Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 36 (2009) 1064–1089
computers & operations research
www.elsevier.com/locate/cor

LP-based disaggregation approaches to solving the open pit mining production scheduling problem with block processing selectivity

Natashia Boland[a], Irina Dumitrescu[b,*], Gary Froyland[b], Ambros M. Gleixner[c]

[a]*Department of Mathematics and Statistics, The University of Melbourne, Parkville, VIC 3010, Australia*
[b]*School of Mathematics and Statistics, The University of New South Wales, Sydney, NSW 2052, Australia*
[c]*Institut für Mathematik, Technische Universität Berlin, D-10623 Berlin, Germany*

Available online 14 December 2007

Abstract

Given a discretisation of an orebody as a block model, the open pit mining production scheduling problem (OPMPSP) consists of finding the sequence in which the blocks should be removed from the pit, over the lifetime of the mine, such that the net present value (NPV) of the operation is maximised. In practice, due to the large number of blocks and precedence constraints linking them, blocks are typically aggregated to form larger scheduling units. We aim to solve the OPMPSP, formulated as a mixed integer programme (MIP), so that aggregates are used to schedule the mining process, while individual blocks are used for processing decisions. We propose an iterative disaggregation method that refines the aggregates (with respect to processing) up to the point where the refined aggregates defined for processing produce the same optimal solution for the linear programming (LP) relaxation of the MIP as the optimal solution of the LP relaxation with individual block processing. We propose several strategies of creating refined aggregates for the MIP processing, using duality results and exploiting the problem structure. These refined aggregates allow the solution of very large problems in reasonable time with very high solution quality in terms of NPV.

Keywords: Open pit mining; Constrained scheduling problems; Net present value; Aggregation; Iterative disaggregation; Mixed integer programming

1. Introduction

We consider an orebody exploited using open pit mining methods. The orebody is represented as a block model: a discretisation of a volume of earth into blocks. In the presence of mining and processing costs and capacities, as well as precedence constraints with respect to the order in which the blocks can be excavated, the open pit mining production scheduling problem (OPMPSP) consists of finding the sequence in which the blocks should be removed from the pit, over the lifetime of the mine, such that the net present value (NPV) of the operation is maximised. It is intuitively clear that better scheduling decisions can be made when the discretisation of the orebody is very fine, and so current mine planning or modelling tools typically model orebodies using up to millions of blocks. While such block models provide a high resolution description of the orebody, the huge number of blocks makes the OPMPSP very difficult to solve.

* Corresponding author. Fax: +61 2 9385 7123.
E-mail address: irina.dumitrescu@unsw.edu.au (I. Dumitrescu).

0305-0548/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2007.12.006

The OPMPSP can be seen as belonging to the general class of constrained scheduling problems, closely related to a number of more general scheduling problems like the precedence constrained knapsack problem, the simple assembly line balancing problem, the labour constrained scheduling problem and the resource constrained project scheduling problem.

1.1. Approaches to solving OPMPSP

The main approaches applied to solving the OPMPSP are heuristics, dynamic programming and integer programming. A well-known heuristic approach [1] is based on the Lerchs–Grossmann algorithm [2], which is an exact method of determining the "ultimate pit". A series of nested ultimate pits are created by decrementing the sale price of ore from its true value. The extraction schedule is then defined by excavating these nested pits, in order, from smallest to largest pit. Historically, for reasons of computing time, so-called "floating cone" methods (see [3] for example) have been used as heuristic and exact alternatives to the Lerchs–Grossman algorithm, although computing time is not an issue today for these algorithms. A dynamic programming approach to the OPMPSP is described in Onur and Dowd [4]. They noted that a purely dynamic programming approach is unlikely to be able to solve large problem instances. Tolwinski and Underwood [5] combine ideas from dynamic programming with stochastic search heuristics to produce feasible solutions to the OPMPSP for two instances with 88 400 blocks. The hybrid heuristic approach of [5] is introduced in response to the exponential growth in dynamic programming states with the number of blocks in the OPMPSP: "For even a small model where 10 000 blocks need to be considered, the number of states is enormous, and to generate all of them is impracticable" [5].

In recent years, there have been significant improvements to mixed integer programming solvers, prompting further investigation of binary integer and mixed integer approaches to solving the OPMPSP. Ramazan and Dimitrakopoulos [6] describe a mixed integer programme (MIP) formulation and outline an approach for reducing the number of binary variables. The formulation of [6] is typical of recent MIP formulations of the OPMPSP in that it assigns blocks to periods of extraction, rather than determining a strict temporal sequence of blocks. Caccetta and Hill [7] solve a binary integer programming (BIP) formulation with an efficient branching scheme and briefly outline some aspects of a specialised branch-and-cut algorithm. Block models up to 210,000 blocks were scheduled over 10 time periods (in the 210,000 block case, a solution within 8.4% of optimality was found after 20 h). The formulations described in [6,7] are reflective of many MIP-based approaches now used in mine planning software, for example, MineMax Scheduler[8] and Gemcom Whittle [9]. Boland et al. [10] extend the BIP formulation of [7] to include more than two ore attributes and develop knapsack cover inequalities that significantly decrease the computational requirements to obtain the optimal integer solution. Finally, Fricke [11] describes a MIP formulation, using ideas from the formulation of [12], which is a generalisation of the BIP formulation of [7,10]. The computational experiments conducted by Fricke show that this generalisation is particularly effective. Similar formulations are now also being used in mine planning software [13]. We therefore use this class of MIP formulation as a starting point for our analysis.

The new formulation we propose in Section 2 has two crucial improvements over the work of [6,7,10,11]. Firstly, we model excavation and processing decisions at different spatial resolutions. This leads to an extremely efficient and flexible approach for making processing decisions at a high spatial resolution, and is discussed in greater detail in Section 1.3. Secondly, the separation of processing decisions from excavation decisions is in itself an important improvement. By including variables that control whether material should be processed or wasted, in mine scheduling parlance we "simultaneously optimise the *cut-off grade*[1] and the extraction schedule". Often, cut-off grade is determined outside the optimisation process, with a decision to process or waste each block made pre-optimisation (eg. [6,7]). Our MIP formulation is more flexible as the process/waste decision is part of the optimisation procedure. An alternative approach to optimising cut-off grade using binary decision variables is described in Menabde et al. [12].

1.2. Aggregation methods for solving OPMPSP

While there have been considerable advances in MIP formulations of OPMPSP, the sheer scale of the problem renders direct solution of OPMPSP as a MIP impractical. A standard method used to tackle large-scale problems

[1] The *grade* of a block is a dimensionless quantity defined as the ratio of saleable metal to rock contained within the block, and the *cut-off grade* is the value of grade above which blocks are to be processed.

is *aggregation* [14,15]. Aggregation methods can be applied to reduce the number of variables, the number of constraints, or both. Variables or constraints that are determined to be "similar" according to some criteria are grouped together into new variables or constraints called *aggregates*. The problems are then solved with the new variables and constraints. Clearly, the effect of variable aggregation is that some decisions lose their freedom in the aggregated model. By disaggregating, i.e. going back to original variables, a solution for the initial problem is obtained. However, the solution is usually not optimal and sometimes it can even be infeasible.

Aggregation for MIPs has received much less attention than aggregation for linear programmes (LPs). In the case of large-scale LPs the focus in the literature has been on aggregation and on control of the error introduced by aggregation. Many aggregation and disaggregation strategies have been proposed, although in most cases they are one-off procedures. Iterative aggregation and disaggregation have also been discussed in the literature (for a review on this topic see [14]). We will give more details about this later in the paper. In most cases constraint aggregation is the method of choice for MIP formulations. In some cases this is used to derive new strong constraints for the large MIP [16], while in others it is used together with disaggregation (see for example [17]). In the present paper we link the two worlds, by incorporating an LP disaggregation strategy into our MIP approach.

Possibly the first aggregation-based method for solving the OPMPSP is due to Smith [18]. Smith studied a MIP formulation of a variant of the OPMPSP, in which some stockpiling issues were considered. Smith remarked that the problems were too hard to solve directly and proposed some strategies to reduce the number of binary variables in the formulation, one of which was grouping some blocks together into larger blocks. In more recent work, Ramazan et al. [19] and Ramazan [20] propose handling the large number of integer variables by means of aggregation. Maximal groups of blocks are created by solving a linear programme so that any group of blocks has a positive value in total and the blocks forming the boundary of the aggregate obey the slope constraints. The MIP from [6] is then solved for these groups of blocks. We emphasise that in [18–20], once the aggregates are determined *no disaggregation* is performed and both the extraction and the processing decisions are made at the level of aggregates. By contrast, we will make processing decisions at a higher spatial resolution via iterative disaggregation.

1.3. Disaggregation of MIP solutions of an aggregated OPMPSP

While several methods of aggregating blocks have been proposed for OPMPSP, no general technique for partial or full *disaggregation* has been developed. In the approaches described above, the mining and the processing decisions are made at the level of aggregates. The present paper fills this gap by providing a general disaggregation technique for processing decisions. Briefly, this works as follows. We assume that some initial aggregates are given; they can be constructed by any of the methods discussed so far or any other available technique. We propose to continue to control the mining at aggregate level, but to allow the processing to be performed at block level. This is equivalent to full disaggregation for processing decisions.

We will propose an efficient exact iterative disaggregation method for the LP relaxation of the MIP formulation of OPMPSP and demonstrate that it is not necessary to have a decision variable associated with each block to achieve block level processing selectivity. Rather, having variables associated with special *groups of blocks* is enough to achieve the effect of block level processing for the LP relaxation of the MIP.

We propose several strategies of creating groups of blocks that control the processing decisions in the MIP by using the results of the iterative method and the structure of OPMPSP. We then solve the MIP with the original aggregates for mining decisions and the groups of blocks for processing decisions. We will do this by using the information obtained from the iterative exact algorithm for LP and the structure of the problem.

Our iterative exact disaggregation algorithm for the LP relaxation is similar to generic LP iterative aggregation–disaggregation methods, especially to adaptive clustering methods [14,21]. The adaptive clustering methods have two stages. In the first stage an aggregated problem is solved and in the second stage re-aggregation is performed. This is repeated until the aggregation is optimal, in the sense that the optimal value of the aggregated LP equals the objective value of the initial (disaggregated) problem. While this is similar to our approach, in the case of adaptive clustering, the number of aggregates is fixed *a priori* and during the re-aggregation step the number of aggregates is kept constant. In our approach the number of aggregates will be free to vary. In fact, we will start with a sensible choice of relatively

few aggregates and proceed to disaggregate, allowing the number of aggregates to increase slowly. We will show that in practice the total number of aggregates stays quite small.

We now give an outline of the paper. In Section 2 we propose a mixed integer formulation for OPMPSP with block processing. We discuss the link between its linear programming relaxation and the precedence constrained knapsack problem and show how sets of blocks with certain properties may be grouped together into what we call "bins". In Section 3 we formally introduce the notion of a bin, which will be the new entities controlling the processing. We then show how one can identify an optimal binning for the LP (i.e. a binning for which the LP with bin processing has the same optimal value as the LP with block processing). In Section 4 we propose an exact disaggregation algorithm that starts with a given binning and systematically creates new bins until it arrives to an optimal LP binning. Several strategies for the bin construction are proposed and numerical results are provided for all of them. In Section 5 we show numerically that a large improvement in the optimal objective value can be obtained when moving from processing decisions controlled at aggregate level to decisions controlled at block level. We propose several heuristic approaches to utilising the optimal LP binning identified in Section 4 in the MIP formulation of OPMPSP and perform numerical tests for them all. Our experiments show that the solutions obtained for the MIP with the binnings proposed are just as good as the solution of the MIP with block processing in terms of objective value, while the MIP solution times with our binnings are much smaller than MIP solution times using block level processing decisions. In fact, for large data sets, it is impossible to solve the MIP using processing decisions at the block level, while our approach provides good solutions relatively quickly. We then conclude our paper and list future work.

We test the proposed methods on block models of two different pits, a small and a large one, provided by our research partner BHP Billiton Ltd. The smaller of these pits is based upon the "Marvin" data provided with the Whittle 4X mine planning software package [1], while the larger one is based on geological data from BHP Billiton Ltd. Our industrial partner constructed three aggregations for the smaller pit and one for the larger pit. These aggregations take into account equipment capacity, rock mechanics properties, geological ore and waste boundaries, as well as safety constraints. There are frequently no precedence relationships (see the beginning of the next section for a definition) between blocks within a single aggregate. The numerical testing was done on an AMD Opteron 252 computer at 2.6 GHz, with 1 MB of cache and 6 GB of RAM, using CPLEX 9.1.

2. A mixed integer programming model for the OPMPSP

The OPMPSP takes as its starting point a block model, created via standard geostatistical methods. The block model provides a set of K blocks to be considered for excavation. We will denote by $\mathscr{K} = \{1, \ldots, K\}$ the set of all blocks. For each block, the block model also quantifies each of a set of possible *attributes* in the block, for example, an attribute might be the total material (rock) in the block, or some mineral of interest, or some impurity in the block. In this paper, we present our exposition in terms of only two attributes, indexed by 0 and 1: we use a_k^0 to denote the total tonnes of rock in block k and a_k^1 to denote the total tonnes of a single base metal of interest in block k, for each $k \in \mathscr{K}$. The use of only two attributes is for simplicity of exposition only; the key ideas in this paper extend easily to multiple attributes.

