Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

ANDREAS BLEY
AMBROS M. GLEIXNER
THORSTEN KOCH
STEFAN VIGERSKE

Comparing MIQCP solvers to a specialised algorithm for mine production scheduling

Supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin.

ZIB-Report 09-32 (October 2009)

Comparing MIQCP solvers to a specialised algorithm for mine production scheduling

Andreas Bley[1], Ambros M. Gleixner[2], Thorsten Koch[2], and Stefan Vigerske[3]

[1] Technische Universität Berlin, Straße des 17. Juni 136, 10623 Berlin,
bley@math.tu-berlin.de
[2] Zuse Institute Berlin, Takustraße 7, 14195 Berlin, {gleixner,koch}@zib.de
[3] Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin,
stefan@math.hu-berlin.de

Abstract. This paper investigates the performance of several out-of-the-box solvers for mixed-integer quadratically constrained programmes (MIQCPs) on an open pit mine production scheduling problem with mixing constraints. We compare the solvers BARON, COUENNE, SBB, and SCIP to a problem-specific algorithm on two different MIQCP formulations. The computational results presented show that general-purpose solvers with no particular knowledge of problem structure are able to nearly match the performance of a hand-crafted algorithm.

1 Introduction

The rise of general mixed-integer programming (MIP) solvers over the last decade [6] is a prime example of how mathematics is able to derive general insights from specific findings and then apply these to other problems without having to look at their inner workings anymore. While in the 90s it was common to write your own branch-and-bound framework employing specialised problem-specific routines, see e.g. [14,17], today it is often very hard to outperform sophisticated state-of-the-art MIP solvers like CPLEX [16], GUROBI [15], CBC [11], or SCIP [2].

Effective general-purpose techniques are currently applicable for most linear mixed-integer and continuous convex optimisation problems. In contrast, for many nonconvex optimisation problems, specialised algorithms are still required to find globally optimal solutions.

Traditional solution methods for nonconvex integer optimisation problems have been developed either as entirely new solvers [22,19], or by directly extending a solver for NLPs to cope with integrality conditions, see e.g. [3,10]. In recent years several groups have started to explore a different direction by trying to extend MIP solvers to handle nonlinearities, see e.g. [1,4,5,9,16].

In this paper we compare the performance of a specialised branch-and-bound code to solve an open-pit mine production scheduling problem with mixing constraints to the performance of several general-purpose solvers on these problems, specifically BARON [22], COUENNE [4], SBB [3], and SCIP [5].

Open-pit mine production scheduling has been chosen as a test case, since the authors were involved in a research project to solve these challenging, large-scale optimisation problems [7]. Now a few years later it can be seen that using recent general-purpose software we are able to get nearly as good solutions out-of-the-box.

In Section 2 we describe our model of the open-pit mine production scheduling problem (OPMPSP). For a more thorough discussion of this application see e.g. [20,12,8]. Section 3 provides two different MIQCP formulations for the OPMPSP. In Section 4 we present the details and results of our computational study.

2 Open pit mine production scheduling with stockpiles

In this section we describe in detail our model of the *open pit mine production scheduling problem* (OPMPSP). Typically, the orebody of an open pit mine is discretised into small mining units called *blocks*. Block models of real-world open pit mines may consist of hundreds of thousands of blocks resulting in large-scale optimisation problems. To overcome this difficulty, groups of blocks are often aggregated to form larger mining units with possibly heterogeneous ore distribution, which we call *aggregates*. We assume such an aggregation of a block model is given a priori, with the set of aggregate indices $\mathcal{N} = \{1, \dots, N\}$.[4] Note that this setting comprises the special case of an unaggregated block model where we have only one block per aggregate.

Moreover, we assume complete knowledge about the contents of each aggregate i: First, its *rock* tonnage R_i, i.e. the amount of material which has to be extracted from the mine. Second, its *ore* tonnage O_i, i.e. the fraction of the rock tonnage sufficiently valuable to be processed further; in contrast, the non-ore fraction of each aggregate is discarded as waste immediately after its extraction from the mine. Finally, the tonnages $A_i^1, \dots, A_i^K$ quantify a number of mineral *attributes* contained in the ore fraction. Attributes may be desirable, such as valuable mineral, or undesirable, such as chemical impurities.

The mining operations consist of several processes: First, rock is extracted from the pit, which we refer to as *mining*. Subsequently, the valuable part of the extracted material is refined further for sale, which is called *processing*; the remaining material not sufficiently valuable is simply discarded as waste. In an intermediate stage between mining and processing, the valuable material may be stored on *stockpiles*. A stockpile can be imagined as "bucket" in which all material is immediately mixed and becomes homogeneous. It may be used for different reasons: blending, storage of excessive mined material, and storage of low grade ore for possible future processing.

The lifespan of the mine is discretised into several, not necessarily homogeneous periods $1, \dots, T$. A feasible mine schedule determines, for each time

[4] Various techniques exist for computing aggregates of blocks, for an example see the fundamental tree method [21].

$N, \mathcal{N}$	number of aggregates and set of aggregate indices $\{1, \dots, N\}$, respectively
$\mathcal{P}(i)$	set of immediate predecessors of aggregate i
R_i, O_i	rock and ore tonnage of aggregate i, respectively [tonnes]
A_i^k	tonnage of attribute k in aggregate i (A_i for a single attribute) [tonnes]
c^k	sales price of attribute k (c for a single attribute) [\$m/tonne]
m, p	mining and processing cost, respectively [\$m/tonne]
T	number of time periods
δ_t	discount factor for time period t (typically $1/(1+q)^t$ with fixed interest rate $q \geqslant 0$)
M_t, P_t	mining and processing capacity, respectively, for time period t [tonnes]

Table 1: List of notation

period, the amount of rock which is to be mined from each aggregate, the fraction of the mined ore which is to be sent for processing or stockpiled, as well as the amount of ore sent from the stockpiles to the processing plant. The major constraints which must be satisfied are *resource constraints* and *precedence constraints*. Resource constraints restrict the amount of rock which may be mined and the amount of ore which may be processed during each time period t by limits M_t and P_t, respectively. Precedence constraints model the requirement that wall slopes are not too steep, ensuring the safety of the mine. Technically, these constraints demand that, before the mining of aggregate i may be started, a set of predecessor aggregates $\mathcal{P}(i)$ must have been completely mined.

Long-term mining schedules have to be evaluated by their *net present value*: For each time period, we take the return from the processed and sold minerals minus the cost for mining and processing, multiplied by a decreasing discount factor to account for the time value of money. The sales price per tonne of attribute k is denoted by c^k; the mining and processing cost per tonne rock is denoted by m and p, respectively. For homogeneous time periods and constant interest rate $q \geqslant 0$ per time period, the profit made in time period t is multiplied by a factor of $1/(1+q)^t$. In general, discount factors $\delta_1, \dots, \delta_T$ may be arbitrary, decreasing values between 0 and 1. The objective is to find a feasible mine schedule with maximum net present value.

