = 1 \tag{6}$$

Equation 1 is the objective function. we would like to maximize the value function b, which depends on the block and the classification, using a decision variable x which indicates if a specific block u has a classification c. Equation 2 indicates that x is a binary decision variable. Equation 3 enforces that each block must have one, and only one, classification. Then to handle mining width we introduce a further decision variable y which is again a binary variable (Equation 4). Each block u, must be in at least one of the selected sets in S (Equation 5), along with the constraint that each selected set must only have blocks with the same classification (Equation 6).

The sets in S are constructed from the structuring element, and the area of interest. Each set s_i is a collection of blocks where the structuring element 'sits' on top of the block model. For example, if a 3x3 mining width is used in a 5x6 block model there are 12 different sets in S, 3 rows of 4, each consisting of 9 elements. The sets will necessarily overlap.

It is sometimes convenient to work with the decision variant of this problem, where we are given some number q in addition to all the other inputs, and we must report if a solution exists with a value greater than q, that is:

$$\sum_{u \in U, c \in C} b_{u,c} \cdot x_{u,c} > q \tag{7}$$

To prove that this problem is NP-hard we will reduce the exact cover problem to our grade control classification problem. In the exact cover problem, we are given a similar collection $A = \{a_1, \dots, a_r\}$ of sets, each of which is a subset of the universe U. We must select a subset $X \subset A$ of sets in A such that every element in the universe is in precisely one set in X. The exact cover problem is one of Karp's 21 NP-Complete problems, and is proven NP-Complete using a reduction from graph coloring [7].

To encode an exact cover problem as our grade control classification problem, add each set a_i in A to S, add a classification for each set such that the value, b is -1 for $x \notin a_i$ and 0 for $x \in a_i$. Then set q to be 0. If we can find an exact cover, we would have a matching maximum classification; by simply assigning each set in the exact cover to its corresponding classification. This implies that a yes answer to the exact cover problem implies a yes answer to this constructed grade control classification problem.

In the other direction, assume that there is a solution to the grade control classification problem such that the value is zero (our maximum). The solution to our problem is to choose a set cover from S, and a classification for each set, which we will represent as a pair (X, f), where $X \in S$ is the set of sets chosen for the cover, and $f: X \to C$ is a function that maps a set in X to a classification in C. It is sufficient to think of this function f as classifying 'sets', and not individual blocks, Equation 6 enforces each set in X consists of blocks of the same classification.

A solution (X, f) to our problem could be turned into the exact cover solution by taking each set in X along with its classification $f(x)$ and simply including the corresponding set from A. Recall that all the sets $X \in S$ and their classifications in C were constructed from A in the first place. A maximum valued solution (X, f) corresponds to an exact cover solution. Except, in our problem, sets with the same classification can overlap. The constraint in equation 5 only indicates that each block must be in at least one set of the final set cover, not exactly one. There are two possibilities here; sets could be completely contained, or could properly overlap.

If sets were perfectly contained, then (X, f) would contain at least two sets $x \in X$ and $x' \in X$ such that $x \subseteq x'$. If this occurs, removing x from X will not change the value of the solution as from Equation 3 we know that $f(x) = f(x')$. Removing all sets that are fully contained within some other set is easily accomplished in polynomial time and creates a new solution (X', f') that does not contain any full contained sets.

We still must prove that the solution (X', f') does not contain any sets $x \in X'$ and $x' \in X'$ such that $x \cap x' \neq \emptyset$. Suppose towards contradiction that there is some set $x_i \in X'$ such that $f(x_i) \neq c_i$. Then $f(x_i) = c_j$ for some $j \neq i$ with $x_j \in X'$ also. However, there is no set $x_k \in X'$ such that $x_i \subseteq x_k$, therefore there must be some block $b \in x_i$ that is not in x_j. This block b must then be assigned the classification c_j and have incurred a penalty of -1, which cannot be recovered by any means. We know that the value of (X', f') is 0, so this is a contradiction. In this constructed instance of the grade control problem there will be no overlapping sets. X' is, thus, an exact cover.