The block model also provides precedence relationships between the blocks. For each block $k \in \mathscr{K}$, we have the set $\mathscr{P}(k) \subseteq \mathscr{K}$ defining the set of blocks that must be mined before extraction of block k can begin, in order for the mining operation to be structurally safe.

In addition to the block model, the OPMPSP also takes as input the *mining capacity* M_t in each time period t, in tonnes of rock extracted, and the *processing capacity* P_t in each time period t, in tonnes of rock processed, where $t = 1, \ldots, T$ indexes the time periods used. The OPMPSP also requires an estimate of the cost per tonne of rock extracted and processed in period t, which we denote by c_t^{mng} and c_t^{proc}, respectively, for each $t = 1, \ldots, T$. It also requires a forecast of the revenue gained per tonne of base metal processed in period t, which we denote by c_t^1 for each $t = 1, \ldots, T$. We assume that the revenue is realised in the period in which the material is processed. All costs and revenues are given in present value terms.

The goal of the OPMPSP problem is to decide for each block, how many tonnes should be extracted in each period and, of those tonnes, how many should be processed, with the remainder sent to waste. The excavation and processing should respect the precedence constraints on mining and the total capacity

constraints on mining and processing, so as to maximise the profit of the mine, over its lifetime, in NPV terms.

Unfortunately, the OPMPSP as just described is practically impossible to solve: typically the number of blocks in a block model number in the hundreds of thousands and sometimes in the millions. The common practice is to aggregate blocks into larger geological units, which we call *aggregates*. An aggregate inherits the properties of the blocks it contains: the quantity of each attribute in the aggregate is simply the sum of the attribute quantities of each block in the aggregate, and one aggregate is in the precedence set of another if any block in the former is in the precedence set of any block in the latter. Thus in practice, open pit mine scheduling is usually scheduling the extraction and processing of aggregates, rather than blocks.

A key point in our work is the observation that aggregation affects the problem in two distinct ways: via precedence and processing selectivity. Recently, aggregation approaches have been developed [19,20] so that aggregates align with blocks in terms of precedence. This means that most of the loss of profitability due to aggregation is caused by the loss of processing selectivity at the block level. Thus we focus our work on addressing a model in which we use aggregates for capturing precedence structure, but allow processing selectivity at the level of individual blocks. Clearly, with a greater processing selectivity one can potentially achieve greater profitability. This approach has an added advantage: binary variables are only needed in the OPMPSP mixed integer linear programming model to capture precedence, so we can work with a model that still keeps a relatively small number of binary variables. With the methods of this paper, we show a dramatic increase in processing selectivity can be achieved with reasonable computational effort, whereas increasing the number of aggregates rapidly leads to intractable problems.

In the section below, we give a mixed integer linear programming model of the OPMPSP problem, which is the focus of our work. We use aggregates for precedence, but allow block level processing selectivity. For a problem with N aggregates, we will use the notation $\mathscr{A}_i \subseteq \mathscr{K}$ to denote the ith aggregate, $i = 1, \ldots, N$, and observe that the aggregates are constructed to be sets of blocks which partition $\mathscr{K}$. We use $i(k)$ to denote the (unique) index of the aggregate containing block $k \in \mathscr{K}$, i.e. $k \in \mathscr{A}_{i(k)}$. For simplicity we will sometimes refer to $\mathscr{A}_i$ as "aggregate i". The total tonnes of rock (attribute 0) in aggregate i can be calculated as $\sum_{k\in\mathscr{A}_i} a_k^0$, and the total tonnes of base metal (attribute 1) as $\sum_{k\in\mathscr{A}_i} a_k^1$. For any aggregate i, $i = 1, \ldots, N$, we denote by $\mathscr{P}(i)$ the set of indices of the aggregates that must be extracted before aggregate i. As mentioned earlier, these can be constructed from the block precedence relations.

For simplicity, throughout this paper we will use the notation $g_{t,k} = c_t^1 a_k^1 - c_t^{\text{proc}} a_k^0$ for each period $t = 1, \ldots, T$ and each block $k \in \mathscr{K}$; the convenience of this term will become apparent later.

2.1. The block mixed integer programme (D-MIP)

We define binary variables $x_{i,t} \in \{0, 1\}$ to indicate whether extraction of aggregate i has begun by or in period t

$$x_{i,t} = \begin{cases} 1, & \text{if excavation of aggregate } i \text{ begins in periods } 1, \ldots, t, \\ 0, & \text{otherwise} \end{cases}$$

and continuous variables $y_{i,t} \in [0, 1]$ to represent the fraction of aggregate i to be excavated in time period t for any $i = 1, \ldots, N$ and $t = 1, \ldots, T$. We also define continuous variables $z_{t,k} \in [0, 1]$ to represent the fraction of block k to be processed in time period t, for any $t = 1, \ldots, T$ and $k = 1, \ldots, K$. We assume that an aggregate is excavated uniformly, in the sense that every block in the aggregate is excavated in the same proportion. In other words, we assume that the fraction of block k excavated in period t is $y_{i(k),t} \in [0, 1]$, for each $k \in \mathscr{K}$ and $t = 1, \ldots, T$. Such an assumption is not uncommon in models of open pit mining projects [13]. We also assume that any material in a block not processed is sent to waste, (there is no stockpiling), so for each aggregate $i = 1, \ldots, N$, block $k \in \mathscr{A}_i$ and time period $t = 1, \ldots, T$, the value $y_{i,t} - z_{t,k}$ represents the fraction of the block k that goes to waste in period t. Now the mixed integer linear programming model, which we call D-MIP as a short form of *Disaggregated MIP* (in reference to the disaggregation of aggregates into blocks for the purpose of processing selectivity), can be written as

follows:

$$\max \quad \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{k\in\mathscr{A}_i}(g_{t,k}z_{t,k} - a_k^0 c_t^{\mathrm{mng}} y_{i,t})$$

$$\text{s.t.} \quad z_{t,k} \leqslant y_{i(k),t}, \quad \forall k \in \mathscr{K},\ t \in \{1,\ldots,T\}, \tag{1}$$

$$\sum_{i=1}^{N}\sum_{k\in\mathscr{A}_i} a_k^0 z_{t,k} \leqslant P_t, \quad \forall t \in \{1,\ldots,T\}, \tag{2}$$

$$\sum_{i=1}^{N}\left(\sum_{k\in\mathscr{A}_i} a_k^0\right) y_{i,t} \leqslant M_t, \quad \forall t \in \{1,\ldots,T\}, \tag{3}$$

$$x_{i,t} \leqslant \sum_{s=1}^{t} y_{j,s}, \quad \forall i \in \{1,\ldots,N\},\ \ j \in \mathscr{P}(i),\ \ t \in \{1,\ldots,T\}, \tag{4}$$

$$\sum_{s=1}^{t} y_{i,s} \leqslant x_{i,t}, \quad \forall i \in \{1,\ldots,N\},\ \ t \in \{1,\ldots,T\}, \tag{5}$$

$$x_{i,t} \leqslant x_{i,t+1}, \quad \forall i \in \{1,\ldots,N\},\ \ t \in \{1,\ldots,T-1\}, \tag{6}$$

$$x_{i,t} \in \{0,1\}, \quad \forall i \in \{1,\ldots,N\},\ \ t \in \{1,\ldots,T\}, \tag{7}$$

$$y_{i,t} \geqslant 0, \quad \forall i \in \{1,\ldots,N\},\ \ t \in \{1,\ldots,T\}, \tag{8}$$

$$z_{t,k} \geqslant 0, \quad \forall t \in \{1,\ldots,T\},\ \ k \in \{1,\ldots,K\}.$$

Inequalities (1) simply ensure that the amount of rock of any block which is processed in any given time period is less than or equal to the amount of rock extracted from that block in the time period considered. Inequalities (2) represent the processing constraints and (3) the mining constraints. Inequalities (4)–(6) ensure that the precedence relationships between aggregates are satisfied.

As discussed earlier, the number of blocks in a block model could number in the millions, and so even solving the LP relaxation of D-MIP is challenging for current MIP solver technology. However, as we will show, its special structure can be exploited, so that the LP can be solved for even large numbers of blocks with reasonable computational effort. We discuss this structure next.

2.2. Special structure in the LP relaxation

It is not hard to see that in the LP relaxation of D-MIP, for each period $t = 1, \ldots, T$, the constraints on the (continuous) $z_{t,k}$ variables take the form of a bounded knapsack problem. The bounded knapsack problem in continuous variables has an especially simple structure, admitting closed form primal and dual solutions. We firstly state these results in terms of our problem, then discuss the implications for its solution. Before we begin, we note that these results hold for the LP relaxation at any node of a branch-and-bound tree for solving D-MIP; we define such an LP relaxation and give results in terms of it.

We use $L, U \subseteq \{1,\ldots,N\} \times \{1,\ldots,T\}$ to represent the index sets for $x_{i,k}$ variables that have been fixed by branching to either their lower bound (zero) or upper bound (one), respectively, at some node of a branch-and-bound tree for solving D-MIP. Now the LP relaxation at this node, which we denote by D-LP(L,U), has the following

form:

$$\max \quad \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{k\in\mathscr{A}_i}(g_{t,k}z_{t,k} - a_k^0 c_t^{\mathrm{mng}} y_{i,t})$$

$$\text{s.t.} \quad 0 \leqslant z_{t,k} \leqslant y_{i(k),t}, \quad \forall k \in \mathscr{K}, \ \ t \in \{1,\ldots,T\}, \tag{9}$$

$$\sum_{i=1}^{N}\sum_{k\in\mathscr{A}_i} a_k^0 z_{t,k} \leqslant P_t, \quad \forall t \in \{1,\ldots,T\}, \tag{10}$$

$$(3)\text{–}(6),\ (8),$$

$$x_{i,t} = 0, \quad \forall (i,t) \in L,$$

$$x_{i,t} = 1, \quad \forall (i,t) \in U, \quad 0 \leqslant x_{i,t} \leqslant 1, \quad \forall (i,t) \in \{1,\ldots,N\}\times\{1,\ldots,T\}\backslash(L\cup U).$$

In the above formulation, only constraints (9) and (10) involve the z variables, and these are separable by time period. This leads to the following result.

Proposition 2.1. *Let $L, U \subseteq \{1,\ldots,N\}\times\{1,\ldots,T\}$ be such that D-LP(L,U) is feasible, and let (x^*, y^*, z^*) denote an optimal solution of D-LP(L,U). Then for each $t = 1,\ldots,T$ and each $i = 1,\ldots,N$, it must be that*

$$z_{t,k}^* = \begin{cases} 0, & \text{if } \dfrac{g_{t,k}}{a_k^0} < v_t, \\ y_{i,t}^*, & \text{if } \dfrac{g_{t,k}}{a_k^0} > v_t \end{cases} \tag{11}$$

for all $k \in \mathscr{A}_i$, where v is a vector of optimal dual multipliers for (10) *in D-LP(L,U). Furthermore, there must exist $\hat{z}$ such that $(x^*, y^*, \hat{z})$ is also optimal for D-LP(L,U), with the property that for any time period $t \in \{1,\ldots,T\}$, there exists an aggregate $i_t \in \{1,\ldots,N\}$ such that the set $\{\hat{z}_{t,k} : k \in \mathscr{A}_i\}$ has at most two distinct elements for all $i \in \{1,\ldots,N\}\backslash\{i_t\}$ and the set $\{\hat{z}_{t,k} : k \in \mathscr{A}_{i_t}\}$ has at most three distinct elements. In other words, for a given time period t, the $\hat{z}_{t,k}$ variables for $k \in \mathscr{A}_i$ take on at most two distinct values for all aggregates i, except in the case of one aggregate, i_t, in which they may take on at most three distinct values.*

Proof. For each time period $t = 1,\ldots,T$, z^* and v any optimal dual multiplier for (10) in D-LP(L,U) must furnish an optimal primal and dual solution, respectively, for the continuous bounded knapsack problem CBKP(y^*):

$$\max \sum_{i=1}^{N}\sum_{k\in\mathscr{A}_i} g_{t,k}z_{t,k}$$

$$\text{s.t.} \sum_{i=1}^{N}\sum_{k\in\mathscr{A}_i} a_k^0 z_{t,k} \leqslant P_t,$$

$$0 \leqslant z_{t,k} \leqslant y_{i,t}^*, \quad \forall i = 1,\ldots,N, \ \ k \in \mathscr{A}_i \tag{12}$$

and obviously any optimal solution z of CBKP(y^*) must yield (x^*, y^*, z) an optimal solution of D-LP(L,U). Then by elementary results for the CBKP, (see for example [22]), (11) must hold. Furthermore, setting $\mathscr{K}_t = \{k \in \mathscr{K} : y_{i(k),t}^* > 0\}$, if the blocks in $\mathscr{K}_t$ are indexed so that $g_{t,1}/a_1^0 \geqslant \cdots \geqslant g_{t,|\mathscr{K}_t|}/a_{|\mathscr{K}_t|}^0$, there exists an optimal solution of (12), $\hat{z}$ say,

given by

$$\hat{z}^*_{t,k} = \begin{cases} 0, & \text{if } k \in \{s+1, \ldots, |\mathscr{K}_t|\}, \\ \dfrac{1}{a^0_s}\left(P_t - \sum_{l=1}^{s-1} a^0_l y^*_{i(l),t}\right), & \text{if } k = s, \\ y^*_{i(k),t}, & \text{if } k \in \{1, \ldots, s-1\}, \end{cases} \tag{13}$$

where s is the index of the block with the property that $\sum_{l=1}^{s-1} a^0_l \leqslant P_t$ and $\sum_{l=1}^{s} a^0_l > P_t$. If we take $i_t = i(s)$ the result follows. □

Remark 2.2. The quantity v_t in (11) is analogous to a cut-off grade (see Section 1.1) in the following sense. If the revenue (net of processing cost) per unit of processing resource consumed by block k (namely $g_{t,k}/a^0_k$ in units of \$/tonnes) is greater than v_t, a decision is made to process block k; if this ratio is less than v_t, block k is not processed.

