Mining, Metallurgy & Exploration (2019) 36:757–764
https://doi.org/10.1007/s42461-019-0062-x

Open-Pit Mine Optimization with Maximum Satisfiability

Matthew Deutsch[1]

Received: 15 January 2019 / Accepted: 14 March 2019 / Published online: 9 April 2019

Abstract
A common casualty of modern open-pit mine optimization is the assurance that the resulting design is actually achievable. Optimized mine plans that consider value and a bare minimum of precedence constraints do not, in general, translate into practical, operational mine designs that can be used in the field. Ultimate pits may come to a sharp point at the bottom. Schedules may require taking small parcels of material from many disparate areas of the pit in a single period, and grade control polygons may be ragged, narrow, and not minable with realistic equipment. In this paper, all of these problems are addressed by encoding these three fundamental open-pit mine optimization problems as maximum satisfiability problems. Maximum satisfiability provides a useful framework for problems that are non-linear and may guarantee the optimality that metaheuristics cannot.

Keywords Mine planning · Optimization · Maximum satisfiability · Ultimate pit · Mine scheduling · Mining width

1 Introduction

Open-pit mines are large and complicated operations, which require significant initial and ongoing investment. A single mine may employ hundreds or thousands of people and create substantial benefits for both the surrounding and global communities. Planning an operation of this magnitude requires a lot of effort and cannot be done optimally by hand.

Mining engineers today rely on many techniques from operations research to guide decision-making and to maximize the value of the mine. Techniques including mixed integer linear programming, metaheuristics, and network algorithms are all used to solve different problems throughout mine design and optimization. Techniques based on Boolean satisfiability, including the optimization extension of maximum satisfiability, are used in many other fields, including electronics design, scheduling, and artificial intelligence, but have not seen much use in mining.

We claim that maximum satisfiability is a useful and meaningful way of expressing and solving optimization problems in mine planning. This technique does not come without challenges; the problem is NP-hard, and modern techniques for solving these problems can struggle with the number of variables and clauses that a typical mining problem requires. Also, some of the constraints that mining must contend with do not fit nicely into the satisfiability paradigm. Despite these challenges, we believe that maximum satisfiability provides a useful framework for considering operational constraints and expressing some problems that other techniques cannot easily express. Maximum satisfiability can be solved exactly, yielding the optimal answer which metaheuristics cannot guarantee.

In this paper, we show the applicability of maximum satisfiability to mining by specifying three fundamental problems in open-pit mine optimization. We show how these fundamental problems are formulated and how they are extended within the framework of maximum satisfiability. For the remainder of the paper, we first give a brief background on common elements between the three problems and an introduction to satisfiability. Then, we tackle the three problems: the ultimate pit problem, the block scheduling problem, and the grade control polygon problem. In the ultimate pit problem, we specifically show how to extend the problem to support a minimum mining width—a coveted result for many mining engineers. We then discuss some of the complexities and shortcomings of this framework and possible research directions.

2 Background

2.1 Block Models

The underlying data structure for all the problems in this paper is a regular block model. We define the geometry of a block

✉ Matthew Deutsch
mvdeutsch@mines.edu

[1] Colorado School of Mines, Denver, CO, USA

model with nine parameters: xmin, ymin, and zmin are real numbers which indicate the coordinates of the block model origin which are the leftmost, frontmost, lowest point on the block model. xsiz, ysiz, and zsiz are real numbers which indicate the size of the blocks. xnum, ynum, znum are positive integers which indicate how many blocks there are in the x, y, and z directions. These parameters are shown in Fig. 1 below.

Each block is identified in two ways: Using three indices ix, iy, and iz which are indices in the x, y, and z directions, respectively, or by using a single index idx which counts first along x, then y, and then z. For the three indices, the ix index increases in the x direction from 0 to xnum − 1, the iy index increases in the y direction from 0 to ynum − 1, and the iz index increases in the z direction from 0 to znum − 1. We calculate the one-dimensional grid index idx from the three indices as

$$\mathrm{idx} = \mathrm{ix} + \mathrm{iy} \times \mathrm{xnum} + \mathrm{iz} \times \mathrm{xnum} \times \mathrm{ynum}$$

The three grid indices are calculated from the one-dimensional grid index as follows, where / and % are integer division (truncates) and the modulus, respectively.

$$\begin{aligned} \mathrm{ix} &= \mathrm{idx}\%\mathrm{xnum} \\ \mathrm{iy} &= (\mathrm{idx}/\mathrm{xnum})\%\mathrm{ynum} \\ \mathrm{iz} &= \mathrm{idx}/(\mathrm{xnum} \times \mathrm{ynum}) \end{aligned}$$

Each block can hold several different attributes such as a metal grade or rock type, but in this paper, we are primarily interested in economic block values which have units of dollars per ton or dollars. These values indicate the profit of extracting this block, or the profit of sending a block to a given destination. The economic block value is calculated as revenues less costs, but it is often not that simple in practice as the mine must consider different processes, blending, royalties, recoveries, geometallurgical properties, and more.