Already without considering stockpiles, open pit mine production scheduling poses an NP-hard optimisation problem, see e.g. [13]. One inherent difficulty is given by the opposing nature of precedence constraints and net present value objective: Whereas we want to mine and process high-value material as early as possible to profit from subsequent interest, the precedence constraints (together with limited mining and processing capacity per time period) just prohibit this since high-value material is typically found in the lower layers of the pit. Furthermore, since heavy aggregation of the block model may decrease the net present value, we ideally want to solve the problem for as highly resolved aggregations, i.e. for as many aggregates, as possible.

This paper focuses on the special case of one attribute – some valuable mineral – and a single stockpile with infinite capacity. The next section gives formulations of this case as mixed-integer quadratically constrained programmes (MIQCP). Although we focus on a special case, it includes the essential problem features. A more general setting comprising multiple attributes, multiple stockpiles, finite stockpiling capacity, starting with non-empty stockpiles, or blending constraints can easily be modelled by minor extensions and modifications. To conclude this section, Table 1 summarises the notation introduced above.

3 MIQCP formulations

In this section we provide *mixed-integer quadratically constrained programming* (MIQCP) formulations of the open pit mine production scheduling problem with one attribute ("metal") and a single, infinite-capacity stockpile, as presented in [7]: an aggregated *"basic" formulation* and an extended *"warehouse" formulation*. These formulations are theoretically equivalent. The results in [7], however, clearly speak in favour of the extended formulation. For the LP relaxation based solvers, this is equally confirmed by our computational study presented in Section 4.

3.1 Basic formulation

To track the various material flows, we define the following continuous decision variables for each aggregate i and time period t:

- $y^{\mathrm{m}}_{i,t} \in [0,1]$ as the fraction of aggregate i mined at time period t,
- $y^{\mathrm{p}}_{i,t} \in [0,1]$ as the fraction of aggregate i mined at time period t and sent immediately for processing,
- $y^{\mathrm{s}}_{i,t} \in [0,1]$ as the fraction of aggregate i mined at time period t and sent to the stockpile,
- o^{s}_t, $a^{\mathrm{s}}_t \geqslant 0$ as the absolute amount of ore respectively metal on the stockpile at time period t, and
- o^{p}_t, $a^{\mathrm{p}}_t \geqslant 0$ as the absolute amount of ore respectively metal sent from the stockpile to the processing plant at time period t.

With this, the net present value of a mine schedule is calculated as

$$NPV(y^{\mathrm{m}}, y^{\mathrm{p}}, o^{\mathrm{p}}, a^{\mathrm{p}}) = \sum_{t=1}^{T} \delta_t \left[c \left(a^{\mathrm{p}}_t + \sum_{i=1}^{N} A_i y^{\mathrm{p}}_{i,t} \right) - p \left(o^{\mathrm{p}}_t + \sum_{i=1}^{N} O_i y^{\mathrm{p}}_{i,t} \right) - m \sum_{i=1}^{N} R_i y^{\mathrm{m}}_{i,t} \right]. \tag{1}$$

In order to model the precedence constraints, we define the binary decision variables

- $x_{i,t} \in \{0,1\}$ as equal to 1 if aggregate i is completely mined within time periods $1, \ldots, t$.

A precedence-feasible extraction sequence is then guaranteed by the two sets of constraints

$$x_{i,t} \leqslant \sum_{\tau=1}^{t} y^{\mathrm{m}}_{i,\tau} \qquad \text{for } i \in \mathcal{N}, t = 1, \ldots, T, \tag{2}$$

$$\sum_{\tau=1}^{t} y^{\mathrm{m}}_{i,\tau} \leqslant x_{j,t} \qquad \text{for } i \in \mathcal{N}, j \in \mathcal{P}(i), t = 1, \ldots, T. \tag{3}$$

Additionally, we may, without altering the set of feasible solutions, require the sequence of binary variables $x_{i,1}, \ldots, x_{i,T}$ to be nondecreasing for each aggregate i:

$$x_{i,t-1} \leqslant x_{i,t} \qquad \text{for } i \in \mathcal{N}, t = 2, \ldots, T. \tag{4}$$

Though redundant from a modelling point of view, these inequalities may help (or hinder) computationally, and have been used in the benchmark algorithm from [7]. Conservation of the mined material is enforced by

$$\sum_{t=1}^{T} y^{\mathrm{m}}_{i,t} \leqslant 1 \qquad \text{for } i \in \mathcal{N}, \text{ and} \tag{5}$$

$$y^{\mathrm{p}}_{i,t} + y^{\mathrm{s}}_{i,t} \leqslant y^{\mathrm{m}}_{i,t} \qquad \text{for } i \in \mathcal{N}, t = 1, \ldots, T, \tag{6}$$

i.e. for each aggregate, the amount sent for processing or to the stockpile during one time period must not exceed the total amount mined. (The difference of $y^{\mathrm{m}}_{i,t} - y^{\mathrm{p}}_{i,t} - y^{\mathrm{s}}_{i,t}$ is discarded as waste.) To model the state of the stockpile, we make

Assumption S *Material sent from the stockpile to the processing plant is removed* at the beginning *of each time period, while material extracted from the pit (and not immediately processed) is sent to the stockpile* at the end of each time period.

Following this assumption, we must not send more material from the stockpile to processing than is available at the end of the previous period:

$$o^{\mathrm{p}}_{t} \leqslant o^{\mathrm{s}}_{t-1} \qquad \text{for } t = 2, \ldots, T, \tag{7}$$

$$a^{\mathrm{p}}_{t} \leqslant a^{\mathrm{s}}_{t-1} \qquad \text{for } t = 2, \ldots, T. \tag{8}$$

If we assume the stockpile to be empty at the start of the mining operations, we have

$$o^{\mathrm{p}}_{1} = a^{\mathrm{p}}_{1} = 0. \tag{9}$$

Starting with a nonempty stockpile requires only small modifications. Now, the book-keeping constraints for the amount of ore on the stockpile read

$$o^{\mathrm{s}}_t = \begin{cases} \sum_{i=1}^N O_i y^{\mathrm{s}}_{i,1} & \text{for } t = 1, \\ o^{\mathrm{s}}_{t-1} - o^{\mathrm{p}}_t + \sum_{i=1}^N O_i y^{\mathrm{s}}_{i,t} & \text{for } t = 2, \dots, T, \end{cases} \tag{10}$$

and analogously for the amount of metal on the stockpile

$$a^{\mathrm{s}}_t = \begin{cases} \sum_{i=1}^N A_i y^{\mathrm{s}}_{i,1} & \text{for } t = 1, \\ a^{\mathrm{s}}_{t-1} - a^{\mathrm{p}}_t + \sum_{i=1}^N A_i y^{\mathrm{s}}_{i,t} & \text{for } t = 2, \dots, T. \end{cases} \tag{11}$$