This proposition suggests a number of ideas for solution of both D-LP, (the LP relaxation of D-MIP, i.e. D-LP(∅,∅)), and D-MIP. It shows that within each time period and each aggregate, most of the z variables will take on the same value; there are never more than three distinct values, and in most cases there are only two. In other words, there must exist in each time period a partition of each aggregate into at most three, and in most cases at most two, sets of blocks, so that in some optimal solution, all z variables for the blocks in a set of the partition have the same value. Clearly, if we could somehow guess the right partition of each aggregate in each time period, then we could *a priori* ask for the z variables to be identical in each set in the partition, without compromising optimality. Substituting out identical z variables would lead to a model with at most three, and in most cases at most two, z variables per aggregate and time period, leading to a much smaller problem.

Of course, we are unlikely to be able to guess the right partition *a priori*. However, we show later in the paper that by an iterative procedure, we are able to discover partitions that correspond to optimal solutions of the D-LP. In principle, this procedure could be used at every node of the branch-and-bound tree for the D-MIP; however we find in Section 5 that this is somewhat unnecessary. We show that further use of the ideas in Proposition 2.1 enable us to use the D-LP solution to create partitions of the aggregates in each time period so that solving the resulting MIP yields very close to optimal solutions for the D-LP. We also find that it is not even necessary to completely solve the D-LP in order to get high quality partitions for use in the MIP.

We call partitions of the aggregates in each time period *bins*; in the next two sections we discuss how to determine if bins are optimal for D-LP and we propose an algorithm that iteratively constructs optimal bins.

3. Using binning to solve the D-LP

We begin by writing down a formulation in which each aggregate in each period is partitioned into sets of blocks. Each such set of blocks is called a *bin*, and the processing variables, i.e. the z variables, are required to take on identical values for all blocks in a bin. For each aggregate $i = 1, \ldots, N$ and each period $t = 1, \ldots, T$, let $B_{i,t}$ denote the number of sets (bins) in the partition of aggregate $\mathscr{A}_i$ in period t. For $b = 1, \ldots, B_{i,t}$ we use $\mathscr{B}_{i,t,b} \subseteq \mathscr{A}_i$ to denote the bth bin in the partition. Thus $\mathscr{A}_i = \bigcup_{b=1}^{B_{i,t}} \mathscr{B}_{i,t,b}$, for all $t = 1, \ldots, T$, where the sets $\mathscr{B}_{i,t,1}, \ldots, \mathscr{B}_{i,t,B_{i,t}}$ are all disjoint. We will use the notation $b_t(k)$ to denote the (unique) bin index which contains block k in period t, i.e. $k \in \mathscr{B}_{i(k),t,b_t(k)}$ for all $k \in \mathscr{K}$. We call the collection of all bins a *binning* and denote it by $\mathbf{B}$, i.e. $\mathbf{B} = \{\mathscr{B}_{i,t,b} : i = 1, \ldots, N, t = 1, \ldots, T, b = 1, \ldots, B_{i,t}\}$.

Next we give a mixed integer programming model for this problem, which we will call the B-MIP, short for *Binning* MIP. The continuous variables $\bar{z}_{i,t,b} \in [0, 1]$ will represent the fraction of the bin b in aggregate i that was excavated and processed in time period t, for each $t = 1, \ldots, T$, $i = 1, \ldots, N$ and $b = 1, \ldots, B_{i,t}$. This variable represents the single value that the original block level process selectivity variables will take over all blocks in the bin. In other words, to get from the D-MIP to the B-MIP for a given binning $\mathbf{B}$, set $z_{t,k} = \bar{z}_{i(k),t,b_t(k)}$ for all $k \in \mathscr{K}$ and $t = 1, \ldots, T$, and use this to substitute out all original z variables. After removing redundant constraints (duplicates of (1) arise), this yields

the following B-MIP model:

$$\max \quad \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{b=1}^{B_{i,t}}\left[\left(\sum_{k\in\mathscr{B}_{i,t,b}} g_{t,k}\right)\bar{z}_{i,t,b} - \left(\sum_{k\in\mathscr{B}_{i,t,b}} a_k^0\right)c_t^{\mathrm{mng}} y_{i,t}\right]$$

$$\text{s.t.} \quad \bar{z}_{i,t,b} \leqslant y_{i,t}, \quad \forall i \in \{1,\ldots,N\},\ \ t \in \{1,\ldots,T\},\ \ b \in \{1,\ldots,B_{i,t}\}, \tag{14}$$

$$\sum_{i=1}^{N}\sum_{b=1}^{B_{i,t}}\left(\sum_{k\in\mathscr{B}_{i,t,b}} a_k^0\right)\bar{z}_{i,t,b} \leqslant P_t, \quad \forall t \in \{1,\ldots,T\}, \tag{15}$$

$$\sum_{i=1}^{N}\sum_{b=1}^{B_{i,t}}\left(\sum_{k\in\mathscr{B}_{i,t,b}} a_k^0\right) y_{i,t} \leqslant M_t, \quad \forall t \in \{1,\ldots,T\}, \tag{16}$$

(4), (5), (6), (7), (8)

$$\bar{z}_{i,t,b} \geqslant 0, \quad \forall i \in \{1,\ldots,N\},\ \ t \in \{1,\ldots,T\},\ \ b \in \{1,\ldots,B_{i,t}\}. \tag{17}$$

When we want to explicitly specify the binning **B**, then we refer to this model as B-MIP(**B**).

We note that if a binning is such that each aggregate contains exactly one bin in each time period, i.e. $\mathscr{B}_{i,t,1} = \mathscr{A}_i$ for all i, t (implicitly $B_{i,t} = 1$ for all i and t), then we call this the *aggregate binning*; this is the fully aggregated model without any block processing selectivity. At the other extreme, we have *block binning*, where each aggregate i contains exactly $|\mathscr{A}_i|$ bins in each time period and so each bin consists of a single block. In this case, we recover the D-MIP.

Now Proposition 2.1 and the discussion following that result imply that there exists a binning **B** with at most three bins per aggregate and period (and in most cases at most two), so that the optimal solution of B-LP(**B**) (the LP relaxation of B-MIP(**B**)) yields the optimal solution of D-LP, and furthermore that there exists a (possibly different) binning **B*** with at most three bins per aggregate and period (and in most cases at most two), so that the optimal solution of B-MIP(**B***) yields the optimal solution of D-MIP.

Our goal in this paper is to show that if we start with a given binning (for example, the aggregate binning) we can systematically create new bins to arrive at a binning **B** for which the optimal solution of B-LP(**B**) yields the optimal solution of D-LP. While we cannot guarantee that this process only creates a small number of bins per aggregate and time period, the results of the previous section suggest this is likely, and our numerical results confirm that in practice this is indeed the case. We will also show how we can then well approximate the solution of D-MIP.

The first step is to establish optimality conditions: when will we know that a binning **B** yields B-LP(**B**) with optimal solution also optimal for D-LP? Our follow-up question is then: what do these optimality conditions suggest should be done in the case where a binning is *not* optimal, and how can we progress towards an optimal binning?

3.1. Optimality conditions for a binning

Given a binning **B**, consider an optimal dual solution of B-LP(**B**), which we denote by (u^*, v^*, Λ), where u^* is the vector of optimal dual values corresponding to constraints (14), v^* is the vector of optimal dual values corresponding to (15) and $\Lambda = (\lambda^1, \ldots, \lambda^5)$ is the vector of all the other optimal dual values corresponding, in order, to (16), (4)–(6) and to the upper bound constraints on the x variables, which are constrained to be no more than 1 in B-LP(**B**).

Since (u^*, v^*, Λ) is optimal, all the dual constraints of B-LP(**B**) are satisfied. The dual constraint corresponding to any $\bar{z}_{i,t,b}$ variable is

$$\sum_{k\in\mathscr{B}_{i,t,b}} (g_{t,k} - a_k^0 v_t^*) - u_{i,t,b}^* \leqslant 0. \tag{18}$$

Similarly, the dual constraint corresponding to any $y_{i,t}$ variable can be written as

$$-\sum_{b=1}^{B_{i,t}}\left(\sum_{k\in\mathscr{B}_{i,t,b}} a_k^0\right)c_t^{\mathrm{mng}}+\sum_{b=1}^{B_{i,t}} u^*_{i,t,b}-\sum_{b=1}^{B_{i,t}}\sum_{k\in\mathscr{B}_{i,t,b}} a_k^0\lambda_t^1+\sum_{s=t}^{T}\sum_{j\in\{1,\ldots,N\},i\in\mathscr{P}(j)}\lambda^2_{i,j,s}-\sum_{s=1}^{T}\lambda^3_{i,s}\leqslant 0. \tag{19}$$

For simplicity, in what follows we will define $R_{i,t}(\Lambda)$ to be the right-hand side of (19), if all terms other than those involving u^* are moved to the right, i.e.

$$R_{i,t}(\Lambda)=\sum_{b=1}^{B_{i,t}}\left(\sum_{k\in\mathscr{B}_{i,t,b}} a_k^0\right)c_t^{\mathrm{mng}}+\sum_{b=1}^{B_{i,t}}\sum_{k\in\mathscr{B}_{i,t,b}} a_k^0\lambda_t^1-\sum_{s=t}^{T}\sum_{j\in\{1,\ldots,N\},i\in\mathscr{P}(j)}\lambda^2_{i,j,s}+\sum_{s=1}^{T}\lambda^3_{i,s}.$$

So (19) is equivalently expressed as

$$\sum_{b=1}^{B_{i,t}} u^*_{i,t,b}\leqslant R_{i,t}(\Lambda). \tag{20}$$

We now show that if for a specified binning **B** the dual of B-LP (**B**) is solved to optimality and we have an optimal solution (u^*, v^*, Λ), then we can construct an optimal solution for the dual of D-LP, provided a certain property holds.

Proposition 3.1. *For a given binning* **B**, *let* (u^*, v^*, Λ) *be an optimal solution of the dual of* B-LP(**B**). *Then there exists* $\alpha\in\mathbb{R}^{T\times K}$ *such that* (α, v^*, Λ) *is an optimal solution for the dual of D-LP* (*where* α *is the vector of duals corresponding to the rightmost inequality of* (9)) *if and only if*

$$\sum_{k\in\mathscr{A}_i}(g_{t,k}-a_k^0 v_t^*)^+\leqslant R_{i,t}(\Lambda),\quad \forall i\in\{1,\ldots,N\},\ \ t\in\{1,\ldots,T\}. \tag{21}$$

Proof. Let $(x^*, y^*, \bar{z}^*)$ be an optimal primal solution of B-LP(**B**) and set $z_{t,k}=\bar{z}^*_{i(k),t,b_t(k)}$ for all $t=1,\ldots,T$ and $k=1,\ldots,K$. Then (x^*, y^*, z) is feasible for D-LP and has the same objective value as B-LP(**B**). If (21) holds, then set $\alpha_{t,k}=(g_{t,k}-\alpha_k^0 v_t^*)^+$ for all $t=1,\ldots,T$ and $k=1,\ldots,K$. The dual constraint in D-LP corresponding to any variable $z_{t,k}$ is

$$g_{t,k}-\alpha_{t,k}-a_k^0 v_t^*\leqslant 0. \tag{22}$$

From the definition of α it follows that (22) is satisfied for any t and k. The dual constraint in D-LP corresponding to any variable $y_{i,t}$ can be written as follows:

$$-\left(\sum_{k\in\mathscr{A}_i} a_k^0 c_t^{\mathrm{mng}}\right)+\sum_{k\in\mathscr{A}_i}\alpha_{t,k}-\left(\sum_{k\in\mathscr{A}_i} a_k^0\right)\lambda_t^1+\sum_{s=t}^{T}\sum_{j\in\{1,\ldots,N\},i\in\mathscr{P}(j)}\lambda^2_{i,j,s}-\sum_{s=1}^{T}\lambda^3_{i,s}\leqslant 0, \tag{23}$$

which is precisely (21) (via (20)). Of course $\alpha\geqslant 0$, from its definition. All other dual constraints for D-LP involve only v^* and Λ, and are identical to those for B-LP(**B**). Thus (α, v^*, Λ) is feasible for the dual of D-LP. Now the objective function of the dual of D-LP is identical to that of the dual of B-LP(**B**), and the latter does not involve u^*. Hence we have found a dual feasible point for D-LP, with objective value identical to that of B-LP(**B**). But (x^*, y^*, z) is feasible for D-LP and has the same objective value as B-LP(**B**). Thus we have found primal and dual feasible points for D-LP with identical value: they must be optimal.