2.2 Precedence Constraints

In the ultimate pit problem and the block scheduling problem, we must consider the precedence constraints. The precedence constraints encode the physical relationships between blocks and the notion that this block cannot be mined until all these other blocks are mined. Typically, we connect each block to a set of blocks above it in a pattern. There are several patterns which are often considered.

In their seminal 1965 paper, Lerchs and Grossmann proposed two simplistic precedence patterns for three-dimensional block models (a and b in Fig. 2) [1].

Gilbert (1966) developed the 1:5:9 pattern where the precedence pattern varies by level with each block connected to either the five blocks above it in a cross or the nine blocks above it in a square (c in Fig. 2) [2]. Lipkewich and Borgman (1969) introduced the "knight's move" pattern which roughly approximates 45 ° (d in Fig. 2) [3].

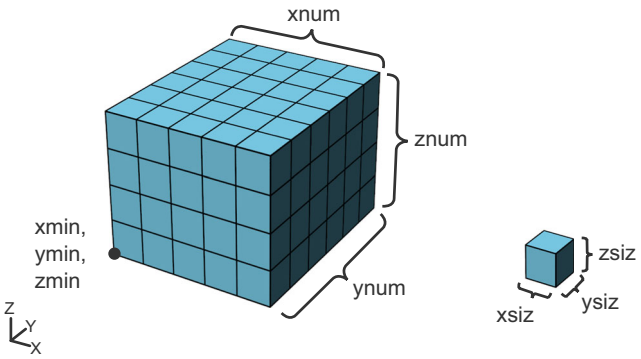

Fig. 1 Description of a regular 3D block model and its defining parameters

Often, the pit slopes are not constant throughout the deposit because of geotechnical constraints, and the slopes vary by location and direction. There are several techniques for locally varying slope constraints including Chen (1976), Khalokakaie and Dowd (2000), and Caccetta and Giannini (1988) [4–6].

Whatever the pattern is, the set of entire precedence constraints comes from considering the given pattern across all blocks and handling any edge effects. That is, the set of all precedence constraint P, contains pairs of block indices (a, b), which indicate that if we want to mine block a, we must first mine block b.

2.3 Satisfiability

In this section, we briefly review the history of satisfiability, present the standard form for satisfiability problems, and show a few examples. "Satisfiability" or "SAT" is the abbreviated form of the "Boolean satisfiability problem", which is sometimes also called the "propositional satisfiability problem." We are specifically interested in the optimization extension, the "maximum satisfiability problem" or "Max-SAT."

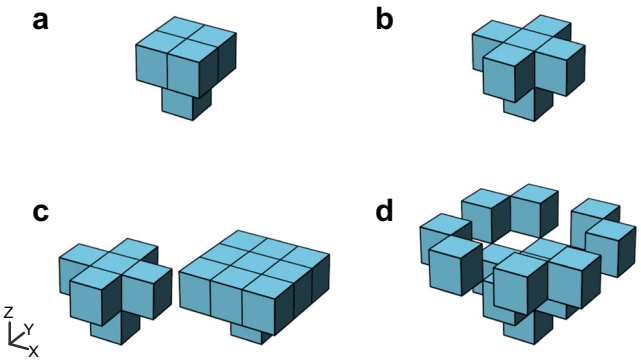

Fig. 2 Possible precedence patterns. **a**, **b** from Lerchs and Grossmann 1965. **c**(1:5:9 pattern) from Gilbert (1966). **d** (Knight's move pattern) from Lipkewich and Borgman (1969)

Satisfiability has its roots in classical studies of logic from millennia ago. In more recent times, Alfred Tarski, Claude Shannon, and many others have developed and formalized the field [7, 8]. In our context, Davis and Putnam's work in 1958 really started things off by using the conjunctive normal form and providing the Davis-Putnam procedure to evaluate satisfiability formulas [9]. Many more researchers have expanded on the Davis-Putnam procedure, most notably Loveland and Logemann who worked to extend the procedure into the seminal Davis-Putnam-Loveland-Logemann (DPLL) algorithm for determining if there is a solution to a given Boolean formula [10]. Franco (2009) provides a comprehensive history of the field [11].