The resource constraints, limiting mining and processing capacity for each time period, read

$$\sum_{i=1}^N R_i y^{\mathrm{m}}_{i,t} \leqslant M_t \qquad \text{for } t = 1, \dots, T, \text{ and} \tag{12}$$

$$o^{\mathrm{p}}_t + \sum_{i=1}^N O_i y^{\mathrm{p}}_{i,t} \leqslant P_t \qquad \text{for } t = 1, \dots, T, \tag{13}$$

respectively. Last, we need to ensure that the ore-metal-ratio of the material sent from stockpile to processing equals the ore-metal-ratio in the stockpile itself. Otherwise, only the profitable metal could be sent to processing and for sale while the ore, only causing processing costs, could remain in the stockpile. This involves the nonconvex quadratic mixing constraints

$$\frac{a^{\mathrm{p}}_t}{o^{\mathrm{p}}_t} = \frac{a^{\mathrm{s}}_{t-1}}{o^{\mathrm{s}}_{t-1}} \qquad \text{for } t = 2, \dots, T.$$

To avoid the singularities for zero denominators, we reformulate them as

$$a^{\mathrm{p}}_t o^{\mathrm{s}}_{t-1} = a^{\mathrm{s}}_{t-1} o^{\mathrm{p}}_t \qquad \text{for } t = 2, \dots, T. \tag{14}$$

All in all, we obtain the *basic formulation*

$$\begin{aligned} \max \quad & NPV(y^{\mathrm{m}}, y^{\mathrm{p}}, o^{\mathrm{p}}, a^{\mathrm{p}}) \\ \text{s.t.} \quad & (2) - (14), \\ & x \in \{0,1\}^{N \times T}, \\ & y^{\mathrm{m}}, y^{\mathrm{p}}, y^{\mathrm{s}} \in [0,1]^{N \times T}, \\ & o^{\mathrm{s}}, a^{\mathrm{s}}, o^{\mathrm{p}}, a^{\mathrm{p}} \geqslant 0. \end{aligned} \tag{BF}$$

3.2 Warehouse formulation

In the basic formulation (BF) the material of all aggregates sent from the pit to the stockpile is aggregated into variables o^{s} and a^{s}. Alternatively, we may track the material flows via the stockpile individually. Instead of variables o^{s}, a^{s}, o^{p}, and a^{p}, we then define for each aggregate i and time period t:

$z^{\mathrm{p}}_{i,t} \in [0,1]$ as the fraction of aggregate i sent from stockpile for processing at time period t and

$z^{\mathrm{s}}_{i,t} \in [0,1]$ as the fraction of aggregate i remaining in the stockpile at time period t.

The net present values in terms of these variables is calculated as

$$NPV(y^{\mathrm{m}}, y^{\mathrm{p}}, z^{\mathrm{p}}) = \sum_{t=1}^{T} \delta_t \left[c \sum_{i=1}^{N} A_i \left(y^{\mathrm{p}}_{i,t} + z^{\mathrm{p}}_{i,t} \right) - p \sum_{i=1}^{N} O_i \left(y^{\mathrm{p}}_{i,t} + z^{\mathrm{p}}_{i,t} \right) - m \sum_{i=1}^{N} R_i y^{\mathrm{m}}_{i,t} \right]. \tag{15}$$

Constraints (2) – (6) remain unchanged. Starting with an empty stockpile gives

$$z^{\mathrm{s}}_{i,1} = z^{\mathrm{p}}_{i,1} = 0 \qquad \text{for } i \in \mathcal{N}. \tag{16}$$

Under Assumption S, the stockpile balancing equations read

$$z^{\mathrm{s}}_{i,t-1} + y^{\mathrm{s}}_{i,t-1} = z^{\mathrm{s}}_{i,t} + z^{\mathrm{p}}_{i,t} \qquad \text{for } i \in \mathcal{N}, t = 2, \ldots, T. \tag{17}$$

The resource constraints on mining are the same as (12), the resource constraints on processing become

$$\sum_{i=1}^{N} O_i \left(y^{\mathrm{p}}_{i,t} + z^{\mathrm{p}}_{i,t} \right) \leqslant P_t \qquad \text{for } t = 1, \ldots, T. \tag{18}$$

Instead of the mixing constraints (14), now we demand that for each time period t, the fraction $z^{\mathrm{p}}_{i,t}/z^{\mathrm{s}}_{i,t}$ is equal for each aggregate i. We obtain a better formulation by introducing, for each time period t, a new variable $f_t \in [0,1]$ called *out-fraction*, and requiring that

$$\frac{z^{\mathrm{p}}_{i,t}}{z^{\mathrm{s}}_{i,t} + z^{\mathrm{p}}_{i,t}} = f_t \qquad \text{for } i \in \mathcal{N}, t = 2, \ldots, T.$$

To avoid zero denominators, we reformulate this as

$$z^{\mathrm{p}}_{i,t}(1 - f_t) = z^{\mathrm{s}}_{i,t} f_t \qquad \text{for } i \in \mathcal{N}, t = 2, \ldots, T. \tag{19}$$

This gives the *warehouse formulation*

$$\begin{aligned} \max \quad & NPV(y^{\mathrm{m}}, y^{\mathrm{p}}, z^{\mathrm{p}}) \\ \text{s.t.} \quad & (2) – (6),\ (12),\ (16) – (19), \\ & x \in \{0,1\}^{N \times T}, \\ & y^{\mathrm{m}}, y^{\mathrm{p}}, y^{\mathrm{s}}, z^{\mathrm{p}}, z^{\mathrm{s}} \in [0,1]^{N \times T}, \\ & f \in [0,1]^{T}. \end{aligned} \tag{WF}$$

Note that the basic formulation is an aggregated version of the warehouse formulation, and thus the LP relaxation (obtained by dropping integrality and mixing constraints) is tighter for the warehouse formulation.