Now, suppose there does exist α such that (α, v^*, Λ) is optimal for the dual of D-LP. Then (α, v^*, Λ) must certainly be dual feasible for D-LP, so it must be that $\alpha_{t,k}\geqslant g_{t,k}-a_k^0 v^* t$ for all t and k, and $\alpha\geqslant 0$, so

$$\alpha_{t,k}\geqslant(g_{t,k}-a_k^0 v_t^*)^+$$

for all t and k. But also for dual feasibility we need

$$\sum_{k\in\mathscr{A}_i}\alpha_{t,k}\leqslant R_{i,t}(\Lambda)$$

for all i and t, so it must be that

$$\sum_{k\in\mathscr{A}_i}(g_{t,k}-a_k^0 v_t^*)^+ \leqslant \sum_{k\in\mathscr{A}_i}\alpha_{t,k}\leqslant R_{i,t}(\Lambda)$$

for all i and t, as required. □

This proposition will certainly help us detect when a binning is, and is not, optimal for D-LP. But in the case it is not optimal, the proposition does not directly help us to point a "finger of blame" at bins (or even blocks) that are "causing" suboptimality, nor does it identify how to refine the binning to move closer to an optimal solution of D-LP. The only thing that is obvious is that a block k in a period t with negative value of $g_{t,k}-a_k^0 v_t^*$ is *not* to blame.

In this and further discussions, we will frequently need to refer to the value $g_{t,k}-a_k^0 v_t^*$ of block k in a period t: we will call this the *quasi-reduced cost* of $z_{t,k}$; sometimes we will simply say that "the block" has the quasi-reduced cost. Intuitively, and in our subsequent development, we will see that if a block has a large quasi-reduced cost, this indicates that it wants to "split up out of" its bin, i.e. that the overall objective might be improved by allowing the fraction of block processed to increase from its current position, where it is perhaps being restrained by the need to process all blocks in its bin in the same amount. Similarly, if it has small (negative of large magnitude) quasi-reduced cost, this indicates that it wants to "split down out of" its bin, i.e. that the overall objective might be improved by allowing the fraction of block processed to decrease from its current position, where it is perhaps being restrained by the other blocks in its bin. Note that positivity or negativity of the quasi-reduced cost is equivalent to the determination of the optimal value $z_{t,k}^*$ as described in (11) (although of course in the present setting, v^* is not necessarily yet dual optimal for D-LP).

Interestingly, we can prove that if the binning is not optimal, and (21) is violated for a particular aggregate i and period t, then there must be at least one bin $b\in\mathscr{A}_i$ that has blocks with both positive and negative quasi-reduced cost in period t.

Proposition 3.2. *For a given binning* **B**, *let* (u^*, v^*, Λ) *be an optimal solution of the dual of* B-LP(**B**), *that does not satisfy* (21) *for some* $i\in\{1,\ldots,N\}$ *and* $t\in\{1,\ldots,T\}$. *Then there exists* $\hat{b}\in\{1,\ldots,B_{i,t}\}$ *such that* $g_{t,k}-a_k^0 v_t^*>0$ *and* $g_{t,k'}-a_{k'}^0 v_t^*<0$ *for some* $k,k'\in\mathscr{B}_{i,t,\hat{b}}$.

Proof. Suppose otherwise, i.e. suppose that

$$\{1,\ldots,B_{i,t}\}=\beta_{i,t}^+\cup\beta_{i,t}^-,$$

where

$$\beta_{i,t}^+=\{b\in\{1,\ldots,B_{i,t}\}: g_{t,k}-a_k^0 v_t^*\geqslant 0,\ \forall k\in\mathscr{B}_{i,t,b}\}$$

and

$$\beta_{i,t}^-=\{b\in\{1,\ldots,B_{i,t}\}: g_{t,k}-a_k^0 v_t^*\leqslant 0,\ \forall k\in\mathscr{B}_{i,t,b}\}.$$

Now set $\alpha_{t,k}=(g_{t,k}-a_k^0 v_t^*)^+$ for all $k\in\mathscr{A}_i$. So $\alpha_{t,k}=(g_{t,k}-a_k^0 v_t^*)$ for all $k\in\mathscr{B}_{i,t,b}$ with $b\in\beta_{i,t}^+$ and $\alpha_{t,k}=0$ for all $k\in\mathscr{B}_{i,t,b}$ with $b\in\beta_{i,t}^-$. Thus

$$\begin{aligned}\sum_{k\in\mathscr{A}_i}\alpha_{t,k}&=\sum_{b=1}^{B_{i,t}}\sum_{k\in\mathscr{B}_{i,t,b}}\alpha_{t,k}\\&=\sum_{b\in\beta_{i,t}^+}\sum_{k\in\mathscr{B}_{i,t,b}}\alpha_{t,k}\\&=\sum_{b\in\beta_{i,t}^+}\sum_{k\in\mathscr{B}_{i,t,b}}(g_{t,k}-a_k^0 v_t^*)\end{aligned}\tag{24}$$

since $b \in \beta^+_{i,t} \cap \beta^-_{i,t}$ only if $g_{t,k} - a^0_k v^*_t = 0$ for all $k \in \mathscr{B}_{i,t,b}$, and hence only if $\alpha_{t,k} = 0$ for all $k \in \mathscr{B}_{i,t,b}$. Now

$$\sum_{k \in \mathscr{B}_{i,t,b}} (g_{t,k} - a^0_k v^*_t) \leqslant u^*_{i,t,b} \tag{25}$$

and

$$0 \leqslant u^*_{i,t,b} \tag{26}$$

for all $b = 1, \ldots, B_{i,t}$ by feasibility of (u^*, v^*, Λ) for the dual of B-LP(**B**). By (24)–(26)

$$\sum_{k \in \mathscr{A}_i} \alpha_{t,k} \leqslant \sum_{b \in \beta^+_{i,t}} u^*_{i,t,b} \leqslant \sum_{b=1}^{B_{i,t}} u^*_{i,t,b} \leqslant R^*_{i,t}$$

with the final inequality following by feasibility of (u^*, v^*, Λ) for the dual of B-LP(**B**) and (20). But this means that (21) *is* satisfied for this i and t, contradicting our assumption. □

Although this proposition certainly suggests that refining bins by dividing those with both positive and negative quasi-reduced cost blocks may be helpful, it is hardly yet a "smoking gun". In what follows, we seek more direct evidence for bins "causing" suboptimality. To do so, we establish a relationship between (i) the binning model, which is effectively modelling the requirement that sets of variables (those corresponding to blocks in the same bin) must take on the *same* value, and (ii) traditional revised simplex ideas, namely a model in which some variables are required to be set to *zero*. We develop these ideas next.

3.2. A new bin-based block model

If we allow some blocks to be processed differently from the rest of the bin and denote by $\delta_{t,k}$ the fractional processing deviation for block k from the bin processing fraction $\bar{z}_{i,t,b}$, $k \in \mathscr{B}_{i,t,b}$, a valid block processing model can be constructed from the bin model as follows:

$$\max \quad \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{b=1}^{B_{i,t}} \left[\left(\sum_{k \in \mathscr{B}_{i,t,b}} g_{t,k} \right) \bar{z}_{i,t,b} - \left(\sum_{k \in \mathscr{B}_{i,t,b}} a^0_k \right) c^{\text{mng}}_t y_{i,t} \right] + \sum_{t=1}^{T} \sum_{\substack{k \in \{1,\ldots,K\} \\ \text{s.t. } |\mathscr{B}_{i,t,b_t(k)}| > 1}} g_{t,k} \delta_{t,k}$$

$$\text{s.t.} \quad \bar{z}_{i,t,b} + \delta_{t,k} \leqslant y_{i,t}, \tag{27}$$

$$\forall i \in \{1, \ldots, N\}, \quad t \in \{1, \ldots, T\}, \quad b \in \{1, \ldots, B_{i,t}\} \quad \text{s.t. } |\mathscr{B}_{i,t,b}| > 1, \quad k \in \mathscr{B}_{i,t,b},$$

$$\bar{z}_{i,t,b} \leqslant y_{i,t}, \tag{28}$$

$$\forall i \in \{1, \ldots, N\}, \quad t \in \{1, \ldots, T\} \quad \text{and} \quad b \in \{1, \ldots, B_{i,t}\} \quad \text{s.t. } |\mathscr{B}_{i,t,b}| = 1,$$

$$\sum_{i=1}^{N} \sum_{b=1}^{B_{i,t}} \left(\sum_{k \in \mathscr{B}_{i,t,b}} a^0_k \right) \bar{z}_{i,t,b} + \sum_{\substack{k \in \{1,\ldots,K\} \\ |\mathscr{B}_{i,t,b_t(k)}| > 1}} a^0_k \delta_{t,k} \leqslant P_t, \quad \forall t \in \{1, \ldots, T\}, \tag{29}$$

$$-\bar{z}_{i,t,b} - \delta_{t,k} \leqslant 0, \tag{30}$$

$$\forall i \in \{1, \ldots, N\}, \quad t \in \{1, \ldots, T\}, \quad b \in \{1, \ldots, B_{i,t}\} \quad \text{s.t. } |\mathscr{B}_{i,t,b}| > 1, \quad k \in \mathscr{B}_{i,t,b},$$

$$(4), (5), (6), (7), (8), (16)$$

$$\bar{z}_{i,t,b} \geqslant 0, \quad \forall i \in \{1, \ldots, N\}, \ t \in \{1, \ldots, T\} \quad \text{and} \quad b \in \{1, \ldots, B_{i,t}\} \quad \text{s.t. } |\mathscr{B}_{i,t,b}| = 1,$$

$$\delta_{t,k} \in \mathbb{R}, \quad \forall t \in \{1, \ldots, T\}, \ k \in \{1, \ldots, K\}.$$

In what follows we will refer to this model as δ-MIP. Clearly, if all $\delta_{t,k}$ that appear in this model are fixed to zero, then the model is identical to B-MIP. We now consider the linear programming relaxations of B-MIP and δ-MIP. We will denote them by B-LP, respectively, δ-LP. As in the case of D-LP, the constraints (7) are replaced by a new set of constraints that appears in order to place an upper bound on the x variables. The new set of constraints is $0 \leqslant x_{i,t} \leqslant 1$, for any i and t, in both B-LP and δ-LP.

Our goal is to find a binning **B** so that an optimal solution of D-LP can be obtained directly from an optimal solution of B-LP(**B**). We will determine such a binning by an iterative method. At each step of the method we perform two operations. First, for the current binning, we solve B-LP. We will show that it is easy to construct a solution for δ-LP that is dual feasible with respect to the $\bar{z}$, y and x variables. We then price out some δ variables for which the corresponding dual constraints are violated. Adding these δ variables to the model effectively means modifying some of the bins (those that contained the blocks associated with the priced-out δ variables) and re-optimising B-LP with the modified binning. This is done until the binning obtained satisfies the optimality conditions of Proposition 3.1, therefore an optimal solution of the dual (and implicitly the primal) of D-LP can be constructed from an optimal solution of the dual of B-LP. In the rest of this section we will focus on the relationship between the duals of B-LP and δ-LP (details on our iterative method will be provided in Section 4).

Starting from (u^*, v^*, Λ), as defined at the beginning of Section 3.1, we seek to construct a feasible dual solution of δ-LP. The constraints (14) to which u^* apply in the dual of B-LP have become two sets of constraints in the dual of δ-LP: (27) and (28) (one set for multiblock bins and one for single block bins). Also, additional non-negativity constraints (30) appear in the dual of δ-LP. In our construction of a dual solution for δ-LP we seek to keep the same dual variables for the constraints that are the same in the duals of δ-LP and B-LP, but we need to find dual variables for the new constraints. We will call μ the dual variables corresponding to (27) and (28) and ν those corresponding to (30). We therefore denote by (μ, v^*, Λ, ν) a candidate dual solution for δ-LP. The vector v^* contains the dual variables corresponding to (29) and Λ those corresponding, in order, to the constraints (16), (4)–(6) and to the upper bound constraints on the x variables, which are constrained to be no more than 1.