We have shown that a yes answer to our constructed instance of the grade control classification problem implies a yes answer to the exact cover problem, and vice versa. Which implies that a no answer to either problem also implies a no answer to the other and thus, the two problems are equivalent. Since Exact Cover is NP-hard, the grade control classification problem is also NP-hard, and if an algorithm existed to solve the grade control problem in polynomial time, then it could also be used to solve the Exact Cover in polynomial time, which is unlikely.

Algorithm Development

In the previous section, we defined the problem and proved that it was NP-hard. In this section, despite this complexity, we develop an algorithm for solving the grade control classification problem. Initially, it may seem reasonable to evaluate all possible solutions and pick the best. However, even in the case of a small block model, 1500 blocks, and a few classifications, 3, there are $3^{1500} \approx 10^{715}$, different possible solutions. Only a small fraction of the possible solutions satisfy the mining width constraint, but there is still far too many to evaluate directly. Instead, there are many techniques to practically solve, or approximate, NP-hard problems including; metaheuristics, integer programming, SAT solvers, enumeration techniques, and others.

The concept of a branch and bound algorithm was originally introduced in [8], although it was only first called "branch and bound" in [9]. As the name suggests, a branch and bound algorithm consists of two main operations which are applied to maximize the objective function; branching and bounding. The branching operation takes a problem node which represents some or all the solution space and splits it into two or more smaller nodes which contain a smaller fraction of the search space. The bound operation computes the upper bound of a node. If the upper bound of the node is less than the best result found so far, it can be trimmed, reducing the search space.

Use s to denote a solution instance which will consist of some assigned, and some unassigned blocks. An unassigned block could take any classification. f represents the objective function. g is the function to compute the upper bound of a node which assumes perfect selection in unassigned blocks. The general form of the algorithm is:

1. Initialize some heuristic solution to the problem s_h then set $B \leftarrow f(s_h)$, B will store the highest value so far. Set $s_b \longleftarrow s_h$, s_b will store the current best solution.
2. Initialize a queue with one node n_0 where every block is unassigned.
3. Until Queue is empty
4. Take a node n from the front of the queue
5. If n is a fully specified node (no unknown blocks) and $f(n) > B$
6. $B \leftarrow f(n), s_b \longleftarrow n$
7. Else
8. If $g(n) \leq B$ or n does not satisfy mining width
9. Continue
10. Else
11. Branch on n by constructing $|C|$ more nodes, one for each classification, setting the first unknown block to c_i
12. For each new node n_i
13. If $g(n_i) > B$
14. Add n_i to the queue

The algorithm as written defers checking mining width until a node is pulled off the queue because checking mining width is the most expensive operation and requires checking many blocks. The upper bound is checked before putting the node onto the queue and after taking a node off the queue because it is a very cheap operation, and the best may have changed between when the node is pushed and when it is up for evaluation. If the queue becomes very large it is useful to sweep through the queue every so often removing all nodes with $g(n) < B$ so that that memory can be reused.

The ordering of the queue has a very meaningful impact on the performance of the algorithm. Through testing, ordering first by depth and then by upper bound yielded the best results. This reaches a good solution quickly, and trims more nodes earlier. If the queue is sorted by upper bound first it takes many iterations to reach the first solution. This is because the upper bound over-estimates the true upper bound, as the upper bound of an instance is calculated assuming perfect selection of the unknown blocks.

Metaheuristic Refinement
Branch and Bound does admirably, but does not terminate in a reasonable amount of time when the problem has more than a few hundred blocks. The over estimation of the upper bound, coupled with the tendency for different solutions to have very similar values means that not enough branches are trimmed, and branch and bound require many iterations. When this happens, we halt the branch and bound and use a metaheuristic to improve the result. Many different metaheuristics could be used: Simulated annealing, genetic algorithms, tabu search, and others.