Boolean satisfiability problems work with an infinite set of Boolean variables X. A single literal l is some variable $X_i \in X$ or it is negation $\neg X_i$. A clause C is a disjunction of literals, as $C = l_1 \vee l_2 \vee \ \ldots \ \vee l_k$. A SAT formula ϕ is a conjunction of clauses, as $\phi = C_1 \wedge C_2 \wedge \ \ldots \ \wedge C_m$. This format for expressing Boolean formulas is called conjunctive normal form, or CNF. It may seem restrictive, but many problems are easily expressed in CNF and all propositional formula can be converted into CNF [12].

We seek an assignment of values to each variable that satisfies the original formula. Because the formula is in the conjunctive normal form, every clause must be satisfied. Then, for a single clause to be satisfied, at least one literal must be satisfied. For a literal to be satisfied, the assigned value must be true if the literal is X, or false if the literal is $\neg X$. For example, the following Boolean satisfiability formula is satisfied with the assignment $X_1 \leftarrow$ false, $X_2 \leftarrow$ false, $X_3 \leftarrow$ true, but it is not satisfied with any other assignment.

$$\phi_1 = (X_1 \vee \neg X_2) \wedge (X_2 \vee X_3) \wedge (\neg X_1 \vee \neg X_3) \wedge (\neg X_1 \vee \neg X_2 \vee X_3)$$

There could be formulas where there is no solution, for example,

$$\phi_2 = \begin{array}{l} (X_1 \vee X_2) \wedge \\ (\neg X_1 \vee X_2) \wedge \\ (X_1 \vee \neg X_2) \wedge \\ (\neg X_1 \vee \neg X_2) \end{array}$$

Note how the above formula systematically rules out all four of the possible true or false assignments to two variables. This formula is unsatisfiable. There is no assignment to the literals which will satisfy all the clauses.

The result from solving a Boolean satisfiability problem is either "SAT", which indicates that the formula is satisfiable and has an appropriate assignment, or "UNSAT", in which case the formula is unsatisfiable, and no assignment exists. This has many uses in circuit design, automatic theorem proving, dependency management, and other fields, but is less obviously useful in mining. We are generally not only looking for any design that satisfies our constraints but also instead we want the optimal design. For this, we will need to extend Boolean satisfiability.

In maximum satisfiability, we associate each clause with a weight, so the clause becomes a pair (C, w) where the weight is a natural number or infinity. In this paper, if the weight is infinity, we will omit the weight to save space. We consider the weight equivalently as either the benefit for satisfying the clause or the penalty for not satisfying the clause, with our goal being equivalently finding the assignment that maximizes the overall benefit or minimizes the overall penalty. We consider the clauses with infinite weight as hard clauses in that they must be satisfied and clauses with a natural weight are soft clauses.

We can revisit the above unsatisfiable formula above and add weights as follows:

$$\phi_3 = \begin{array}{l} (X_1 \vee X_2, \infty) \wedge \\ (\neg X_1 \vee X_2, 1) \wedge \\ (X_1 \vee \neg X_2, 5) \wedge \\ (\neg X_1 \vee \neg X_2, 4) \end{array}$$

There are now three different solutions to this problem, note how the infinite weight clause makes $X_1 \leftarrow$ false, $X_2 \leftarrow$ false unsatisfying. The best assignment is $X_1 \leftarrow$ true, $X_2 \leftarrow$ false. For equivalently, a penalty of 1 or a total benefit of 9. This is a weighted partial MAX-SAT problem and is the format for the problems in this paper.

This format is used in a variety of other large-scale applications. Marques-Silva provides a review of practical applications of Boolean satisfiability to a variety of problems including artificial intelligence, model checking, and electronic circuit design [13]. Berg et al. provide realistic benchmarks and applications of maximum satisfiability to three problems in data analysis [14]. This format lends itself to a wide range of research areas, and now, we will apply it to mining.

3 Ultimate Pit Problem

The ultimate pit problem is to determine the final pit contour such that the mine extracts all the economic ore and leaves any unnecessary waste in place. This problem ignores complexities such as the time value of money and any mining or mill capacities. It is simplistic by today's standards but is still useful in practice. It can be used to help decide where to develop permanent infrastructure, to help limit the size of the scheduling problem, and to roughly evaluate mine life and project economics. The ultimate pit problem is schematically shown in Fig. 3, with the normal pit limits on the left and the

augmented results considering a minimum mining width on the right.

Lerchs and Grossmann were the first authors to describe the ultimate pit problem in 1965 [1]. In their paper, Lerchs and Grossmann developed a graph-based algorithm to solve the ultimate pit problem which remained in use in the mining industry until within the last decade or so. Lerchs and Grossmann indicated that the problem could be expressed as a flow problem but recommended their graph-based algorithm instead. In 1968, Johnson also showed that the ultimate pit problem could be expressed as a linear program and that the dual of this LP model is a flow problem [15].