3.3. Feasibility of δ-LP

Next, we will investigate the conditions under which (μ, v^*, Λ, ν) is feasible for the dual problem of δ-LP. We first observe that μ and ν need to be non-negative vectors. Also, due to the fact that when the bins contain exactly one block the constraints (14) coincide with the constraint (28), we can fix some of the components of μ. We define $\mu_{t,k} = u^*_{i(k),t,b_t(k)}$ for any time period t and block k that forms a bin by itself, i.e. $|\mathscr{B}_{i(k),t,b_t(k)}| = 1$.

The constraints corresponding to the x variables are identical in the duals of B-LP and δ-LP. Since (u^*, v^*, Λ) is optimal for the dual of B-LP, these constraints are satisfied in the dual of B-LP and therefore are satisfied in the dual of δ-LP. We therefore focus on the dual constraints of δ-LP corresponding to the $\bar{z}$, y and δ variables.

3.3.1. Dual constraints of B-LP

Since (u^*, v^*, Λ) is optimal for the dual of B-LP, all the dual constraints of B-LP are satisfied (in particular (18) and (19), the constraints corresponding to $\bar{z}_{i,t,b}$ and $y_{i,t}$).

3.3.2. Dual constraints of δ-LP

Using the dual solution (μ, v^*, Λ, ν) we first write the dual constraints corresponding to the variables $\bar{z}_{i,t,b}$.

From the definition of μ we know that $\mu_{t,k} = u^*_{i,t,b}$ for any i, t and bin b that contains the block k only, i.e. $\mathscr{B}_{i,t,b} = \{k\}$. Therefore for any i, t and b for which $|\mathscr{B}_{i,t,b}| = 1$ we can write the constraints corresponding to the variables $\bar{z}_{i,t,b}$ as follows:

$$\sum_{k \in \mathscr{B}_{i,t,b}} g_{t,k} - u^*_{i,t,b} - \left(\sum_{k \in \mathscr{B}_{i,t,b}} a^0_k \right) v^*_t \leqslant 0. \tag{31}$$

Since (18) and (31) are identical and (18) are satisfied, it follows that (31) are also satisfied.

We now look at the bins that contain more than just one block (i.e. $|\mathscr{B}_{i,t,b}| > 1$). The dual constraints corresponding to the variables $\bar{z}_{i,t,b}$ can be written as follows:

$$\sum_{k\in\mathscr{B}_{i,t,b}} g_{t,k} - \sum_{k\in\mathscr{B}_{i,t,b}} \mu_{t,k} - \left(\sum_{k\in\mathscr{B}_{i,t,b}} a_k^0\right) v_t^* + \sum_{k\in\mathscr{B}_{i,t,b}} v_{t,k} \leqslant 0. \tag{32}$$

Similarly, we write the dual constraints corresponding to the variable $y_{i,t}$ of the δ-LP model. They are

$$-\sum_{b=1}^{B_{i,t}} \left(\sum_{k\in\mathscr{B}_{i,t,b}} a_k^0\right) c_t^{\mathrm{mng}} + \sum_{b=1}^{B_{i,t}} \sum_{k\in\mathscr{B}_{i,t,b}} \mu_{t,k} - \sum_{b=1}^{B_{i,t}} \left(\sum_{k\in\mathscr{B}_{i,t,b}} a_k^0\right) \lambda_t^1 + \sum_{s=t}^{T} \sum_{j\in\{1,\ldots,N\}, i\in\mathscr{P}(j)} \lambda_{i,j,t}^2 - \sum_{s=1}^{T} \lambda_{i,s}^3 \leqslant 0. \tag{33}$$

From (19) it follows that if

$$\sum_{b=1}^{B_{i,t}} \sum_{k\in\mathscr{B}_{i,t,b}} \mu_{t,k} \leqslant \sum_{b=1}^{B_{i,t}} u_{i,t,b}^*, \tag{34}$$

then (33) is satisfied.

We summarise these results in the following lemma, which guarantees that under certain conditions the dual constraints corresponding to $\bar{z}$ and y for δ-LP are satisfied.

Lemma 3.3. *Let* (u^*, v^*, Λ) *be an optimal solution for the dual of* B-LP. *If* μ *and* v *are two non-negative vectors such that for any* i, t *and* b, $|\mathscr{B}_{i,t,b}| > 1$, *we have*

1. $\sum_{k\in\mathscr{B}_{i,t,b}} (\mu_{t,k} - v_{t,k}) \geqslant \sum_{k\in\mathscr{B}_{i,t,b}} (g_{t,k} - a_k^0 v_t^*)$ *and*
2. $\sum_{k\in\mathscr{B}_{i,t,b}} \mu_{t,k} \leqslant u_{i,t,b}^*$,

then the constraints (32) *and* (33) *are satisfied for all* i, t *and* b *such that* $|\mathscr{B}_{i,t,b}| > 1$.

Proof. The result follows directly from the conditions of the lemma, from the fact that (19) is satisfied. □

Lemma 3.4. *Non-negative vectors* μ *and* v *that satisfy the conditions of Lemma* 3.3 *exist*.

Proof. For any $k \in \mathscr{K}$ such that $|\mathscr{B}_{i(k),t,b_t(k)}| > 1$ let $\mu_{t,k} = u_{i(k),t,b_t(k)}^* / |\mathscr{B}_{i(k),t,b_t(k)}|$ and $v_{t,k} = 0$. The vectors μ and v satisfy the conditions of Lemma 3.3. □

We now move on to discuss the dual constraints corresponding to δ variables for δ-LP. For any $t = 1, \ldots, T$ and $k \in \mathscr{K}$ such that $|\mathscr{B}_{i(k),t,b_k(t)}| > 1$ the dual constraint corresponding to $\delta_{t,k}$ is

$$g_{t,k} - a_k^0 v_t^* - \mu_{t,k} + v_{t,k} = 0. \tag{35}$$

In what follows, we will decide whether a constraint (35) is satisfied or not by looking at the amount by which the constraint is violated. This amount is $|g_{t,k} - a_k^0 v_t^* - \mu_{t,k} + v_{t,k}|$. Clearly, if we find some vectors μ and v so that the violation of the constraints (35) is zero across all time periods for all blocks in bins of cardinality at least two, then we know that the dual constraint corresponding to any $\delta_{t,k}$ variable from δ-LP is satisfied. We note that since we want to keep feasibility of the dual constraints corresponding to the $\bar{z}$ and y variables, we will require that the vectors μ and v also satisfy the conditions described in Lemma 3.3. Since adding new $\delta_{t,k}$ variables to the model effectively means changing the bins to which the blocks k belong, we will look at the dual constraints corresponding to the δ variables in batches (bin by bin), so that we can easily identify the bins that need to be modified.

Therefore we check whether or not violation of (35) occurs by defining a subproblem for each aggregate i, time period t and bin b that contains more than one block (i.e. $|\mathscr{B}_{i,t,b}| > 1$). The subproblem attempts to find the components of the vectors μ and v corresponding to t and to the blocks in bin b of $\mathscr{A}_i$, which minimise the violation of (35) over all blocks in the bin, while satisfying (32) and (33). The subproblem can be written as follows:

$$\begin{aligned}
\min_{\mu_{t,k}, v_{t,k}} \quad & \max_{k \in \mathscr{B}_{i,t,b}} |(g_{t,k} - a_k^0 v_t^*) - (\mu_{t,k} - v_{t,k})| \\
\text{s.t.} \quad & \sum_{k \in \mathscr{B}_{i,t,b}} (\mu_{t,k} - v_{t,k}) \geqslant \sum_{k \in \mathscr{B}_{i,t,b}} (g_{t,k} - a_k^0 v_t^*), \\
& \sum_{k \in \mathscr{B}_{i,t,b}} \mu_{t,k} \leqslant u^*_{i,t,b}, \\
& \mu_{t,k} \geqslant 0, \quad \forall k \in \mathscr{B}_{i,t,b}, \\
& v_{t,k} \geqslant 0, \quad \forall k \in \mathscr{B}_{i,t,b}.
\end{aligned} \tag{36}$$

We note that from Lemma 3.4 it follows that the above subproblem is feasible. If the optimal value of (36) is zero, it follows that for the time period, aggregate and bin considered all dual constraints corresponding to the $\delta_{t,k}$ variables, where $k \in \mathscr{B}_{i,t,b}$, are satisfied. By Lemma 3.3, the optimal $\mu_{t,k}$ and $v_{t,k}$ for the subproblem are feasible for the dual of δ-LP. If all subproblems (i.e. for all i, t and b s.t. $|\mathscr{B}_{i,t,b}| > 1$) have optimal value zero, it follows that all dual constraints corresponding to the δ variables are satisfied, and therefore (μ, v^*, Λ, v) with μ and v determined by the subproblems is a feasible solution of the dual δ-LP.

We now describe the necessary and sufficient conditions under which the optimal value of a subproblem is zero.

Lemma 3.5. *Let i, t and b be such that $|\mathscr{B}_{i,t,b}| > 1$ and $K^+_{i,t,b} = \{k \in \mathscr{B}_{i,t,b} : g_{t,k} - a_k^0 v_t^* > 0\}$. The optimal value of subproblem* (36) *is zero if and only if $\sum_{k \in K^+_{i,t,b}} (g_{t,k} - a_k^0 v_t^*) \leqslant u^*_{i,t,b}$.*

Proof. *The If Case*: For the given t, we can define a feasible solution $(\mu_{t,k}, v_{t,k})_{k \in \mathscr{B}_{i,t,b}}$ of (36) as follows:

$$\mu_{t,k} = \begin{cases} g_{t,k} - a_k^0 v_t^*, & k \in K^+_{i,t,b}, \\ 0, & \text{otherwise} \end{cases} \tag{37}$$

and

$$v_{t,k} = \begin{cases} 0, & k \in K^+_{i,t,b}, \\ -g_{t,k} + a_k^0 v_t^*, & \text{otherwise.} \end{cases} \tag{38}$$

First suppose that $K^+_{i,t,b} \neq \emptyset$. Since $\sum_{k \in K^+_{i,t,b}} (g_{t,k} - a_k^0 v_t^*) \leqslant u^*_{i,t,b}$, the solution given by (37) and (38) satisfies all the constraints of problem (36). Also, the value of the objective function for this solution is zero. Since the objective function is non-negative, the solution defined is optimal.

Second, if $K^+_{i,t,b} = \emptyset$, we define $\sum_{k \in K^+_{i,t,b}} (g_{t,k} - a_k^0 v_t^*) = 0$ and the proof follows similarly as above.

The Only If Case: We prove this case by contradiction. We assume that the optimal value of (36) is zero, but that $\sum_{k \in K^+_{i,t,b}} (g_{t,k} - a_k^0 v_t^*) > u^*_{i,t,b}$. We note that this case can only happen if $K^+_{i,t,b} \neq \emptyset$.

For the given t, let $(\mu^*_{t,k}, v^*_{t,k})_{k \in \mathscr{B}_{i,t,b}}$ be an optimal solution of (36). Since the objective value of (36) is zero, it follows that $g_{t,k} - a_k^0 v_t^* - (\mu^*_{t,k} - v^*_{t,k}) = 0$ for any $k \in \mathscr{B}_{i,t,b}$. This equality can also be written as

$$\mu^*_{t,k} - v^*_{t,k} = g_{t,k} - a_k^0 v_t^*. \tag{39}$$

From the definition of the set $K^+_{i,t,b}$ we know that for any $k \in K^+_{i,t,b}$ we have $g_{t,k} - a_k^0 v_t^* > 0$. Since $v^*_{t,k} \geqslant 0$, from (39) it follows that $\mu^*_{t,k} \geqslant g_{t,k} - a_k^0 v_t^*$ for any block $k \in K^+_{i,t,b}$. This implies $\sum_{k \in K^+_{i,t,b}} \mu^*_{t,k} \geqslant \sum_{k \in K^+_{i,t,b}} g_{t,k} -$

$a_k^0 v_t^*$. Since our working assumption is $\sum_{k\in K^+}(g_{t,k} - a_k^0 v_t^*) > u_{i,t,b}^*$, it follows that $\sum_{k\in K_{i,t,b}^+} \mu_{t,k}^* > u_{i,t,b}^*$. Therefore $(\mu_{t,k}^*, v_{t,k}^*)_{k\in\mathscr{B}_{i,t,b}}$ is not a feasible solution of (36), which is a contradiction as $(\mu_{t,k}^*, v_{t,k}^*)_{k\in\mathscr{B}_{i,t,b}}$ is optimal and hence feasible. □

Finally, we note that if the optimal values of all subproblems are zero, then Proposition 3.1 holds for the current binning (this is easy to see from Lemma 3.5 and duality conditions). Next we propose several strategies for modifying the binning if it is not optimal, by changing the bins for which the optimal value of the subproblem (36) is not zero.

4. A generic algorithm for solving D-LP

The previous section provides necessary and sufficient conditions, in terms of certain dual multipliers, under which the optimal solution of B-LP can be used to construct an optimal solution of D-LP. We now tackle the question of how to arrive at a binning **B** for which the optimal solution of the dual of B-LP satisfies the conditions of Proposition 3.1.