3.3 A priori out-fraction discretisation

The application-specific algorithm of Bley et al. [7] is based on the linear MIP relaxation obtained by first dropping all of the quadratic constraints. To tighten this relaxation for the warehouse formulation, they propose a rough a priori discretisation of the out-fractions, which we describe in the following. Choose a series of fixed ratio levels $0 = \phi_0 < \phi_1 < \ldots < \phi_L < \phi_{L+1} = 1$. Let $\Delta_l = \phi_l - \phi_{l-1}$ for $l = 1, \ldots, L+1$. We define auxiliary binary variables

$$u_{t,l} \in \{0,1\} \text{ as equal to 1 if the out-fraction } f_t \geqslant \phi_l,$$

for each time period t and level l. Trivially, we demand

$$u_{t,l} \leqslant u_{t,l-1} \qquad \text{for } l = 2, \ldots, L, t = 1, \ldots, T. \tag{20}$$

Any binary vector $\tilde{u} \in \{0,1\}^{T \times L}$ satisfying (20) corresponds, for each time period t, to an interval $[\alpha^t, \beta^t] = \left[\sum_{l=1}^{L} \Delta_l \tilde{u}_{t,l}, \Delta_1 + \sum_{l=1}^{L} \Delta_{l+1} \tilde{u}_{t,l}\right]$. The constraints

$$\sum_{l=1}^{L} \Delta_l u_{t,l} \leqslant f_t \leqslant \Delta_1 + \sum_{l=1}^{L} \Delta_{l+1} u_{t,l} \qquad \text{for } t = 1, \ldots, T, \tag{21}$$

force variable f_t to be in the interval $[\alpha^t, \beta^t]$ for each time period t. We obtain the same effect for the out-fractions of each aggregate by the constraints

$$(\phi_l - 1) z^{\mathrm{p}}_{i,t} + \phi_l z^{\mathrm{s}}_{i,t} \leqslant \sum_{l'=1}^{l} \Delta_{l'} (1 - u_{t,l'}) \qquad \text{for } i \in \mathcal{N}, l = 1, \ldots, L, t = 1, \ldots, T, \tag{22}$$

$$(1 - \phi_l) z^{\mathrm{p}}_{i,t} - \phi_l z^{\mathrm{s}}_{i,t} \leqslant \sum_{l'=l}^{L} \Delta_{l'+1} u_{t,l'} \qquad \text{for } i \in \mathcal{N}, l = 1, \ldots, L, t = 1, \ldots, T. \tag{23}$$

Constraints (22) force $z^{\mathrm{p}}_{i,t} / (z^{\mathrm{s}}_{i,t} + z^{\mathrm{p}}_{i,t})$ greater than or equal to α^t, while constraints (23) force it less than or equal to β^t.

4 Computational study

4.1 Application-specific benchmark algorithm

As benchmark algorithm we used the application-specific approach described in [7]. It features a branch-and-bound algorithm based on a mixed-integer linear programming relaxation of the problem obtained by dropping the nonlinear mixing constraints, i.e. (14) for the basic and (19) for the warehouse formulation, respectively. A specialised branching scheme is used to force the maximum violation of the nonlinear constraints arbitrarily close to zero. As long as integer variables with fractional values are present, we branch on these to obtain an integer feasible solution. At nodes of the branch-and-bound tree with integer feasible relaxation solution, but violated nonlinear constraints, a specialised spatial branching is performed.

For the basic formulation, if constraint (14) is violated for some time period $t \in \{2, \ldots, T\}$, then – since the current solution is LP optimal – the metal fraction taken out of the stockpile exceeds the ore fraction taken out. Hence, there is a ratio ϕ with

$$\frac{o^{\mathrm{p}}_t}{o^{\mathrm{s}}_{t-1}} < \phi < \frac{a^{\mathrm{p}}_t}{a^{\mathrm{s}}_{t-1}}.$$

From this, two branches are created: one with

$$o^{\mathrm{p}}_t \leqslant \phi o^{\mathrm{s}}_{t-1} \text{ and } a^{\mathrm{p}}_t \leqslant \phi a^{\mathrm{s}}_{t-1},$$

the other branch with

$$o^{\mathrm{p}}_t \geqslant \phi o^{\mathrm{s}}_{t-1} \text{ and } a^{\mathrm{p}}_t \geqslant \phi a^{\mathrm{s}}_{t-1}.$$

In both branches, the current solution is cut off and the possible maximum violation of the mixing constraint for time period t is reduced.

Similarly, for the warehouse formulation, suppose that constraint (19) is violated for some time period $t \in \{1, \ldots, T\}$. Then there exist at least two aggregates $i_1, i_2 \in \mathcal{N}$ with different out-fractions, thus there is a ratio ϕ with

$$\frac{z^{\mathrm{p}}_{i_1,t}}{z^{\mathrm{s}}_{i_1,t} + z^{\mathrm{p}}_{i_1,t}} < \phi < \frac{z^{\mathrm{p}}_{i_2,t}}{z^{\mathrm{s}}_{i_2,t} + z^{\mathrm{p}}_{i_2,t}}.$$

This gives rise to two branches, one with the additional constraints

$$(1-\phi) z^{\mathrm{p}}_{i,t} \leqslant \phi z^{\mathrm{s}}_{i,t} \qquad \text{for } i \in \mathcal{N},$$

forcing the out-fractions of all aggregates in time period t below ϕ, the other branch with constraints

$$(1-\phi) z^{\mathrm{p}}_{i,t} \geqslant \phi z^{\mathrm{s}}_{i,t} \qquad \text{for } i \in \mathcal{N},$$

forcing them above ϕ. Again, in both branches, the current solution is cut off and the possible maximum violation of the mixing constraints for time period t is reduced.

Note especially that the branches are created by adding multiple linear equalities. Such an aggressive branching strategy is usually invalid for general MINLP solvers, since it cuts off feasible solutions. In this special application it is feasible because of our additional knowledge, that the out-fractions of each aggregate must be equal for each time period.

Bley et al. [7] implemented this approach using the state-of-the-art MIP solver CPLEX 11.2.1 with tuned parameter settings. They apply a variable fixation scheme and cutting planes derived from the underlying precedence constrained knapsack structure which has been shown to improve the dual bound for linear mine production scheduling models. To obtain good primal solutions, they make extensive use of CPLEX's rounding heuristics and post-process the integer feasible solutions by adapting mixing ratios heuristically. For further details, see [7]. We used the same implementation in our computational study.

4.2 General-purpose MIQCP solvers

For our computational experiments, we had access to four general-purpose solvers for MIQCPs: BARON, Couenne, SCIP, and SBB. This section gives a brief overview of their algorithmic features.

BARON BARON [22] is a commercial mixed-integer nonlinear programming (MINLP) solver that implements a branch-and-reduce algorithm. The algorithm first reformulates the problem into a factorable form by introducing new auxiliary variables. The reformulated problem contains only sums of univariate functions and products of two variables. Based on knowledge about convex underestimators for univariate functions and products of variables, a linear relaxation of the problem is constructed, yielding a valid dual bound for the problem. Branching on integer variables allows the solver to reduce fractionality of the relaxation solution, while branching on variables in nonconvex terms allows to compute tighter convex underestimators. The algorithmic performance is improved by certain reduction techniques for variable domains and branching variable selection heuristics.