Simulated annealing [10, 11] is reasonable. It is a randomized algorithm, wherein the current solution is repeatedly changed. Changes which increase the objective function are always kept and changes which decrease the objective function are accepted with decreasing probability. The ability to accept changes which make the solution worse is used to avoid getting stuck in local optima. In the grade control classification problem simulated annealing can be directly applied by repeatedly changing small collections of blocks, provided they do not violate the mining width.

Reducing the Problem
An important aspect of the grade control classification problem is that we know, from the beginning, what the 'best' classification for each block is, and the maximum possible value. We used this knowledge to develop the bounding function for the branch and bound optimization, but this can also be used to reduce the search space in the beginning. If there are large swaths of blocks where each block has the same baseline classification, they do not need to be evaluated in the branch and bound optimization or in any subsequent steps. Also, if this reduction leads to several separate components, each component can be optimized independently.

The fixed regions are roughly identified by using Minkowski subtraction of each baseline classification with the mining width element. They then are manipulated slightly to pass the mining width; details are omitted for space. In Figure 2 we show the results of the reduction algorithm which identifies blocks which do not need to be looked at. Both the fixed regions and the changeable regions satisfy the mining width. On the left a 2x2 mining width element is used, and on the right a 3x3 mining width element is used.

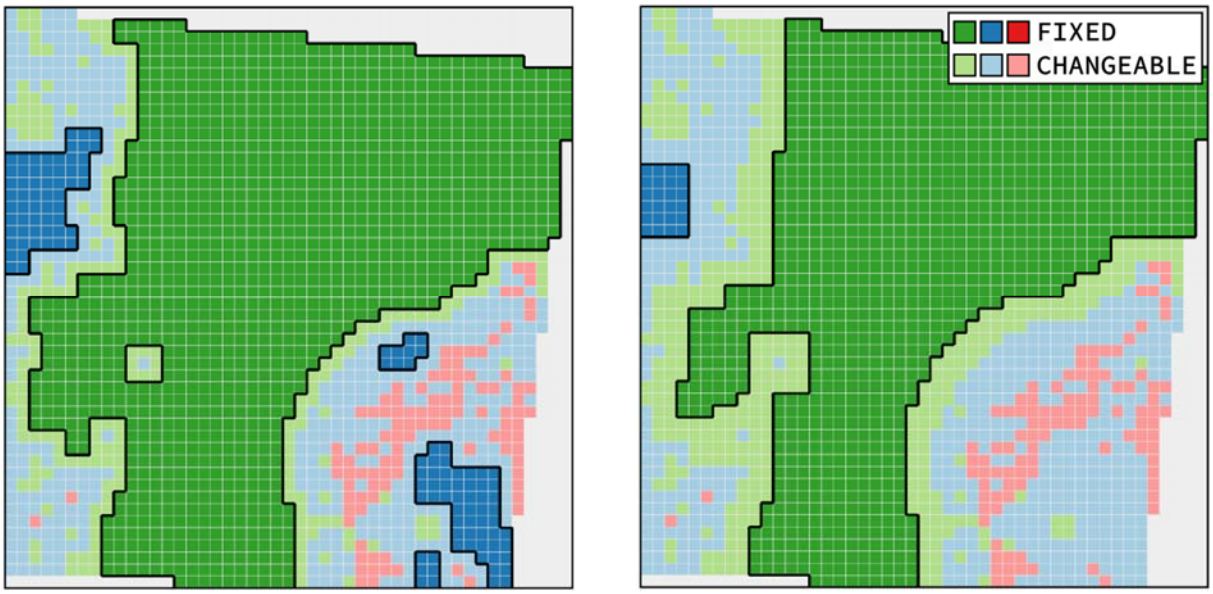

Figure 2: Blocks which do not need to be evaluated in the optimization, with a 2x2 mining width (Left), and a 3x3 mining width (Right)

Using the 2x2 element the problem is reduced from an original 2023 blocks, or $\sim 10^{935}$ permutations, to 708 blocks, or $\sim 10^{338}$ permutations. This problem is then split into three problems of 304, 9, and 395 blocks. With the 3x3 element the problem is reduced to 959 blocks, and two problems with 448 and 551 blocks each.