Our approach to constructing optimal binnings will be to begin with a very coarse binning, eg. the aggregate binning, and to iteratively refine this binning until the conditions of Proposition 3.1 are satisfied. At this point we have found an optimal binning. Below we describe a generic algorithm to carry out this iterative refinement. We investigate an exact and an approximate method, controlled by a fixed non-negative value ε: the refining is either performed to optimality (in the sense that the optimal solution of the dual of δ-LP with the current binning allows us to construct an optimal dual solution of D-LP), or until no significant improvement between iterations is recorded. The issues of how to algorithmically identify bins for refinement and how to refine them are dealt with shortly.

Algorithm 4.1. *A Generic Algorithm for Solving D-LP* (GA(**B**, ε)).

Step 0: *Initialisation*
  update_flag = TRUE
  best_objective_value $= -\infty$
Step 1: *Binning Refining*
  while (*update_flag*) **do**
    optimality_flag = TRUE
    update_flag = FALSE
    Solve B-LP(**B**). Let (u^*, v^*, Λ) be its dual optimal solution and W^* its optimal value.
    optimality_flag = `optimality_check`(u^*, v^*, Λ)
    if (*optimality_flag*) **then** Goto Step 2 **endif**
    if ($\frac{W^* - best\_objective\_value}{W^*} < \varepsilon$) **then** Goto Step 2 **endif**
    if ($best\_objective\_value < W^*$) **then** $best\_objective\_value = W^*$ **endif**
    for $(i, t) \in \{1, \ldots, N\} \times \{1, \ldots, T\}$ **do**
      for $b = 1, \ldots, B_{i,t}$ **do**
(i) **if** $\sum_{k\in K_{i,t,b}^+}(g_{t,k} - a_k^0 v_t^*) > u_{i,t,b}^*$ **then**
          update_flag = TRUE
          B = `update_binning`(i, t, b)
        endif
      endfor
    endfor
  endwhile
Step 2: *Output the Solution*
  Return W^*, **B**, $(x^*, y^*, \bar{z}^*)$ the optimal primal solution to B-LP(**B**),
  and v^*, the optimal duals corresponding to (15).

The `optimality_check` procedure from Algorithm 4.1 ensures that optimality of D-LP is checked according to Proposition 3.1. Explicitly, the procedure can be written as follows:

`optimality_check` (u^*, v^*, Λ)
for $(i,t) \in \{1,\ldots,N\} \times \{1,\ldots,T\}$ **do**

$$R_{i,t}(\Lambda) = \sum_{b=1}^{B_{i,t}} \left(\sum_{k \in \mathscr{B}_{i,t,b}} a_k^0 \right) c_t^{\text{mng}} + \sum_{b=1}^{B_{i,t}} \sum_{k \in \mathscr{B}_{i,t,b}} a_k^0 \lambda_t^1 - \sum_{s=t}^{T} \sum_{j \in \{1,\ldots,N\}, i \in \mathscr{P}(j)} \lambda_{i,j,s}^2 + \sum_{s=1}^{T} \lambda_{i,s}^3.$$

if $(\sum_{k \in \mathscr{A}_i} (g_{t,k} - a_k^0 v_t^*)^+ > R_{i,t}(\Lambda))$ **then** return FALSE
endif
endfor
return TRUE.

We note that if $\varepsilon = 0$, when Algorithm 4.1 terminates we have an available optimal solution to the dual of B-LP and, implicitly of D-LP (see construction in the proof of Proposition 3.1).

In the rest of this section we propose two ways of updating the binning in Step 1 of the algorithm. We will call them *Single Block Splitting* and *Set Splitting* and will describe the `update_binning`(i,t,b) procedure for each of them. When $\varepsilon = 0$ the methods are exact, in the sense that the optimal value of B-LP equals the optimal value of D-LP. When $\varepsilon > 0$, the methods are heuristics, in the sense that the binning is not further refined if a significant improvement in the objective value of **B**-LP is not recorded. This improvement is controlled by ε.

We propose to always run Algorithm 4.1 with the simplest initial binning **B** that one can choose: the binning in which each aggregate contains exactly one bin in all time periods, i.e. $\mathscr{B}_{i,t,1} = \mathscr{A}_i$, $i = 1,\ldots,N$, $t = 1,\ldots,T$.

4.1. Single block splitting

If for a given aggregate i, time period t and bin b for which $|\mathscr{B}_{i,t,b}| > 1$ the corresponding subproblem (36) does not have the optimal objective value zero we know that for all feasible solutions to the subproblem (36) some dual constraints associated with δ variables are violated. From Lemma 3.5 and its proof we know that this happens because of blocks in $K_{i,t,b}^+$. Therefore we propose to separate one of the blocks in the set $\arg\max_{k \in K_{i,t,b}^+} (g_{t,k} - a_k^0 v_t^*)$. If this set has more than one element, then we choose to separate the block for which the difference between the *grade* of the block (the ratio of units of attribute 1 to tonnes of rock) and the grade of the bin is the greatest. In case of ties, we select the lowest index block. Next we describe in detail the procedure `update_binning` of Algorithm 4.1 (Step 1).

Single-Block Splitting:
`update_binning`(i,t,b)
Let $\hat{K}_{i,t,b}^+ = \arg\max_{k \in K_{i,t,b}^+} (g_{k,t} - a_k^0 v_t^*)$.
Choose k^* to be the block of lowest index in $\arg\max_{k \in \hat{K}_{i,t,b}^+} \left| \dfrac{a_k^1}{a_k^0} - \dfrac{\sum_{k \in \mathscr{B}_{i,t,b}} a_k^1}{\sum_{k \in \mathscr{B}_{i,t,b}} a_k^0} \right|$
Remove k^* from $\mathscr{B}_{i,t,b}$ and update the characteristics of $\mathscr{B}_{i,t,b}$.
Create a new bin $\mathscr{B}_{i,t,B_{i,t}+1} = \{k^*\}$.
Return **B**.

We note that this method is closely related to the adaptive clustering method [14,21] for very large-scale LPs. Litvinchev and Tsurkov [14] show that the iterative adaptive clustering method finishes with an optimal solution for the LP considered in the presence of non-degeneracy assumptions. In the case of the adaptive clustering method; however, the number of aggregates is fixed *a priori* (equal to the number of constraints plus one), whereas in our setup the number of aggregates (more precisely, bins) is allowed to slowly increase. Using quasi-reduced costs (which translates into violation of dual constraints), at every iteration the block that most wants to separate from its bin is singled out and forms its own bin for processing decisions.

We do not anticipate that single block splitting will lead to a numerically efficient method of bin refinement. Mathematically, it is the simplest type of refinement to consider and we will use it as a base case for evaluating Algorithm 4.1.

4.2. Set splitting

We now generalise this idea to *sets* of blocks. Instead of allowing at most one block to separate from each bin for which the optimal value of its corresponding subproblem (36) is not zero, we allow at most one set of blocks to leave the bin and form a new bin. Again, since for such any aggregate i, time period t and bin b with $|\mathscr{B}_{i,t,b}| > 1$ the blocks in $K^+_{i,t,b}$ are those that may create problems, we choose to move them into a new bin. The content of `update_binning` procedure for this approach is given below:

Set Splitting:
`update_binning`(i, t, b)
Remove all blocks in $K^+_{i,t,b}$ from $\mathscr{B}_{i,t,b}$ and update the characteristics of $\mathscr{B}_{i,t,b}$.
Create a new bin $\mathscr{B}_{i,t,B_{i,t}+1} = K^+_{i,t,b}$.
Return **B**.

We now show that when the set splitting method is used if `update_binning` is called, then there is at least one block in the bin with negative quasi-reduced cost and at least one block with positive quasi-reduced cost. This shows that (a) there is always a nontrivial splitting occurring when `update_binning` is called and (b) the test used at Step 1(i) is able to identify bins with the property described in Proposition 3.2. We first introduce some additional notation: let $K^0_{i,t,b} = \{k \in \mathscr{B}_{i,t,b} : g_{t,k} - a^0_k v^*_t = 0\}$.

Lemma 4.1. *Let* $(i, t) \in \{1, \ldots, N\} \times \{1, \ldots, T\}$ *and* $b \in \{1, \ldots, B_{i,t}\}$ *for which* $\sum_{k\in K^+_{i,t,b}}(g_{t,k} - a^0_k v^*_t) > u^*_{i,t,b}$. *Then there exist* $k, k' \in \mathscr{B}_{i,t,b}$ *such that* $g_{t,k} - a^0_k v^*_t > 0$ *and* $g_{t,k'} - a^0_{k'} v^*_t < 0$.

Proof. Since $\sum_{k\in K^+_{i,t,b}}(g_{t,k} - a^0_k v^*_t) > u^*_{i,t,b}$ and $u^*_{i,t,b} \geqslant 0$, it follows that $\sum_{k\in K^+_{i,t,b}}(g_{t,k} - a^0_k v^*_t) > 0$. Hence, $K^+_{i,t,b} \neq \emptyset$, i.e. there exists $k \in \mathscr{B}_{i,t,b}$ with $g_{t,k} - a^0_k v^*_t > 0$.

We now want to show that there exists $k' \in \mathscr{B}_{i,t,b}$ with $g_{t,k'} - a^0_{k'} v^*_t < 0$. We assume that the opposite is true, i.e. $K^+_{i,t,b} \cup K^0_{i,t,b} = \mathscr{B}_{i,t,b}$. Since $\sum_{k\in K^+_{i,t,b}}(g_{t,k} - a^0_k v^*_t) > u^*_{i,t,b}$, we know that subproblem (36) has a non-zero optimal value. We also know that (36) is feasible (as a direct consequence of Remark 3.4), therefore there exists a feasible solution $(\mu_{t,k}, v_{t,k})$ of (36) such that $\sum_{k\in\mathscr{B}_{i,t,b}}(\mu_{t,k} - v_{t,k}) \geqslant \sum_{k\in\mathscr{B}_{i,t,b}}(g_{t,k} - a^0_k v^*_t)$. Since we assumed that $\mathscr{B}_{i,t,b} = K^+_{i,t,b} \cup K^0_{i,t,b}$, we can write the previous inequality as $\sum_{k\in\mathscr{B}_{i,t,b}}(\mu_{t,k} - v_{t,k}) \geqslant \sum_{k\in K^+_{i,t,b}\cup K^0_{i,t,b}}(g_{t,k} - a^0_k v^*_t) = \sum_{k\in K^+_{i,t,b}}(g_{t,k} - a^0_k v^*_t)$. Since our working assumption is that $\sum_{k\in K^+_{i,t,b}}(g_{t,k} - a^0_k v^*_t) > u^*_{i,t,b}$ and $v_{t,k} \geqslant 0$, it follows that $\sum_{k\in\mathscr{B}_{i,t,b}} \mu_{t,k} > u^*_{i,t,b}$, which contradicts the feasibility conditions of (36). □

4.3. Simplification of the linear programming relaxation

As Algorithm 4.1 requires repeated solving of B-LP, we investigated the possibility of reducing the size of the model by reducing the number of variables. We form a reduced LP, denoted B-LP$_{\text{red}}$, by removing all x variables.

B-LP$_{\text{red}}$ has the same objective function as B-LP and retains the constraints (14)–(16), as well as (8) and (17). The constraints (4) and (5) are merged into a new constraint,

$$\sum_{s=1}^{t} y_{i,s} \leqslant \sum_{s=1}^{t} y_{j,s}, \quad \forall i \in \{1, \ldots, N\} \quad \text{s.t. } \mathscr{P}(i) \neq \emptyset, \quad j \in \mathscr{P}(i), \quad t \in \{1, \ldots, T\}, \tag{40}$$

while constraint (6) will disappear. A new set of constraints is added:

$$\sum_{t=1}^{T} y_{i,t} \leqslant 1, \quad \forall i \in \{1, \ldots, N\}. \tag{41}$$

Table 1
Numerical results for Algorithm 4.1 (GA(**B**, 0))

Problem name	GA(**B**,0)					
	No. iter.	Total B-LP time (s)	Total subpr. time (s)	No. subpr. solved	Final avg. no. of bins per agg.	Opt. sol. last iter./10^8
115-8513	575	34 769	4.09	37 613 975	33.72	7.2412928

B is the aggregate binning and single block splitting was used to update the binnings. The column "Final avg. no. of. bins" gives the average number of bins per aggregate at the end of the algorithm, in other words, for the B-LP solved at the last iteration. The problem names contain first the number of aggregates, then the number of blocks. The number of time periods allowed is 20.

Lemma 4.2. 1. *Let* $(x, y, \bar{z})$ *be a feasible solution of* B-LP. *Then* $(y, \bar{z})$ *is a feasible solution of* B-LP_{red} *with the same objective value*.

2. *Let* $(y, \bar{z})$ *be a feasible solution of* B-LP. *We define* $x_{i,t} = \sum_{s=1}^{t} y_{i,s}$, *for any* $i = 1, \ldots, N$ *and* $t = 1, \ldots, T$. *Then* $(x, y, \bar{z})$ *is a feasible solution of* B-LP_{red} *with the same objective value*.