This transformation of the ultimate pit problem into a flow problem proves extremely advantageous. Flow-based algorithms are many times faster than the graph-based Lerchs and Grossmann algorithm and yield precisely the same results. Push-relabel (Goldberg and Tarjan 1988) and pseudoflow (Hochbaum 2001) are two sophisticated flow-based algorithms that are commonly used [16, 17]. In a recent study (Deutsch 2014), pseudoflow was the fastest algorithm considered for the ultimate pit problem [18]. In this paper, we provide yet another, slower, means of expressing the ultimate pit problem, but this time in terms of maximum satisfiability.

The ultimate pit problem is encoded into maximum satisfiability simply. For each block in the block model, BM, add a variable X_i, this variable indicates if the block is mined $X_i \leftarrow$ true or is left in place $X_i \leftarrow$ false. Also, add a length 1 clause for each block with the block's variable and a weight equal to the economic block value. Then, for each precedence constraint, add a length 2 clause with infinite weight of the form $(\neg a \vee b)$ for each precedence relationship where a depends on b. The encoding is

$$\phi_{UPP} = \bigwedge_{i \in BM} (X_i, EBV_i) \bigwedge_{i,j \in \boldsymbol{P}} (\neg X_i \vee X_j)$$

This encoding is appealing because it is straight-forward and simple and because it leads to precisely the same results as the flow-based solution, but, it is not practically very useful yet. We have just shown a slower and less efficient way of representing the ultimate pit problem. The value of this encoding comes from how simply we can add a minimum mining width constraint.

The purpose of the minimum mining width constraint is to avoid the greedy tendency of the calculated ultimate pit to come to a sharp point at the bottom of the pit that the equipment could not even fit into. Mining engineers are used to handling this constraint manually when designing the pushbacks, but it can be difficult to optimally decide between chopping the blocks that do not satisfy the minimum mining width and expanding the pit manually.

We encode the minimum mining width by considering a structuring element. This term, borrowed from mathematical morphology, refers to a small shape or collection of blocks which is generally very simple, such as a 3 × 3 square or a small cross. This structuring element is then scanned over the entire block model, and for each location in the block model, we add an additional variable mw_l to the encoding, and a series of infinite weight clauses which enforce that this variable is true if and only if all the blocks which belong to it are true. Then, for each block in the model, we add an infinite weight clause which says either this block is not mined or at least one of the overlapping mining width variables is true. That is, given a collection of mining width sets S augment the encoding is as follows:

$$\phi_{UPPMW} = \phi_{UPP} \wedge \bigwedge_{i \in \boldsymbol{S}_l} (\neg mw_l \vee X_i) \quad \forall l \in \boldsymbol{S} \bigwedge_{i \in BM} (\neg X_i \vee mw_l \vee \ldots \vee mw_n)$$

The following example, Fig. 4, shows a very small example of the encoding with only seven blocks (a…g). For brevity, the mining width constraint is only added to the bottom bench of three blocks. A two block mining width is considered, so there are two mw variables (mw1, mw2). The precedence constraints are shown with arrows, and the economic block values are in the nodes.

The full encoding of this ultimate pit problem considering a minimum mining width constraint into maximum satisfiability is as follows:

$$\begin{aligned}\phi_{\mathrm{ex}} = {} & (\neg a, 3) \wedge (\neg b, 2) \wedge (\neg c, 3) \wedge (\neg d, 3) \wedge \\ & (e, 1) \wedge (f, 6) \wedge (g, 2) \wedge \\ & (\neg e \vee a) \wedge (\neg e \vee b) \wedge \\ & (\neg f \vee b) \wedge (\neg f \vee c) \wedge \\ & (\neg g \vee c) \wedge (\neg g \vee d) \wedge \\ & (\neg \mathrm{mw1} \vee e) \wedge (\neg \mathrm{mw1} \vee f) \wedge \\ & (\neg \mathrm{mw1} \vee f) \wedge (\neg \mathrm{mw1} \vee g) \wedge \\ & (\neg e \vee \mathrm{mw1}) \wedge \\ & (\neg f \vee \mathrm{mw1} \vee \mathrm{mw2}) \wedge \\ & (\neg g \vee \mathrm{mw2})\end{aligned}$$

4 Block Scheduling Problem

In the block scheduling problem, we are not only concerned with which blocks to mine but also when to mine them. This problem is shown schematically in Fig. 5. Lerchs and Grossmann recommended a technique they called parametric analysis whereby the volume of the pit was successively reduced, by reducing the block values, to achieve a sequence