For our experiments, we used BARON 8.1.5 with CPLEX 11.2.1 [16] as LP solver and MINOS 5.51 [18] as NLP solver.

Couenne Couenne [4] is a very recent open source MINLP solver that implements a similar technique to BARON. It is built on top of the MIP solver CBC [11], and therefore has immediate access to a large set of MIP solution techniques, such as cut generators, branching variable selectors, and node selectors.

For our experiments, we used Couenne 0.2 (stable branch, rev. 256) with CBC 2.3 as branch-and-bound framework, CLP 1.10 [11] as LP solver, and the interior point solver Ipopt 3.6 [23] to handle NLPs.

SCIP SCIP [2] is a constraint integer programming solver that is freely available for academic use. The strength of SCIP is its flexibility in incorporating general kinds of constraints into a branch-cut-and-price framework, and its strong techniques for handling mixed-integer linear programs. It has recently been extended to handle quadratic constraints within an LP based branch-and-cut algorithm [5].

For that purpose, linear underestimators for products and squares of variables are used. Further, a propagation method for variable domains and a local search heuristic has been implemented.

For the mining instances considered in this paper, an extension of a relaxation enforced neighborhood search (RENS) heuristic has proved very successful. The heuristic creates a sub-MIQCP problem by fixing integer variables to nonfractional values in the solution of the LP relaxation at some node of the branch-and-bound tree. This sub-MIQCP is then solved by a separate SCIP instance. During the solution of the sub-MIQCP, a QCP solver is used from time to time to search for a feasible solution of the sub-MIQCP with all remaining integer variables fixed.

For our experiments, we used SCIP 1.1.0.11 with CPLEX 11.2.1 as LP solver and IPOPT 3.6 as QCP solver.

SBB SBB [3] is a commercial solver for MINLPs that implements an NLP based branch-and-bound algorithm. Thus, instead of constructing and solving a linear relaxation of the problem, SBB employs a nonlinear relaxation obtained by relaxing integrality requirements in the original MINLP. The solution of the NLP relaxation is used as dual bound for the branch-and-bound algorithm.

Note, however, that the NLP relaxation is usually solved by an NLP solver that guarantees global optimality only for convex problems. Thus, for a nonconvex MINLP, it is possible that SBB erroneously prunes nodes of the branch-and-bound tree in the case where the NLP solver returns a nonglobal optimal solution. Even though the dual bounds reported by SBB for our mining instances cannot be trusted, we still decided to include SBB into our testset, since NLP based branch-and-bound algorithms often obtain very good primal solutions also for nonconvex MINLPs. Therefore, it offers an interesting comparison to the other LP relaxation based approaches.

For our experiments, we used SBB with CONOPT 3.14T [3] and IPOPT 3.6 as NLP solvers. CONOPT implements *sequential linear programming* (SLP) and *sequential quadratic programming* (SQP) algorithms.

4.3 Test instances

Obtaining real-world problem data from open pit mines is difficult, in part due to the high costs involved in gathering them. Public benchmark instances are typically not available. Our industry partner BHP Billiton Pty. Ltd.[5] has provided us with realistic data from two open pit mines.

Data set *Marvin* is based on a block model provided with the Whittle 4X mine planning software[6], originally consisting of 8513 blocks which were aggregated to 85 so-called “panels”, i.e. single layers of blocks without block-to-block

[5] http://www.bhpbilliton.com/

[6] Gemcom Whittle, http://www.gemcomsoftware.com/products/whittle/

precedence relations. The lifespan of this mine, i.e. the time in which the profitable part of the orebody can be fully mined, is 15 years. Each panel has an average of 2.2 immediate predecessor aggregates.

Data set *Dent* is based on the block model of a real-world open pit mine in Western Australia, originally consisting of 96821 blocks which were aggregated to 125 panels. Each panel has an average of 2.0 immediate predecessor aggregates. The lifespan of this mine is 25 years.

The aggregations to panels, the cutoff grades (determining which blocks in each panel are immediately discarded as waste), and precedence relations between the panels were pre-computed by our industry partner. Scheduling periods are time periods of one year each with a discount rate of 10% per year. Realistic values for mining costs and processing profits as well as for mining and processing capacities per year were chosen by our industry partner.

Using this data, we tested the performance of the general-purpose MIQCP solvers from Section 4.2 on the basic formulation (BF) and the warehouse formulation (WF) – the very formulations on which the benchmark algorithm described in Section 4.1 is based. Additionally, we evaluated the effect of using the a priori out-fraction discretisation described in Section 3.3 by testing formulation (WF)+(20–23) with $L = 1$, 2, and 4 levels using SCIP. Table 2 gives an overview over the size of these MIQCP formulations for problem instances *Marvin* and *Dent*.

Formulation	*Marvin*						*Dent*					
	no. variables			no. constraints			no. variables			no. constraints		
	total	bin.	cont.	total	linear	quadr.	total	bin.	cont.	total	linear	quadr.
(BF)	5848	1445	4403	7598	7582	16	12600	3125	9475	15774	15750	24
(WF)	8687	1445	7242	10404	9044	1360	18775	3125	15650	21900	18900	3000
(WF)+(20–23), 1 level	8704	1462	7242	13328	11968	1360	18800	3150	15650	28200	25200	3000
(WF)+(20–23), 2 levels	8721	1479	7242	16235	14875	1360	18825	3175	15650	34475	31475	3000
(WF)+(20–23), 4 levels	8755	1513	7242	22049	20689	1360	18875	3225	15650	47025	44025	3000

Table 2: Size of MIQCPs for instances *Marvin* and *Dent* (before presolving)

4.4 Computational results

Our computational experiments were run single-threaded on an Intel Core2 Extreme CPU X9650 with 3.0 GHz and 8 GB RAM. For each run, we imposed a time limit of 10000 seconds, within which no solver was able to close the optimality gap. We report primal and dual bound and number of nodes processed after one hour and at the end of the time limit.

Solver settings The solver BARON was run with default and tuned settings. In the latter, we limited the time spent in preprocessing to 30 minutes (option `maxpretime 1800`), which by default appeared to be very time-consuming. We also reduced the amount of probing to depth 5 of the branch-and-bound tree (option `PEnd 5`) and a maximum of 50 variables (option `PDo 50`).

Couenne was also run with default and tuned settings. Here, the tuning consists in turning off expensive bound propagation techniques (options `aggressive_fbbt no` and `optimality_bt no`), which appeared to be too time-consuming in our experiments.