The proof of Lemma 4.2 is straightforward and omitted. It follows from Lemma 4.2 that the optimal value of B-LP is equal to that of B-LP_{red}. Furthermore, the results developed in Sections 3.2 onwards also apply to B-LP_{red}.

This simplification of B-LP is a basic one. More sophisticated approaches can be designed; however, this is not our goal in this paper. We found that this simple change can sometimes improve the running time. In all numerical results reported we use the reduced model to solve B-LP.

4.4. Numerical results for B-LP and D-LP

We demonstrate Algorithm 4.1 on four data sets. Three of the data sets are constructed from a block model with 8513 blocks and are named 115-8513, 296-8513, and 1038-8513. These data sets are comprised of 115, 296 and 1038 aggregates, respectively. Our fourth data set, named 125-96821 contains 96821 blocks and is partitioned into 125 aggregates.

4.4.1. Single block splitting

We tested the single block splitting rule for updating the binning on the smallest data set (115-8513), with $\varepsilon = 0$. After 575 iterations, the optimality check of Algorithm 4.1 is satisfied. At this point, the algorithm terminates and we are guaranteed that the optimal value of B-LP(**B**) with **B** the final binning produced by Algorithm 4.1 equals the optimal value of D-LP. This is borne out in Table 1.

While the single block splitting approach correctly produces an optimal binning, it is not a computationally efficient approach. We now detail results for the more efficient approach of set splitting.

4.4.2. Set splitting

Table 2 demonstrates that the set splitting approach is far superior to single block splitting, as expected. Algorithm 4.1 does not require a large number of iterations (at most five) and in all cases can be performed to optimality (i.e. it can be solved even for $\varepsilon = 0$). The number of iterations does decrease when $\varepsilon = 0.1$ is used; however, this seems to have an impact only on the total time spent on re-solving B-LP. The number of bins created is also smaller, but only significantly so in the case of 125-96821. In all cases but one, that of problem 115-8513, the time required to solve B-LP for the binning constructed when $\varepsilon = 0.1$ is significantly less than for that found for $\varepsilon = 0$. In summary, our main goal of solving D-LP instances that have too many blocks to solve directly has been achieved by identifying small optimal binnings for B-LP.

5. LP-based bin construction for OPMPSP

In Section 2 we argued that greater processing selectivity created the opportunity for higher quality solutions of OPMPSP. We return to the observation made at the conclusion of Section 2 that the solution quality of D-MIP could

Table 2
Numerical results for Algorithm 4.1 (GA(**B**, ε)) for the set splitting

Problem name	GA(**B**,0)						GA(**B**,0.1)					
	No. iter.	Total B-LP time (s)	Total subpr. time (s)	No. subpr. solved	Final avg. no. of bins per agg.	Opt. sol. last iter./10^8	No. iter.	Total B-LP time (s)	Total subpr. time (s)	No. subpr. solved	Final avg. no. of bins per agg.	Opt. sol. last iter./10^8
115-8513	5	21.4	0.07	17375	1.72	7.2412928	3	12.05	0.05	5555	1.42	7.2412583
296-8513	5	122.08	0.07	31048	1.48	7.285218	3	73.97	0.04	13698	1.31	7.2851975
1038-8513	5	8546.28	0.09	118282	1.18	7.3186772	3	6586.41	0.05	44711	1.15	7.3186755
125-96821	5	37.19	2.15	33869	2.79	0.50918786	3	22.57	1.72	8776	1.80	0.50916137

B represents the aggregate binning and the values considered for ε are 0 and 0.1. The column "Final avg. no. of. bins" gives the average number of bins per aggregate at the end of the algorithm, in other words, for the B-LP solved at the last iteration. The problem names contain first the number of aggregates, then the number of blocks. The number of time periods allowed is 20 for the problems with 8513 blocks and 25 for the problem with 96821 blocks.

be achieved by B-MIP for suitably defined bins. In this section, we describe methods to determine a binning with few bins such that the optimal value of B-MIP(**B**) is very close to that of D-MIP.

As mentioned before, two extreme choices of the binning are the *aggregate binning* where each aggregate contains exactly one bin in each time period and the *block binning* where each aggregate i contains exactly $|\mathscr{A}_i|$ bins in each time period. The former is the most coarse binning while the latter is the most refined. Solving B-MIP with these two binnings provides the interval in which the optimal objective value lies as the binning changes. We note that solving B-MIP with the block binning is equivalent to solving D-MIP. Our goal is to find a binning **B** with relatively few bins so that the optimal value of B-MIP(**B**) is as close as possible to the optimal value obtained with the block binning.

As discussed in Section 2.2 the procedure of finding a binning **B** so that B-LP(**B**) is equivalent to D-LP could be used at every node of the branch-and-bound tree for D-MIP. However, we will show that doing this only at the root node produces results very close to optimality.

5.1. Binning based on an optimal binning of D-LP

The simplest heuristic for creating a good binning for B-MIP is to take a small binning **B** so that B-LP(**B**) is equivalent to D-LP, and use this binning to solve B-MIP(**B**). In particular, we will take **B** as the output of Algorithm 4.1. We keep this binning throughout the branch-and-bound tree, and obtain very good results. In principle, re-binning at every node could yield even better results. However, the gap, observed with fixed binning is so small, re-binning can offer only marginal benefits, and we do not attempt it.

5.2. Binning based on optimal dual values of D-LP

Proposition 2.1 describes an alternative method for creating bins by splitting the aggregates using the optimal dual values v^* corresponding to the processing constraints of D-LP. These optimal dual values v^* are output at the termination of Algorithm 4.1. However, it is not clear that using the optimal dual values provide the best boundaries for splitting the aggregates. Therefore we propose a way of creating a binning that allows some "slack" in the splitting by creating additional bins at important locations. Proposition 2.1 states that there exists an optimal binning for B-LP that splits aggregates according to (11), using the optimal dual values v^* corresponding to the processing constraints of D-LP. If an optimal binning for B-LP is close to an optimal binning for B-MIP, then it will be most difficult to decide whether a block k should be placed in a "waste" bin (corresponding to $z_{t,k}=0$) or an "ore" bin (corresponding to $z_{t,k}=y_{i(k),t}$) when $g_{t,k}/a_k^0 \approx v_t^*$. Thus, we let $\theta \in [0,1]$ and for any time period t we define $v_t^{*l}=(1-\theta)v_t^*$ and $v_t^{*u}=(1+\theta)v_t^*$. For any aggregate i we create the bins $\mathscr{B}_{i,t,1}=\{k \in \mathscr{A}_i : g_{t,k}/a_k^0 < v_t^{*l}\}$ and $\mathscr{B}_{i,t,2}=\{k \in \mathscr{A}_i : g_{t,k}/a_k^0 > v_t^{*u}\}$. The blocks $k \in \mathscr{A}_i$ that are not contained in $\mathscr{B}_{i,t,1}$ or $\mathscr{B}_{i,t,2}$, if they exist, are grouped according to their $g_{t,k}/a_k^0$ values. For each distinct such value a new bin will be created, comprising all blocks k with equal $g_{t,k}/a_k^0$. We note that some of the bins defined above can be empty.

It is easy to see that if $\theta=0$, then the binning has at most three bins per time period. In this case, the binning obtained should be a coarsening of the binning output by GA(**B**,0). We know that an optimal binning of B-MIP also has this property. If the binning defined above were optimal for B-MIP, then the constraints $\bar{z}_{i,t,1}=0$ and $\bar{z}_{i,t,2}=y_{i,t}$ could be added to B-MIP, for any i and t. However, in general, this is not the case.

5.3. Binning based on inspection of an optimal solution of D-LP

If (x^*, y^*, z^*) is an optimal solution of D-LP, a third straightforward heuristic is to create bins based on the values $z_{t,k}^*$. More precisely, for a given aggregate i and time period t we define $B_{i,t}=|\{z_{t,k}^* : k \in \mathscr{A}_i\}|$ bins by grouping all blocks k with equal $z_{t,k}^*$ values into the same bin. An obvious problem with this heuristic is that for the time periods in which $y_{i,t}^*=0$, only one bin, which contains the whole aggregate, is created. Thus there can be no additional processing selectivity for such an $\mathscr{A}_i$ in period t when solving B-MIP. We will call this binning *variable in time*.

An easy way to correct this shortcoming is to select for each aggregate i a time period t_i so that $t_i \in \arg\max_{t=1,\ldots,T} y_{i,t}^*$ and create bins based on the $z_{t_i,k}^*$ values only. In any time period, an aggregate i would contain $B_{i,t_i}=|\{z_{t_i,k}^* : k \in \mathscr{A}_i\}|$ bins and the blocks k in any such bin have equal $z_{t_i,k}^*$ values. We remark that it is typical for aggregates to be excavated

Table 3
Numerical results for B-MIP solved for the two extreme choices of binning: aggregate and block binning

Problem name	**B**-MIP with aggregate binning				**B**-MIP with block binning (i.e. D-MIP)			
	Best int. sol./10^8	Time (s)	Number of B&B nodes	Gap (%)	Best int. sol./10^8	Time (s)	Number of B&B nodes	Gap (%)
115-8513	5.49711620	89.96	60	0.79	7.14398448	658.36	20	0.96
296-8513	5.86208811	491.10	129	0.65	7.18134983	1229.82	30	0.93
1038-8513	6.36998649	15380.07	246	0.96	7.24476992	15847.54	80	0.62
125-96821	0.28134624	56204.24	224183	1.00	–	–	–	–

The problem names contain first the number of aggregates, then the number of blocks. The number of time periods allowed is 20 for the group of problems with 8513 blocks and 25 for the problem with 96821 blocks. In all cases, CPLEX is run with emphasis 0 (the default value) and tolerance 1%. If a "–" appears in the columns corresponding to a method, it means that the problem was not solved before our time limit of 60000 s was reached.

fully over a contiguous set of periods, so that the choice of selecting t_i as the period of maximal $y^*_{i,t}$ is reasonable as the aggregate will be fully excavated in a temporal neighbourhood of t_i. We call this binning *constant in time*.

5.4. A generic approximate algorithm for solving D-MIP

The algorithm below is a generic approximation algorithm for producing a binning $\mathbf{B}'$ and solving B-MIP($\mathbf{B}'$). In all numerical experiments, the input binning $\mathbf{B}$ will be the aggregate binning.

Algorithm 5.1. *A Generic Approximate Algorithm for Solving D-MIP* (GAA($\mathbf{B}$, ε)).

Let $\mathbf{B}$ be a binning for our problem and ε a non-negative value smaller than 1.
Run GA($\mathbf{B}$,ε).
$\mathbf{B}' =$ `establish_binning`.
Solve B-MIP($\mathbf{B}'$).

The `establish_binning` procedure will be one of the following four options:

METHOD 1: Set $\mathbf{B}'$ to be the output binning of GA($\mathbf{B}$,ε) (Section 5.1),
METHOD 2: Create $\mathbf{B}'$ based on optimal dual values of D-LP (Section 5.2),
METHOD 3: Create $\mathbf{B}'$ as a "variable in time" binning (Section 5.3),
METHOD 4: Create $\mathbf{B}'$ as a "constant in time" binning (Section 5.3).

5.5. Numerical results

We first establish the optimal value of D-MIP. Table 3 compares the results obtained for solving B-MIP with the aggregate and with the block binning. As expected, solving B-MIP with the block binning takes longer than with the aggregate binning, but the value of the additional flexibility in processing decisions is very significant. The comparison between this two extreme binnings can be made only in the case of the group of problems 8513 blocks; solving B-MIP with block processing becomes impractical for the data set with 96821 blocks. In the latter case the problem was not solved within our time limit of 60000 s; in fact, even the root note LP did not solve within this time limit.

The numerical results that we present in Tables 4–6 describe only the final step of Algorithm 5.1 where B-MIP($\mathbf{B}'$) is solved. The reader needs to keep in mind that the total running time of the algorithm will have to also include the time needed by GA($\mathbf{B}$,ε) for which we presented separate results in Section 4.4. Extra time is also needed to establish the binning to use in B-MIP($\mathbf{B}'$), but the time required for this is always insignificant with respect to the total running time. For the data sets considered, it turns out that we do not gain too much by using $\varepsilon > 0$, and for brevity we only present results obtained for Algorithm 5.1 with $\varepsilon = 0$.

Firstly, we tested the performance of B-MIP with the binning output by GA($\mathbf{B}$, 0) (METHOD 1), where the initial binning $\mathbf{B}$ is the aggregate binning; see Table 4. Already, this approach is performing exceedingly well; the solution times (when combined with the bin-creation times listed in Table 2) are around those for aggregate binning in Table 3,

Table 4
Numerical results for B-MIP with the binning output by GA(**B**, 0) (METHOD 1)

Problem name	**B**-MIP with the binning output by GA(**B**, 0) (METHOD 1)					
	Avg. no. of bins per agg.	Total no. of bins	Best int. sol. $/10^8$	Time (s)	No. of B&B nodes	Gap (%)
115-8513	1.72	3962	7.14894035	29.9	30	0.94
296-8513	1.48	8759	7.19693970	223.36	70	0.97
1038-8513	1.18	24530	7.24171577	7099.78	40	0.55
125-96821	2.79	8727	0.49372764	3429.82	2759	1.00

B is the aggregate binning. The binning method used by GA(**B**, 0) was the set splitting. The problem names contain first the number of aggregates, then the number of blocks. The number of time periods allowed is 20 for the problems with 8513 blocks and 25 for the problem with 96821 blocks. In all cases, CPLEX is run with emphasis 0 (the default value) and tolerance 1%.