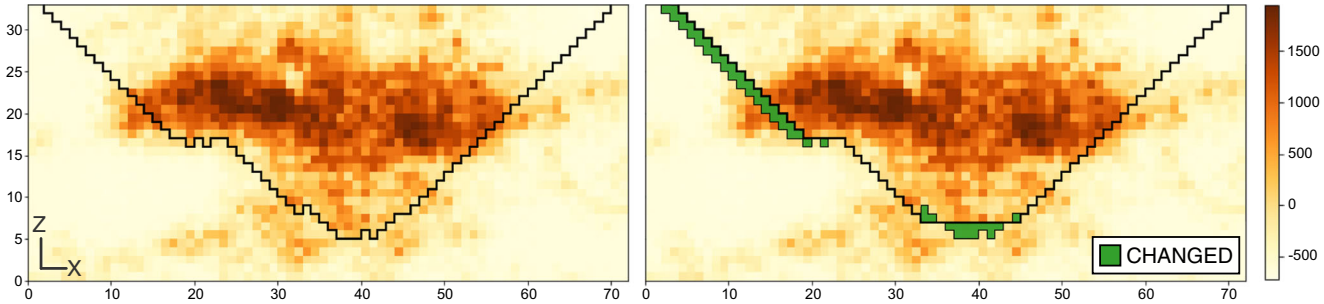

Fig. 3 The ultimate pit problem. Left, the ultimate pit as calculated with pseudoflow. Right, the ultimate pit as calculated with maximum satisfiability with a mining width of 12 blocks. Model is colored by economic block value

that they claimed would maximize the integral of the cash flow [1]. Additional early work in this area by Dagdelen and Johnson (1986) worked to maximize net present value subject to production constraints by using a Lagrangian relaxation of the problem [19]. This is a heavily researched problem with many exact and metaheuristic approaches; Newman (2010) provides an overview [20].

In the block scheduling problem, we consider a set of variables $X_{i,t}$ for each block $i \in \mathrm{BM}$; these variables indicate if the block is mined in time period $t \in T$. We add hard clauses to ensure that the block is mined in at most one time period. The length 1 clauses representing the block's value are modified to account for the time value of money. Then, we relax the precedence constraints to say that a block satisfies precedence if the above block is mined in either the same or any earlier period. For resource constraints, we add pseudo-Boolean (PB) constraints which are of the form

$$\sum w_i l_i \leq k$$

where w_i is a positive integer weight (in this case tonnage of the block), l_i is the Boolean variable, and k is a positive integer which is the upper limit. A lower limit can be similarly encoded. This gives us the following Max-SAT formulation of the block scheduling problem.

$$\begin{aligned}
\phi_{BSP} = & \bigwedge_{i \in BM} \left(X_{i,t}, DEBV_{i,t}\right) \\
& \bigwedge_{i \in BM} \bigwedge_{t \in T} \left(\neg X_{i,t} \vee X_{j,1} \vee \ldots \vee X_{j,t}\right) \\
& \bigwedge_{i \in BM} \bigwedge_{j=1}^{|T|-1} \bigwedge_{k=j+1}^{|T|} \left(\neg X_{i,j} \vee \neg X_{i,k}\right) \\
& \sum_{i \in BM} w_i X_{i,t} \leq RC_t \qquad \forall t \in T
\end{aligned}$$

where RC_t is the maximum tonnage mined in a given time period. This is a very simplistic formulation of the block scheduling problem, with only the precedence constraints and an overall mining constraint—but we could consider several extensions. Some standard attributes of a block scheduling problem, including blending, stockpiling, and managing head grades, are not so easy to consider in a pure maximum satisfiability model. We should do more research to develop appropriate encodings.

Note that there are many techniques for encoding pseudo-Boolean constraints in SAT. If we say that all blocks have a constant tonnage, we can use cardinality constraints of which there are several described in Frisch (2010) [19]. If the tonnages are different on a block basis, we can use another technique such as those described in Eén (2006) or the generalized totalizer encoding from Joshi (2015) [21, 22].

Again, we are not content with the above encoding and should add the mining width constraint as in the ultimate pit problem. Pourrahimian 2009 provides two mixed integer linear programming formulations for the block scheduling problem that contain considerations for a minimum mining width [23]. In the first, each block is constrained to have a certain number of its neighbors extracted in the same period, and in the second, they aggregate the blocks before the optimization. Neither of these considers minimum mining width structuring elements the way that we are doing here.

We can add the mining width constraint accounting for the minimum mining width-structuring element as described above; however, we have a few options. We could say that a mining width variable is true if the blocks within the structuring element are mined in any period. This effectively ensures that the final pit limits satisfy a minimum mining width. We could also say that a block only satisfies the mining width constraint if all the blocks are mined in the same period. This ensures a minimum pushback width as shown in Fig. 5.