For SBB, we generally switched on the option `acceptnonopt`, ensuring that SBB did not prune a node if the NLP subsolver did not conclude optimality or infeasibility of the node's QCP relaxation. Besides default settings, we also tested a tuned version with option `dfsstay 25`. Usually, SBB uses a mix of depth first search and best bound node selection rule. If a node is processed without creating subnodes, e.g. because an integer feasible solution is found, SBB jumps to the branch-and-bound node with best dual bound. With option `dfsstay n`, SBB is forced to continue searching the neighbourhood of this node in a depth first search manner for `n` more nodes before applying the best bound node selection rule. This setting can help to improve previously found primal solutions.

Furthermore, we ran SBB with a setting where the NLP solver Ipopt is invoked when the default NLP solver (CONOPT) concludes infeasibility of a QCP relaxation (option `infeasseq 100 ipopt`). Using this setting, we intended to reduce the number of cases in which a node is erroneously pruned due to a suboptimal solution of the QCP relaxation. However, Ipopt always confirmed the result of CONOPT ("locally infeasible"), thus only reducing the number of nodes which could be processed within the time limit and leading to a larger final gap. Hence, we do not report the detailed results for this setting in our tables.

The results for tuned settings are indicated by '*' in tables and figures. We parenthesised the dual bound and gap for solver SBB, since they might be invalid due to the nonconvexity of the problem.

SCIP was run with one setting only. The extended RENS heuristic was called frequently. The QCP solver was only used inside the RENS heuristic to search for feasible solutions of the sub-MIQCP with all integer variables fixed.

Results for the basic formulation Table 3 shows the performance of the application-specific benchmark algorithm from Section 4.1 and the general-purpose solvers when using the basic formulation. The application-specific algorithm yields the smallest primal-dual gaps among the LP relaxation based solvers, all of which, however, terminate with large dual bounds: Throughout the whole solution process, the gap remains above 57.37% for SCIP, BARON, and Couenne, and 13.71% for the application-specific algorithm, respectively. Among the LP based general-purpose solvers, BARON has the best dual bounds, while it is outperformed by Couenne and SCIP in terms of primal solutions. However, including the benchmark algorithm, all LP based solvers perform rather unsatisfactory on the basic formulation.

In contrast, the tightest dual bounds clearly are obtained by the NLP based approach of solver SBB – although they cannot be trusted. It produces the best primal solution for problem instance *Marvin* and terminates with the smallest gap of 3.25%. For instance *Dent*, however, the best solution found by SBB was 18.1% worse than the best primal solution found by COUENNE, resulting in a final gap larger than for the benchmark algorithm.

Solver	*Marvin*							*Dent*						
	after 3600 seconds			after 10000 seconds				after 3600 seconds			after 10000 seconds			
	primal	dual	nodes	primal	dual	nodes	gap	primal	dual	nodes	primal	dual	nodes	gap
Benchmark	678.2	916.6	249100	678.2	916.4	476456	35.13	47.3	54.0	100500	47.3	53.8	269023	13.71
BARON	240.3	1347.8	15	240.3	1302.9	42	442.29	6.6	108.3	1	6.6	105.7	12	1508.90
BARON*	240.3	1324.0	29	508.0	1134.7	4600	123.37	6.6	104.3	6	6.6	104.3	1999	1487.32
COUENNE	–	1655.6	2	–	1650.0	104	–	48.1	113.9	0	48.1	113.9	2	137.04
COUENNE*	283.9	1645.9	2893	642.6	1636.0	15429	154.58	48.1	113.9	0	48.1	113.4	2955	135.81
SCIP	669.9	1579.1	286700	671.4	1575.1	810908	57.37	45.8	110.1	87700	45.8	109.9	281776	58.31
SBB	676.6	(706.1)	7980	682.7	(705.0)	25266	(3.25)	39.4	(50.2)	500	39.4	(50.2)	1214	(27.53)
SBB*	683.0	(706.0)	8040	685.1	(704.9)	24922	(2.88)	39.4	(50.2)	540	39.4	(50.2)	1220	(27.33)

Table 3: Results for basic formulation (BF)

Results for the warehouse formulation Table 4 shows the results for the warehouse formulation. First note that the LP based approaches perform significantly better on this formulation. The application-specific algorithm shows excellent performance on the warehouse formulation. It produces the best primal solutions and terminates with the smallest primal-dual gaps of 0.02% for instance *Marvin* and 0.33% for instance *Dent*. Nevertheless, the best primal solutions found by the general-purpose solvers are only 0.4% and 0.2% below the solution found by the benchmark algorithm for *Marvin* and *Dent*, respectively.

The best dual bounds from the general-purpose solvers are 1.4% and 0.2% away from the benchmark values for *Marvin* and *Dent*, respectively. Note that this difference is not only due to the handling of the nonlinear constraints. Also, the benchmark algorithm uses knowledge about the underlying precedence constrained knapsack structure of the linear constraints in order to fix binary variables and separate induced cover inequalities. This structure is not directly exploited by the general-purpose solvers.

In contrast to the LP relaxation based solvers, the QCP relaxation based approach of SBB appears to be less dependent on the change in formulation. For the basic formulation, SBB computed a dual bound almost as tight as for the warehouse formulation – even though these bounds cannot be trusted. Notably, the *Dent* instance appears more challenging to SBB than *Marvin*, while for SCIP the situation is reversed. This is probably due to the increased problem size, which affects the solvability of the QCP relaxation in SBB more than the solvability of the LP relaxation in SCIP.

For both instances, SCIP was able to compute better primal solutions than SBB. For instance *Marvin*, SBB produced a solution only slightly worse than SCIP when using the option `dfsstay 25` (see page 13). Here, the forced depth first search after nodes with integer feasible solution appears to function as an improvement heuristic, compensating for SBB's lack of heuristics.