Practical Challenges
There are several practical challenges when implementing this algorithm. For example, the edges of the block model and the area of interest may need to have different semantics depending on the operation. A solution may be allowed

to 'mine' outside the area of interest if it's already been mined previously but not if that area is the highwall. Also, there may be custom constraints on a block by block basis; certain blocks might not be allowed to have certain classifications. These constraints are handled by having three different types of constraints:

- Minable blocks allow the mining width to fit in a location.
- Each block also has a set of Allowed classifications.
- Enabled blocks are blocks which the optimizer will possibly change. For a block to be enabled it must be both minable, and have at least two allowed classifications.

These constraints are sufficient to describe the full nature of the problem, but can lead to an unsatisfiable problem. To identify this, the problem is formulated as a SAT problem and solved quickly with no expectation of value - this generates either a baseline solution or a proof of unsatisfiability, which should prompt changes to the constraints.

Another challenge is with custom mining widths. The problem, as defined, allows for any arbitrary mining width element, square, rectangular, or something different. This can be used to handle mining direction, and to customize the algorithm for each mining operation. However, the mining element can occasionally have unexpected, cascading effects - especially around the edges of the deposit. It is important that the mining width element 'tiles' nicely, or else setting a block a classification could force the algorithm to set the next block that same classification, and so on.

Results

Figures 3 and 4 show the results of the algorithm on two different datasets. Misclassified blocks are shown lightly hatched. A 3x3 and 4x4 mining width are shown for both datasets; along with an irregular angled mining element for Dataset A, and a 5x2 mining width element for Dataset B. The results are summarized in Table 1.

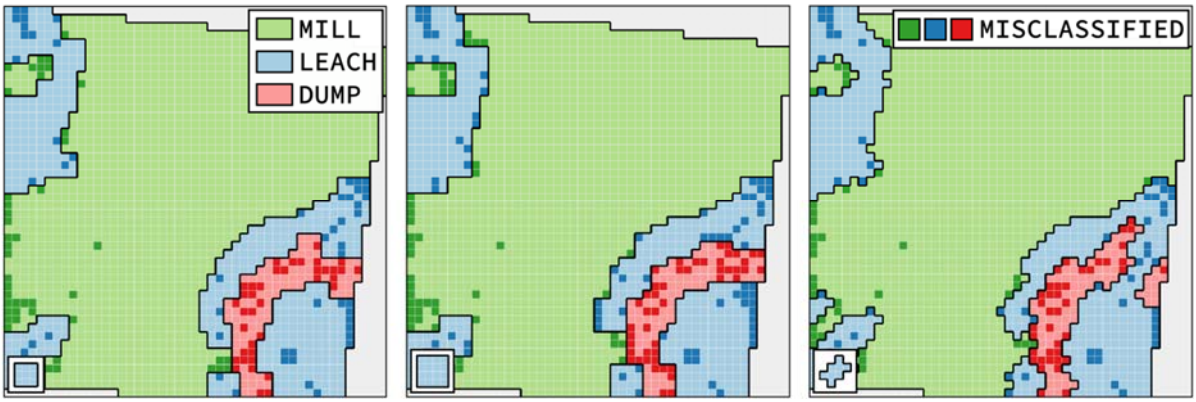

Figure 3: Results on Dataset A with mining elements: 3x3 (Left), 4x4 (Center), and irregular (Right)