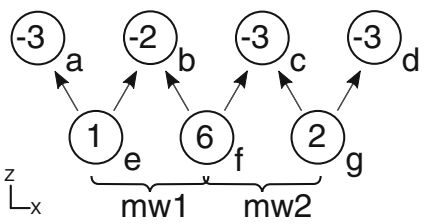

Fig. 4 Minimum example of the ultimate pit problem with a minimum mining width encoded in maximum satisfiability

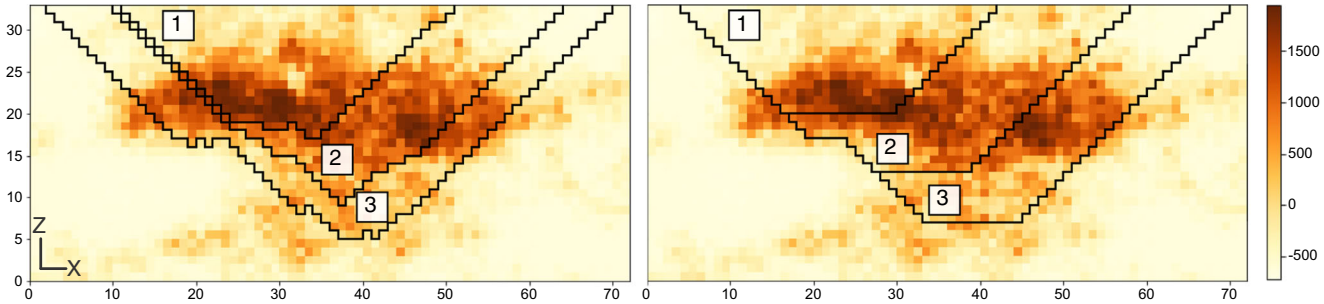

Fig. 5 Block scheduling problem. Left, sequence not considering mining width. Right, the sequence considering a minimum mining width/ pushback width of 12 blocks

This avoids a common problem in block scheduling where algorithms will greedily move the pit wall for a very short distance.

5 Grade Control Polygon Problem

The last problem addressed is the problem of creating optimal, and minable, polygons in open-pit grade control. Grade control in open-pit mines is a large process with many different aspects including sampling, estimating, classifying, and mining material. The grade control polygon problem only covers the classification part of grade control, in that we are given a block model which represents the area of interest with geologic attributes already sampled and estimated. We must then classify the material so that we maximize the economic value and ensure that the resulting classification is practical and could be extracted by reasonably sized equipment.

The grade control polygon problem has been expressed in a variety of ways and solved with a variety of algorithms including those based on metaheuristics such as simulated annealing [24, 25], genetic algorithms [26], or hierarchical techniques [27], but also exact techniques such as branch and bound [28]. An overview of the problem is shown in Fig. 6.

In the grade control polygon problem, we consider the Boolean variable $X_{i,\ c}$ with one for each block $i \in \mathrm{BM}$ and each classification $c \in C$. Again, we add hard clauses to indicate that each block has at most a single classification, but we also add a clause to enforce that each block has one classification. In the block scheduling problem, it was possible for the block to not be mined, but that is not an option here. We add length 1 soft clauses to encode the value of the different classifications. We encode the mining width constraint for this problem in the same way as for the previous problems. We add a set of mining width variables, one for each mining width structuring element location and for each classification. A given mining width variable is constrained, by hard clauses, to be true if and only if all the blocks inside of it are sent to the same destination. Then, for every block, we say that there must be at least one mining width set variable that is true. This is summarized below.

$$\phi_{GCPP} = \bigwedge_{i\in BM}\bigwedge_{c\in C}(X_{i,c}, EBV_{i,c})$$
$$\bigwedge_{i\in BM}\Big(\bigvee_{c\in C} X_{i,c}\Big)$$
$$\bigwedge_{i\in BM}\bigwedge_{j=1}^{|C|-1}\bigwedge_{k=j+1}^{|C|}(\neg X_{i,j} \vee \neg X_{i,k})$$
$$\bigwedge_{i\in S_l}(\neg mw_{l,c} \vee X_{i,c})$$
$$\bigwedge_{i\in BM}(\neg X_{i,c} \vee mw_l \vee \ldots \vee mw_n)$$

The above encoding presumes that there is at most about seven different classifications for each block, or else, the binomial clauses to ensure at most one classification per block will become unwieldy, some alternatives are given in [29].