Solver	*Marvin*							*Dent*						
	after 3600 seconds			after 10000 seconds				after 3600 seconds			after 10000 seconds			
	primal	dual	nodes	primal	dual	nodes	gap	primal	dual	nodes	primal	dual	nodes	gap
Benchmark	694.8	695.9	41057	695.0	695.1	115103	0.02	48.8	49.1	7300	48.9	49.0	23401	0.33
BARON	175.1	717.9	7	175.1	717.4	19	309.84	in preprocessing			12.5	50.3	1	303.18
BARON*	372.8	716.2	965	513.2	715.1	4552	39.33	12.5	49.9	281	12.5	49.9	1756	299.79
COUENNE	–	719.5	0	–	719.5	2	–	47.3	50.3	0	47.3	50.3	2	6.43
COUENNE*	681.2	718.5	193	687.6	715.7	1534	4.09	47.3	50.2	2	47.3	50.2	10	6.11
SCIP	691.0	705.4	40100	691.9	704.7	128655	1.81	48.6	49.2	12900	48.6	49.1	57877	1.08
SBB	677.8	(705.9)	8940	689.0	(705.0)	27498	(2.32)	40.2	(50.1)	580	40.2	(50.1)	1546	(24.69)
SBB*	684.3	(705.9)	9020	691.8	(705.0)	27095	(1.92)	40.3	(50.1)	660	40.3	(50.1)	1611	(24.19)

Table 4: Results for warehouse formulation (WF)

Comparison of LP based solvers BARON, Couenne, and SCIP For the basic formulation, the best dual bounds were found by BARON, while for the warehouse formulations SCIP computed tighter bounds. For all formulations, SCIP computed the best primal solutions among the global solvers – all found by the extended RENS heuristic – and terminated with the smallest gaps. BARON spent much time in preprocessing and per node – also with reduced probing – which results in a comparably small number of enumerated nodes. COUENNE, in contrast, spent much time in its primal solution heuristics. A significant amount of this time was used by the underlying NLP solver IPOPT, which seems to have difficulties solving the (nonconvex) QCPs obtained from fixing integer variables in the original formulation.

Figure 1 compares the progress of the primal and dual bounds from the start to the time limit of 10000 seconds for all three solvers. It can be seen that even with tuned settings BARON and COUENNE spent a significant amount of time in presolving, especially for instance *Dent*. BARON found a number of primal solutions in presolving. Since the BARON log files do not show the times when primal solutions were found during presolving, we plot these at the end of presolving. It is notable that for instance *Dent*, BARON found no improved primal solutions after presolve, and COUENNE produced only one primal solution, which also was found before the branch-and-bound started. Throughout, SCIP shows the highest primal bound.

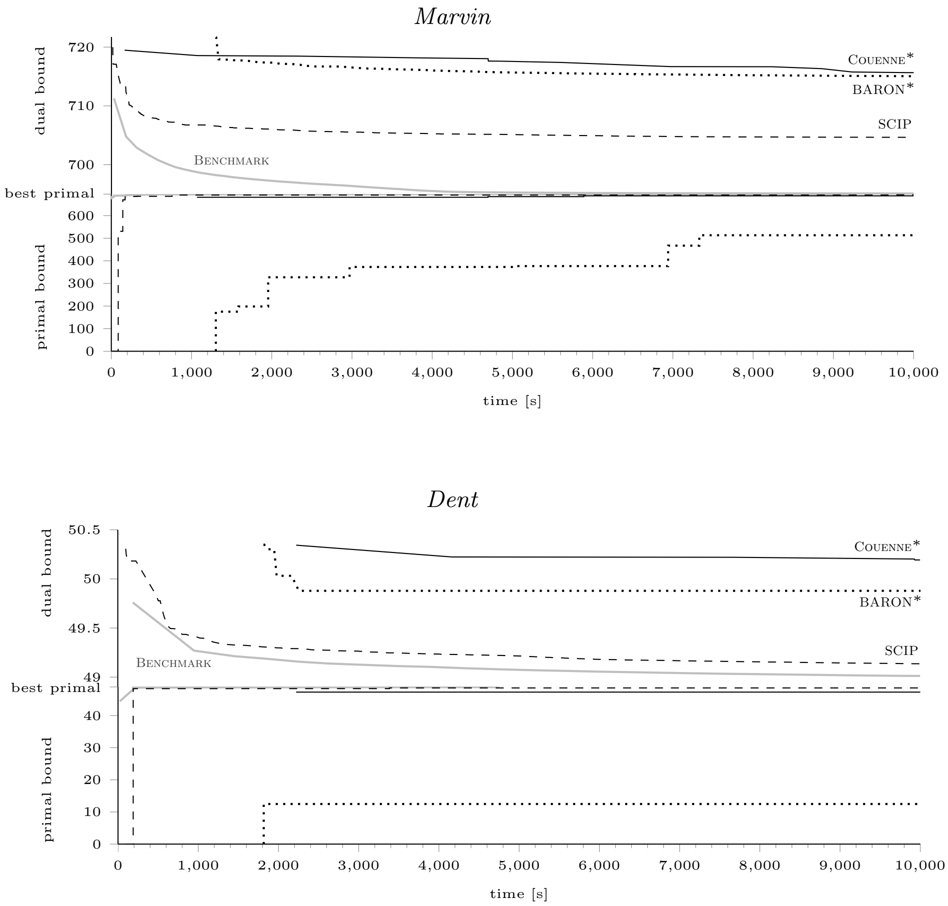

Fig. 1: Progress in primal and dual bounds for the application-specific algorithm (grey) and the global solvers BARON* (dotted), Couenne* (continuous), and SCIP (dashed) for warehouse formulation (WF). The "best primal" axis is level with the best known primal solution value from the application-specific benchmark algorithm. Note the different scales for primal and dual bounds.

For the dual bound it can be seen that all three solvers start approximately with the same dual bound. SCIP, however, is able to decrease it more rapidly and comes closer to the best known primal solution values from the application-specific algorithm. For both instances, SCIP's dual bound after 1800 seconds is already less than 0.35% above its final value at 10000 seconds.

To evaluate whether increasing the time limit could yield significantly better results, we conducted separate experiments running SCIP on both instances for 40 hours. For instance *Marvin*, the primal bound did not improve further and the gap was reduced only slightly from 1.81% to 1.75%. For instance *Dent*, primal and dual bounds improved to yield a final gap of 0.47% compared to a gap of 1.08% after 10000 seconds.

	Marvin							*Dent*						
	after 3600 seconds			after 10000 seconds				after 3600 seconds			after 10000 seconds			
	primal	dual	nodes	primal	dual	nodes	gap	primal	dual	nodes	primal	dual	nodes	gap
no discr.	691.0	705.4	40100	691.9	704.7	128655	1.81	48.6	49.2	12900	48.6	49.1	57877	1.08
1 level	690.1	702.7	35000	690.1	701.2	102190	1.60	48.4	49.2	12500	48.6	49.1	58696	1.03
2 levels	690.2	702.9	19200	690.2	700.8	51332	1.51	48.1	49.3	7800	48.6	49.2	35019	1.31
4 levels	688.1	704.8	12000	688.1	702.5	31009	2.04	47.8	49.3	2800	48.3	49.2	19451	1.84

Table 5: Performance comparison for solver SCIP on warehouse formulation (WF) without and with a priori discretisation constraints (20–23) using 1, 2, and 4 levels.

Experiments with a priori out-fraction discretisation To evaluate the effect of the a priori out-fraction discretisation described in Section 3.3, we conducted experiments running SCIP on the warehouse formulation (WF) with additional a priori discretisation constraints (20–23) for 1, 2, and 4 levels. Results can be seen in Table 5 in comparison to the performance on the plain warehouse formulation (WF).