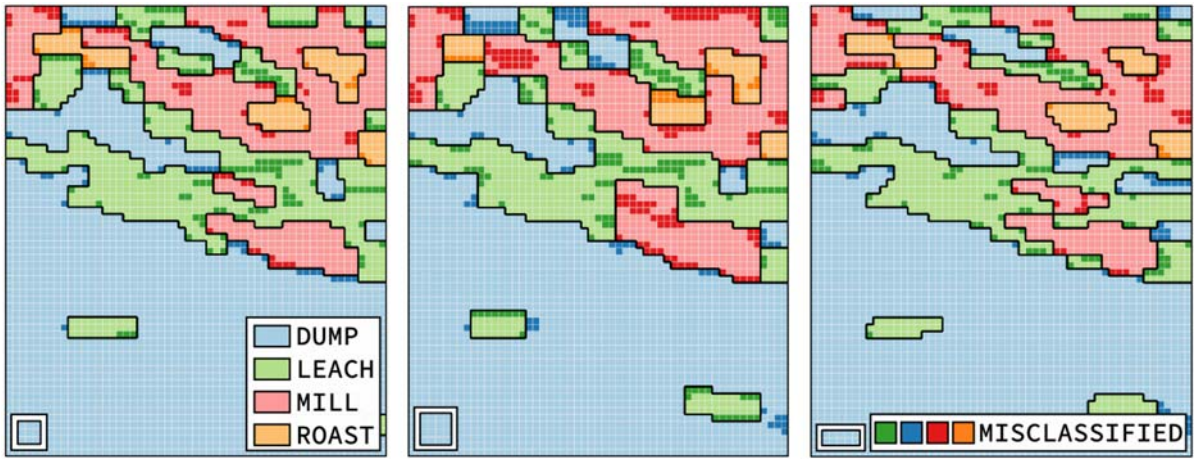

Figure 4: Results on Dataset B with mining elements: 3x3 (Left), 4x4 (Center), and 5x2 (Right)

Dataset A contains economic information for each block, however Dataset B does not. Dataset B only contains a prior classification based on cutoffs, this is optimized by assigning the economic value of a correct classification 1, and all others 0. This essentially optimizes to minimize misclassifications.

	Dataset A			Dataset B		
	3x3 MW	4x4 MW	Irregular	3x3 MW	4x4 MW	5x2 MW
Final Value	$219,655	$219,369	$219,649	N/A	N/A	N/A
Final / Upper	99.7%	99.6%	99.7%	N/A	N/A	N/A
Classified	1886	1849	1890	3284	3151	3293
Misclassified	137	174	133	291	424	282
Class / Blocks	93.2%	91.4%	93.4%	91.9%	88.1%	92.1%

Table 1: Summary of the Optimization Results

As the mining width element's size increases so too does the number of misclassified blocks. The tradeoff between selectivity and value is intuitively easy to understand, but can be hard to quantify. In Dataset A going from a 3x3 mining width to a 4x4 mining width leads to an additional 37 blocks being misclassified, but only a very minute change in value. If the 4x4 mining width corresponds with larger equipment which can extract the material quicker, then it may be worthwhile to accept that small loss in value. Each of the optimization runs in this analysis took less than 30 seconds to complete.

The results from this optimization are on a block basis, but the actual mined polygons are not expected to follow block boundaries precisely. An algorithm has been implemented to generate these minable polygons as a constrained contouring operation where the polygons are penalized for extra vertices, short segments, and sharp angles. Details are omitted for space, an example based on Dataset A with the 3x3 MW is shown in Figure 5.

Conclusions

This paper introduced a new algorithm for solving the grade control classification problem based, in part, on branch and bound. This algorithm efficiently explores the large search space, and can be used with arbitrary mining width constraints to generate optimal grade control polygons. There are many advantages to automating grade control polygon design, such as improved operational efficiency, minimizing ore loss and dilution, and facilitating sensitivity studies and more structured analysis.

Constrained grade control classification was shown to be NP-hard, but that does not preclude an efficient algorithm for real world cases. The combination of exact and metaheuristic optimization described in this paper performs very well on models with several thousand blocks and up to a dozen classification. For larger models the algorithm will slow down, and optimality is harder to guarantee - the search space is simply too large. This algorithm should be applied on a blast to blast basis, and if the model is still too large, reblocking could be used.