6 Discussion

These encodings are fundamentally quite simple, with reasonably few clauses of similar form, but as the input grows, the problem grows rapidly. Especially for the block scheduling problem and the grade control polygon problem, as the number of time periods and number of destinations grow, the problem gets substantially harder. This is vitally important because the Max-SAT problem is NP-complete, and there are no known polynomial algorithms available.

There are many different techniques for solving Max SAT problems, and it is not immediately clear which of these techniques is most suitable to our mining problems. There are broadly two main classifications of Max-SAT solvers: incomplete and complete. A complete solver if given enough time will give the single exact optimal result along with a certificate of optimality. Incomplete solvers do not guarantee that the

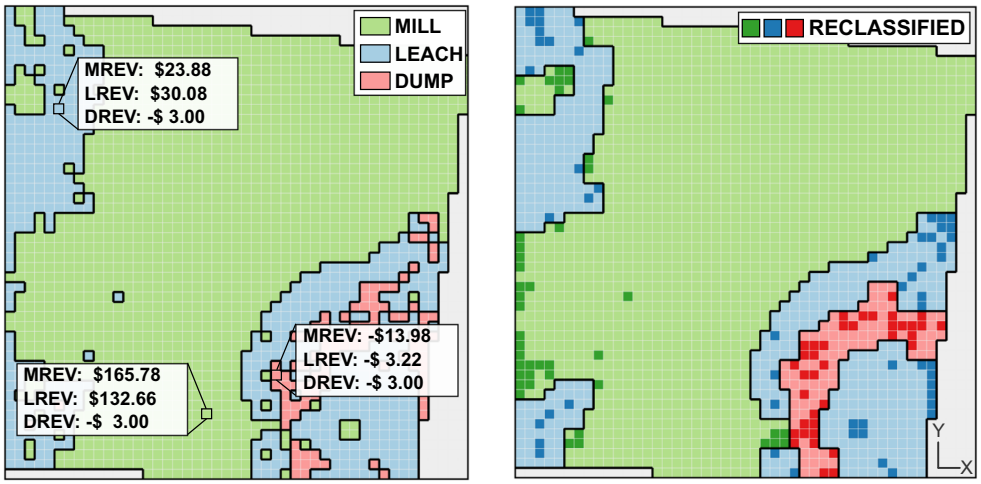

Fig. 6 The grade control polygon problem. Left, input to the grade control polygon problem, economic values for each classification for each block. Right, output of each block assigned to the best classification such that a 3 × 3 mining width is honored and economic value is maximized; reclassified blocks are colored darker

optimal result is found but may give higher quality results quicker or may handle larger problems. There are then many different further classes within these complete and incomplete solvers.

It is interesting to note that Max-SAT problems can be translated into a mixed integer linear program and solved as such. That is, we have opportunities to use sophisticated solvers and features of a mixed integer linear program if it is advantageous.

The pseudo-Boolean constraints that are used in the block scheduling and grade control polygon problems are sometimes directly supported by solvers to further exploit their form. Solvers which support this may do very well on these specific problems. We have not yet performed a full comparison of the many different solution techniques. An initial review has found that many of the complete solvers are not immediately applicable to full problems. They are too slow, or the problems are too big. This is an area for future research.

There is a lot of opportunity for reducing the problem size through appropriate analysis or splitting the problem up into smaller problems that are solved independently. There are opportunities for using combined techniques where heuristics that exploit the structure of the problem are used to provide initial solutions, or to improve intermediate solutions. There could also be better encodings of the problems that have less variables or clauses; however, due to the great sophistication of modern SAT solvers, this may or may not lead to increased performance; the encodings would have to be experimentally validated.

We have examined the grade control polygon problem in detail and have used our encoding extensively on real problems. A commercial implementation of this encoding (with additional improvements) is available and is currently in use at several different mines [28, 30]. The encodings of the ultimate pit problem and the block scheduling problem are still in the early research phase.

7 Conclusion

We have proposed alternate encodings to three fundamental problems in open-pit mine optimization using maximum satisfiability. Maximum satisfiability has proven useful in allowing us to consider more operational constraints and create mine designs that are more realistic. That is, with maximum satisfiability, we can explicitly optimize with a mining width constraint. It is not clear yet whether these encodings are scalable to full-size mining problems, or what sort of work needs to be done—we have provided at least a base framework, and some avenues to explore.

One of the advantages of maximum satisfiability in general is that it is very useful outside of the mining industry. Many other fields use these solvers and are continuously working on them and improving them. Here in the mining industry, we should try to take advantage of this and apply these results to our problems and help us to make better decisions, design better mines, and create more value.

Currently, except for the grade control polygon problem, this introductory work is not suitable for immediate application on full-size realistic problems. Additional research must be done to fully evaluate the different solvers, the different options, and the different optimizations that can be done to see if this work can have real practical significance beyond toy problems.