For both instances, using 1 discretisation level helps to improve the dual bound slightly. On the other hand, for problem instance *Marvin* the best primal solution produced without discretisation was not found anymore. A larger number of discretisation levels seemed to weaken the computational performance and hinder finding primal solutions. The main reason for this appears to be the increased size of the LP relaxation (see Table 2), which grows rapidly with the number of discretisation levels. This leads to more time being consumed by solving the LP relaxation and fewer branch-and-bound nodes being processed within the time limit.

5 Conclusion

We have compared the performance of state-of-the-art generic MIQCP solvers on two realistic instances arising from the scheduling of open pit mine production.

The problem can be characterised as a large mixed-integer linear program which is complemented by a nonlinear part consisting of quadratic mixing constraints.

The performance of SCIP and the application-specific algorithm indicates that for such problems, extending a MIP framework compares favourably to other approaches. Intuitively, the reason might be that integer variables usually model decisions, whereas nonlinear constraints model conditions. Once a linear relaxation of the nonlinear constraints has been solved and all variables are integer feasible, what remains is to fix violations of the nonlinear constraints. One could argue that in many applications this is easier than trying to fix violated integrality constraints once a continuous nonlinear relaxation has been solved.

On the other hand, the performance of the QCP relaxation based solver SBB shows that employing nonlinear relaxations can make the solver more robust with respect to the choice of the formulation used. Unfortunately, as long as there is no way to prove global optimality for the relaxation used, this can only be used as a heuristic. Further, our experiments with SBB indicate that for QCP based branch-and-bound an SLP/SQP based solver outperforms interior point algorithms due to better warmstart capabilities.

Comparing the LP based general purpose solvers, Couenne exploits some sophisticated heuristics in the root node, which enable it to produce good primal solutions. However, the low number of enumerated nodes, partly due to using Ipopt as QCP solver, yields weaker dual bounds. BARON computed better dual bounds, but was unable to produce compatible primal solutions. Our experiments demonstrated that SCIP is able to perform nearly as well as a problem-specific implementation. In a pure MIP setting, SCIP would employ 25 primal heuristics. At the time of testing, only one of these has been extended to handle nonlinearities. As discussed before, solving arising QCPs by an SLP or SQP based solver may improve the performance of SCIP even further.

Acknowledgements We thank our industry partner BHP Billiton Pty. Ltd.[7] for providing us with the necessary data sets to conduct this study, and GAMS Development Corp.[8] for providing us with evaluation licenses for BARON and SBB. Many thanks to Olivia Smith and Jácint Szabó for their thorough proofreading. This research was partially funded by the DFG Research Center Matheon[9], Project B20.

References

1. Kumar Abhishek, Sven Leyffer, and Jeffrey T. Linderoth. FilMINT: An outer-approximation-based solver for nonlinear mixed integer programs. Technical Report ANL/MCS-P1374-0906, Argonne National Laboratory, Mathematics and Computer Science Division, 2006.

[7] http://www.bhpbilliton.com/
[8] http://www.gams.com/
[9] http://www.matheon.de/

2. Tobias Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, 2007.
3. ARKI Consulting & Development A/S. CONOPT and SBB. `http://www.gams.com/solvers/solvers.htm`.
4. P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4-5):597–634, 2009.
5. Timo Berthold, Stefan Heinz, and Stefan Vigerske. Extending a CIP framework to solve MIQCPs. Technical Report 09-23, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), 2009. `http://opus.kobv.de/zib/volltexte/2009/1186/`.
6. Robert E. Bixby, Marc Fenelon, Zonghao Gu, Ed Rothberg, and Roland Wunderling. MIP: Theory and practice – closing the gap. In M. J. D. Powell and S. Scholtes, editors, *System Modelling and Optimization: Methods, Theory and Applications*. Kluwer, 2000.
7. Andreas Bley, Natashia Boland, Gary Froyland, and Mark Zuckerberg. Solving mixed integer nonlinear programming problems for mine production planning with a single stockpile. Technical Report 2009/21, Institute of Mathematics, TU Berlin, 2009.
8. Natashia Boland, Irina Dumitrescu, Gary Froyland, and Ambros M. Gleixner. LP-based disaggregation approaches to solving the open pit mining production scheduling problem with block processing selectivity. *Computers & Operations Research*, 36:1064–1089, 2009. doi:10.1016/j.cor.2007.12.006.
9. Pierre Bonami, Lorenz T. Biegler, Andrew R. Conn, Gérard Cornuéjols, Ignacio E. Grossmann, Carl D. Laird, Jon Lee, Andrea Lodi, François Margot, Nicolas Sawaya, and Andreas Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5:186–204, 2008.
10. Oliver Exler and Klaus Schittkowski. A trust region sqp algorithm for mixed-integer nonlinear programming. *Optimization Letters*, 1:269–280, 2007.
11. John J. Forrest. COIN-OR linear programming and COIN-OR branch and cut. `http://projects.coin-or.org/{Clp,Cbc}/`.
12. Christopher Fricke. *Applications of Integer Programming in Open Pit Mining*. PhD thesis, University of Melbourne, August 2006.
13. Ambros M. Gleixner. Solving large-scale open pit mining production scheduling problems by integer programming. Master's thesis, Technische Universität Berlin, June 2008.
14. Martin Grötschel, Clyde L. Monma, and Mechthild Stoer. Polyhedral and Computational Investigations for Designing Communication Networks with High Survivability Requirements. *Operations Research*, 43(6):1012–1024, 1995.
15. Gurobi Optimization. Gurobi Solver. `http://www.gurobi.com/`.
16. IBM ILOG. CPLEX. `http://www.ilog.com/products/cplex/`.
17. Thorsten Koch and Alexander Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32:207–232, 1998.
18. Bruce A. Murtagh and Michael A. Saunders. *MINOS 5.5 User's Guide*. Department of Operations Research, Stanford University, 1998. Report SOL 83-20R.
19. Ivo Nowak and Stefan Vigerske. LaGO: a (heuristic) branch and cut algorithm for nonconvex MINLPs. *Central European Journal of Operations Research*, 16(2):127–138, 2008.
20. Morteza G. Osanloo, J. Gholamnejad, and Behrooz Karimi. Long-term open pit mine production planning: a review of models and algorithms. *International Journal of Mining, Reclamation and Environment*, 22(1):3–35, 2008.

21. Salih Ramazan. The new fundamental tree algorithm for production scheduling of open pit mines. *European Journal of Operational Research*, 177(2):1153–1166, 2007.
22. Mohit Tawarmalani and Nikolaos V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, 2002.
23. Andreas Wächter and Lorenz T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.