Possible avenues for improvement include accounting for blending, optimizing in the presence of uncertainty, and incorporating operational constraints such as target tonnages. In many deposits blending is extremely important, by blending different material types together a mine may be able to process more material, and recover more value. The current difficulty with supporting blending is defining the precisely how blending is input to the problem, and what effects it has on value. With uncertainty and operational constraints there are difficulties in weighting the different components of the objective function, and in computing an accurate upper bound, but the algorithm itself is flexible enough to handle an arbitrary objective function.

Open pit mines have a lot of opportunities to improve grade control, and optimal polygon design is a big part of that. Engineers and geologists should be making informed decisions on how best to guide the mining operation - not having to digitize polygons every morning.

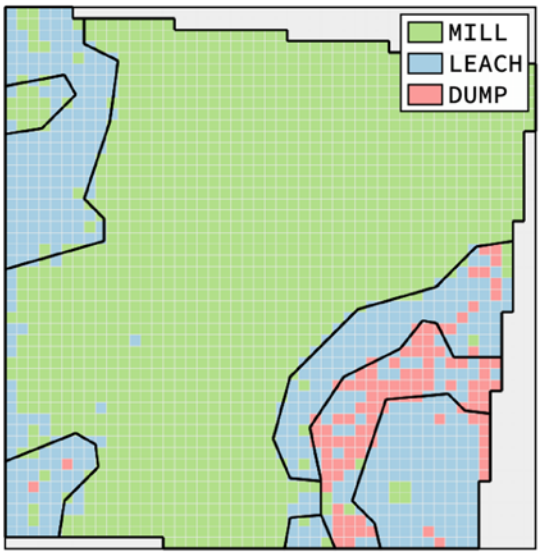

Figure 5: Improved contouring following block optimization with a 3x3 mining width.

Acknowledgements
The author would like to thank the management of Maptek for permission to publish this paper, and acknowledge the discussions and contributions made by the following people: D. Paek, K. Ramsay, S. Uecker, B. Ramsay, M. Williams, R. Melito, and J. Underhill. The author would also like to acknowledge the assistance, data, and fruitful discussions with the following members of Newmont Mining Corporation: M. Godoy, N. Kusuma, and J. Deutsch, and the following members of Barrick Gold Corporation: J. Baar, and C. Cavasin.

Disclosure Statement
Maptek sells a commercial implementation of this algorithm in the Vulcan 10.1 software package.

References
[1] A. J. Sinclair, G. H. Blackwell, “Applied mineral inventory estimation”, Cambridge University Press, 2002
[2] E. Isaaks, I. Treloar, T Elenbaas, “Optimum dig lines for open pit grade control”, Proceedings of Ninth International Mining Geology Conference, 2014
[3] C.T. Neufeld, K.P. Norrena & C.V. Deutsch “Guide to Geostatistical Grade Control and Dig Limit Determination”, Centre for Computational Geostatistics, 2015
[4] J. R. Ruiseco, J. Williams & M. Kumral, “Optimizing Ore-Waste Dig-Limits as Part of Operational Mine Planning Through Genetic Algorithms.” Natural Resources Research, 25, 473-485, 2016
[5] M. Tabesh, H. Askari-Nasab, “Automatic creation of mining polygons using hierarchical clustering techniques”, Journal of Mining Science, 49, 426-440, 2013
[6] J. Serra, “Image analysis and mathematical morphology”, v. 1. Academic press, 1982.
[7] R. M. Karp, “Reducibility among combinatorial problems”, Complexity of Computer computations, 85-103, 1972
[8] A. H. Land, A. G. Doig, “An automatic method of solving discrete programming problems”, Econometrica, 497-520, 1960
[9] J. DC. Little, K. G. Murty, D. W. Dura, C. Karel, “An algorithm for the travelling salesman problem”, Operations Research, 11, 972-989, 1963
[10] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi. "Optimization by simulated annealing." Science, 671-680, 1983
[11] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, P. B. Kramer, “Numerical Recipes: The Art of Scientific Computing”, 1987