Compliance with Ethical Standards

Conflict of Interest The author declares that there is no conflict of interest.

References

1. Lerchs H, Grossmann I (1965) Optimum design of open-pit mines. Oper Res 12:B59
2. Gilbert JW (1966) A mathematical model for the optimal design of open pit mines (doctoral dissertation). University of Toronto

3. Lipkewich MP, Borgman L (1969) Two-and three-dimensional pit design optimization techniques. A decade of digital computing in the mineral industry. pp 505–523
4. Chen T (1976) 3d pit design with variable wall slope capabilities. Proc. of the 14th APCOM
5. Khalokakaie R, Dowd PA, Fowell RJ (2000) Lerchs–grossmann algorithm with variable slope angles. Min Technol 109(2):77–85
6. Caccetta L, Giannini L (1988) Generation of minimum search patterns in the optimum design of open pit mines. AusIMM Bull Proc 293:57–61
7. Tarski A (1956) The concept of truth in formalized languages. Logic Semant Metamath 2:152–278
8. Shannon CE (1938) A symbolic analysis of relay and switching circuits. Electr Eng 57(12):713–723
9. Davis M, Putnam H (1958) Computational methods in the propositional calculus. Rensselaer Polytechnic Institute, p 64
10. Davis M, Logemann G, Loveland D (1962) A machine program for theorem-proving. Commun ACM 5(7):394–397
11. Franco J, Martin J (2009) A history of satisfiability. In: Biere, Armin, Heule M, and van Maaren H (eds) Handbook of Satisfiability 185:3–74
12. Tseitin G (1968) On the complexity of derivation in propositional calculus. Studies in constructive mathematics and mathematical logic, pp 115–125
13. Marques-Silva J (2008). Practical applications of boolean satisfiability. In: 2008 9th International Workshop on Discrete Event Systems. IEEE, pp 74–80
14. Berg J, Hyttinen A, Järvisalo M (2015) Applications of MaxSAT in data analysis. Pragmatics of SAT
15. Johnson TB (1968) Optimum open pit mine production scheduling (tech. rep.). California Univ Berkeley Operations Research Center
16. Goldberg AV, Tarjan RE (1988) A new approach to the maximum-flow problem. J ACM (JACM) 35(4):921–940
17. Hochbaum DS (2001) A new-old algorithm for minimum-cut and maximum-flow in closure graphs. Networks 37(4):171–193
18. Deutsch M, Gonzalez E, Williams M (2015) Using simulation to quantify uncertainty in ultimate-pit limits and inform infrastructure placement. Min Eng 67(12):49–55
19. Dagdelen K, Johnson T (1986) Optimum open pit mine production scheduling by Lagrangian parameterization. Proc. of the 19th APCOM, pp 127–142
20. Newman AM, Rubio E, Caro R, Weintraub A, Eurek K (2010) A review of operations research in mine planning. Interfaces 40(3): 222–245
21. Eén N, Sörensson N (2006) Translating pseudo-boolean constraints into sat. JSAT 2:1–26
22. Joshi S, Martins R, Manquinho V (2015) Generalized totalizer encoding for pseudo-boolean constraints. In: International conference on principles and practice of constraint programming. pp 200–209
23. Pourrahimian Y, Askari-Nasab H, Tannant, D (2009) Production scheduling with minimum mining width constraints using mathematical programming
24. Isaaks I, Treloar E, Elenbaas T (2014) Optimum dig lines for open pit grade control. In: Proceedings of ninth international mining geology conference. The Australasian Institute of Mining and Metallurgy, pp 425–432
25. Neufeld K, Norrena C, Deustch C (2005) Guide to geostatistical grade control and dig limit determination, vol 1. p 63
26. Ruiseco JR, Williams J, Kumral M (2016) Optimizing ore–waste dig-limits as part of operational mine planning through genetic algorithms. Nat Resour Res 25(4):473–485
27. Tabesh M, Askari-Nasab H (2013) Automatic creation of mining polygons using hierarchical clustering techniques. J Min Sci 49(3): 426–440
28. Deutsch M (2017) A branch and bound algorithm for open pit grade control polygon optimization. Proc. of the 19th APCOM
29. Frisch AM, Giannaros PA (2010) Sat encodings of the at-most-k constraint. Some old, some new, some fast, some slow. In: Proc. of the tenth int. workshop of constraint modelling and reformulation
30. Deutsch M, Kusuma N, Allen L, Godoy M (2019) Implementing optimal grade control polygons at Newmont's mines. Presentation at the 2019 SME Conference, Denver, CO