Table 5
Numerical results for B-MIP solved for binning determined using a dual optimal solution of D-LP (METHOD 2)

Problem name	**B**-MIP with binning based on optimal dual values of D-LP (METHOD 2)											
	$\theta = 0$						$\theta = 0.1$					
	Avg. no. of bins per agg.	Total no. of bins	Best int. sol. $/10^8$	Time (s)	No. of B&B nodes	Gap (%)	Avg. no. of bins per agg.	Total no. of bins	Best int. sol. $/10^8$	Time (s)	No. of B&B nodes	Gap (%)
115-8513	1.45	3335	7.14298466	34.61	50	0.99	2.11	4847	7.16049272	44.41	40	0.80
296-8513	1.32	7837	7.1987392	268.74	80	0.99	1.39	8205	7.1977348	196.44	30	0.90
1038-8513	1.16	24043	7.2536959	6525.57	30	0.41	1.22	25385	7.2432929	5134.53	10	0.52
125-96821	1.86	5819	0.49314388	959.83	1160	1.00	2.23	6957	0.49407884	2112.13	2319	0.99

θ takes two values: 0 and 0.1. The problem names contain first the number of aggregates, then the number of blocks. The number of time periods allowed is 20 for the problems with 8513 blocks and 25 for the problem with 96821 blocks. In all cases, CPLEX is run with emphasis 0 (the default value) and tolerance 1%.

Table 6
Numerical results for B-MIP solved for binning determined using an optimal solution of D-LP (METHOD 3 and 4)

Problem name	**B**-MIP with binning based on inspection of an optimal solution of D-LP											
	Binning variable in time (METHOD 3)						Binning constant in time (METHOD 4)					
	Avg. no. of bins per agg.	Total no. of bins	Best int. sol. $/10^8$	Time (s)	No. of B&B nodes	Gap (%)	Avg. no. of bins per agg.	Total no. of bins	Best int. sol. $/10^8$	Time (s)	No. of B&B nodes	Gap (%)
115-8513	1.06	2437	7.1532821	17.22	30	0.63	1.42	3260	7.1294375	8.06	0	0.96
296-8513	1.05	6200	7.1881359	200.12	30	0.94	1.32	7840	7.1837552	120.77	40	0.78
1038-8513	1.03	21323	7.2283491	3494.97	30	0.71	1.19	24700	7.2289816	5394.89	80	0.80
125-96821	1.19	3724	0.49392999	419.77	796	1.00	1.87	5850	0.49323200	2023.95	1762	1.00

The problem names contain first the number of aggregates, then the number of blocks. The number of time periods allowed is 20 for the problems with 8513 blocks and 25 for the problem with 96821 blocks. In all cases, CPLEX is run with emphasis 0 (the default value) and tolerance 1%.

while the objective values are within the optimality gap of 1% of the best possible values (those obtained with block binning) in Table 3. In addition, for the largest problem 125-96821 we are able to obtain an improvement in objective value for B-MIP(**B**) over the aggregate binning of 75.49%.

We now demonstrate that METHODS 2–4, are moderate improvements over METHOD 1 in the sense that they will create fewer bins, leading to somewhat shorter running times, while maintaining solution quality.

Table 5 provides some insight on the quality of the B-MIP solution obtained using the binning based on dual values of D-LP (METHOD 2). We construct a binning for two values of θ (0 and 0.1). The numerical results show that the quality of the objective value obtained is very good for both binnings. Since we do not solve B-MIP all the way to optimality, but stop when the gap is lower than or equal to 1%, it can even happen that the objective value obtained with the METHOD 2 binnings are higher than the one obtained for the block binning. While the time needed to solve B-MIP for $\theta = 0.1$ is higher than for $\theta = 0$ for the first problem, it is significantly reduced in the case of problems 296-8513 and 1038-8513. In general, the number of bins created when $\theta = 0.1$ is higher, but that does not seem to affect much the performance of the algorithm with the exception of running times. If we compare the best integer solution obtained for the two binnings considered with the best integer solution obtained for block processing, we see that the objective values are very close in all cases. For all problems with 8513 blocks the binnings defined perform very well, in fact, the results are close to those of D-MIP. We cannot perform such a comparison in the case of problem 125-96821, since D-MIP was too difficult to solve directly.

When the splitting is based on an optimal solution of D-LP (METHODS 3 and 4) the number of bins created is smaller than in the case of splitting based on dual optimal values; see Table 6. As expected, the running time for solving B-MIP also decreases, both for binnings constant and variable in time. The only exception is when B-MIP with a constant binning is solved for problem 1038-8513. The quality of the solutions obtained is again very good; the best integer solutions identified are again very close to those obtained for D-MIP (Table 3).

6. Conclusions and future work

We have demonstrated an efficient iterative disaggregation approach to using a finer spatial resolution for processing decisions in the OPMPSP, while allowing extraction decisions to be made at an aggregated level. Furthermore we have shown that this approach can lead to significant improvements in NPV. We investigated the characteristics of the problem and showed how the aggregates can be systematically divided into bins (groups of blocks) so that the solution of the LP relaxation with all processing decision variables fully disaggregated to block level (D-LP) can be recovered from the solution of our compactly disaggregated LP relaxation (B-LP) with processing decisions made at the level of bins. As the number of bins is much smaller than the number of blocks, using our binning approach, D-LP can be solved to optimality for much larger data instances than by a direct disaggregation approach.

We proposed four strategies for integrating our exact efficient LP solving method into an algorithm for rapidly solving the OPMPSP. Each strategy represented a different approach to determining binnings with which to solve the MIP formulation of OPMPSP. The binning strategies were based on structural properties of the problem and on information obtained from exact solutions of D-LP or its dual. We showed that all binning strategies performed very well in terms of achieving a MIP objective that was very close to the MIP with all processing decisions disaggregated to the level of blocks (D-MIP). Moreover, the running times of our Algorithm 5.1 were typically less than the solution time of the MIP with the coarsest of binnings (and lowest quality of solution).

In the future we would like to obtain additional block models on which to test our approach. The method we have constructed may be viewed as general purpose machinery that may be applied to extensions of OPMPSP such as models involving stockpiling and models that incorporate stochastic geological and financial elements.

Acknowledgements

The authors are very grateful to Merab Menabde, Peter Stone and Mark Zuckerberg (BHP Billiton) for their ongoing support and guidance on a variety of practical mining-related issues and for numerous technical suggestions and insightful feedback that improved the content and exposition of this work.

This research is supported by the Australian Research Council Linkage Project LP0561744 and by BHP Billiton Limited.

Notation

$t = 1, \ldots, T$	Time periods from 1 to T
$i = 1, \ldots, N$	Aggregates from 1 to N
$k = 1, \ldots, K$	Block numbers, from 1 to K

a_k^0	Amount of rock in block k
a_k^1	Number of units of attribute 1 in block k
c_t^{mng}	Estimated mining cost in time period t, per tonne of rock
c_t^{proc}	Estimated processing cost in time period t, per tonne of rock
$\mathscr{K} = \{1, \ldots, K\}$	The set of blocks
$\mathscr{A}_i$	Aggregate i
$\mathscr{B}_{i,t,b}$	Bin number b of aggregate i in time period t
$B_{i,t}$	The number of bins in aggregate i in time period t
$\mathscr{P}(i)$	The set of predecessors of aggregate i
$g_{t,k}$	$c_t^1 a_k^1 - c^{\text{proc}} a_k^0$
$K^+_{i,t,b}$	The set of blocks in bin b of aggregate i in time period t for which $g_{t,k} - a_k^0 v_t^* > 0$
$\mathbf{B}$	The partition of the set of blocks into bins ("the binning")
u^*	B-LP Case: Non-negative dual variables corresponding to constraints (14)
v^*	B-LP Case: Non-negative dual variables corresponding to (15)
$\Lambda = (\lambda^1, \ldots, \lambda^5)$	B-LP Case: Non-negative dual variables corresponding to (16), (4)–(6) and to x upper bounds
μ	δ-LP Case: Non-negative dual variables corresponding to (27) and (28)
v^*	δ-LP Case: Non-negative dual variables corresponding to (29)
$\Lambda = (\lambda^1, \ldots, \lambda^5)$	δ-LP Case: Non-negative dual variables corresponding to (16), (4)–(6) and x upper bounds
v	δ-LP Case: Real dual variables corresponding to (30)
α	D-LP Case: Non-negative dual variables corresponding to (1)
v^*	D-LP Case: Non-negative dual variables corresponding to (2)
$\Lambda = (\lambda^1, \ldots, \lambda^5)$	D-LP Case: Non-negative dual variables corresponding to (3)–(6) and x upper bounds
D-MIP	MIP with block binning
D-LP	LP relaxation of D-MIP
B-MIP	MIP with bins
B-LP	LP relaxation of B-MIP
δ-MIP	MIP with bins and δ var (representing the difference betweenblock and bin processing)
δ-LP	LP relaxation of δ-MIP

References

[1] Whittle Programming Pty. Ltd. Four-X Whittle multi-element open pit optimisation software—user manual, 1998.

[2] Lerchs H, Grossmann IF. Optimum design of open-pit mines. Transactions CIM 1965;LXVIII:17–24.

[3] Dowd PA, Onur AH. Open-pit optimization—part 1: optimal open-pit design. Transaction of the Insitution of Mining and Metallurgy (Section A: Mining Industry) 1993;102:A95–104.

[4] Onur AH, Dowd PA. Open-pit optimization part 2: production scheduling and inclusion of roadways. Transactions of the Institution of Mining and Metallurgy (Section A: Mining Industry) 1993;102:A105–13.

[5] Tolwinski B, Underwood R. A scheduling algorithm for open pit mines. IMA Journal of Mathematics Applied in Business and Industry 1996;7:247–70.

[6] Ramazan S, Dimitrakopoulos R. Recent applications of operations research in open pit mining. Transactions of the Society for Mining, Metallurgy and Exploration 2004;316:73–8.

[7] Caccetta L, Hill SP. An application of branch and cut to open pit mine scheduling. Journal of Global Optimization 2003;27:349–65.

[8] MineMax Ltd. MineMax Scheduler. ⟨http://www.minemax.com/products/scheduler⟩.

[9] Gemcom Ltd. Gemcom Whittle. ⟨http://www.gemcomsoftware.com/products/Whittle⟩.

[10] Boland N, Fricke C, Froyland G. A strengthened formulation for the open pit mine production scheduling problem. Available at Optimization Online; 2007.

[11] Fricke C. Applications of integer programming in open pit mining. PhD thesis, The University of Melbourne; 2006.

[12] Menabde M, Froyland G, Stone P, Yeates G. Mining schedule optimisation for conditionally simulated orebodies. In: Proceedings of the international symposium on orebody modelling and strategic mine planning: uncertainty and risk management. 2004. p. 347–52.
[13] Whittle J. Optimizing the NPV of very large mining complexes. In: Seminar presented at Australian society for operations research meeting, Melbourne, 20th June 2007.
[14] Litvinchev I, Tsurkov V. Aggregation in large-scale optimization. Applied optimization Dordrecht: Kluwer Academic Publishers; vol. 83, 2003.
[15] Rogers D, Plante R, Wong R, Evans J. Aggregation and disaggregation techniques and methodology in optimization. Operations Research 1991;39(4):553–82.
[16] Marchand H, Wolsey L. Aggregation and mixed integer rounding to solve MIPS. Operations Research 2001;49(3):363–71.
[17] Iyer R, Grossmann I, Vasantharajan S, Culik A. Optimal planning and scheduling of offshore oil field infrastructure investment and operations. Industrial and Engineering Chemistry Research 1998;37:1380–97.
[18] Smith ML. Optimizing inventory stockpiles and mine production: an application of separable and goal programming to phosphate mining using ampl/cplex. CIM Bulletin 1999;92:61–4.
[19] Ramazan S, Dagdelen K, Johnson TB. Fundamental tree algorithm in optimising production scheduling for open pit mine design. Transactions of the Institution of Mining and Metallurgy (Section A: Mining Technology) 2005;114:A45–54.
[20] Ramazan S. The new fundamental tree algorithm for production scheduling of open pit mines. European Journal of Operational Research 2007;177:1153–66.
[21] Jörnsten K, Leisten R, Storoy S. Convergence aspects of adaptive clustering in variable aggregation. Computers & Operations Research 1999;26(10–11):955–66.
[22] Kellerer H, Pferschy U, Pisinger D. Knapsack problems. Berlin: Springer; 2004.