MineLib file format specification V1.0

D. Espinoza* • M. Goycoolea† • E. Moreno† • A. Newman‡

*Department of Industrial Engineering, University of Chile, Santiago, Chile

†School of Business, Universidad Adolfo Ibañez, Santiago, Chile

‡Division of Economics and Business, Colorado School of Mines, Golden, CO 80401

daespino@dii.uchile.cl • marcos.goycooles@uai.cl • eduardo.moreno@uai.cl • newman@mines.edu

July 23, 2012

1 Introduction

Similar to the mixed-integer programming library (MIPLIB), MineLib 2011 [**?**] is a library of publicly available test problem instances for three classical types of open pit mining problems: the ultimate pit limit problem and two variants of open pit production scheduling problems. The data comes from real-world mining projects and simulated data. Instances in MineLib are each made up of three files. The first file contains geological data defining a block-model of the mineral body. This file describes block characteristics such as position in the mine, tonnage, grades, etc. The second type of file is for describing precedence relationships between blocks. Finally, the last type of file contains the processed data required to formulate three production scheduling problems. This document describes in detail the file format for these tree types of files.

2 General assumptions

All files are ASCII, and lines beginning with the character '*%*' are assumed to be comments. Each line contains *fields* delimited (separated) by a space, a horizontal tabular character, or a colon character (ASCII codes 32, 9 and 58, respectively). All separators at the beginning of a line are discarded. Multiple contiguous separators are treated as a single separator.

All entries are of the form <*keyword*> : <*parameter_type*>, or simply, a sequence of <*parameter_type*> definitions. <*keyword*> is an alphanumerical name used to identify certain entries, and <*parameter_type*> defines a variable or data of a certain type. The types <*str*>, <*int*>, <*char*> and <*dbl*> correspond to a string (not containing a separator), an integer, a character, or a double type, as in C and other popular programming languages. We assume that all entries are given in the order specified in this document.

We introduce a flexible format, because this is the way in which many practitioners transfer information about block models at the time of this writing. By maintaining the status quo, we hope that the mining community will contribute to and use the library. Additionally, in practice, not all mines use the same information. For example, in some mines, the blocks have information on three different concentrations of minerals; some information might pertain to contaminants, while other information might pertain to sub-products. Having that information is essential to compute correct values for the block depending on the processing technology used.

3 The Block-Model Descriptor File

The Block-Model Descriptor File stores model information at a block-by-block level. Each line in the file corresponds to a block in the model. All lines have the same number of columns. These columns are organized as follows:

<*int id*> <*int x*> <*int y*> <*int z*> <str_1> ... <str_k>

Each row contains the following information about a block:

- *id* stores a unique identifier for the block, where the block identifiers are numbered, starting with zero.
- x, y, z represent the coordinates of the block, where a zero z-coordinate corresponds to the bottom-most shelf in the orebody and the z-axis points in the upwards direction. The y-axis points directly towards the viewer while the x-axis points to the left of the viewer.
- $str_1, \ldots, str_k$ represent optional user-specified fields that may represent, e.g., tonnage, ore grade, or information about impurities. These values can be of any pure type declared before and must comply with our delimiter rules for parsing. This flexibility is allowed to match the usual formats used in the industry.

4 The Block-Precedence Descriptor File

The Block-Precedence Descriptor File articulates precedence relationships between blocks in the model. Information is represented at a block-by-block level. Each line in the file corresponds to a block in the model and its corresponding set of predecessors. Precedence relationships are described as follows:

$<int\ b> <int\ n> <int\ p_1> \ldots <int\ p_n>$

Each row gives the following precedence information:

- b stores the unique identifier of a block.
- n stores the number of predecessors specified for block b.
- $p_1, \ldots, p_n$ store the identifiers of the n predecessors of block b.

In general, we assume that $p_1, \ldots, p_n$ are immediate predecessors of block b, but this is not a strict requirement. We assume that no two entries in the file can begin with the same identifier. If a block b has no predecessors, then the corresponding value n is set to 0 and no values p_i are specified in the line.

5 Optimization-Model Descriptor File

The Optimization-Model Descriptor File is used to store the necessary information to formulate $(UPIT)$, $(CPIT)$, and $(PCPSP)$.

5.1 Concepts and Notation

Before defining these three problems consider the following notation: Let $B = \{1, \ldots, n\}$ represent the set of all blocks. For each block $a \in B$ let $P(a)$ represent the precedences of a. That is, if $a \in B$ and $b \in P(a)$ then b should be extracted before a. In general it is a good rule of thumb to let $P(a)$ contain only *immediate* precedessors of a. That is, if $b \in P(a)$ and $c \in P(b)$ then $c \notin P(a)$. Note that this is not a strict requirement. Let t_{max} represent the number of time periods being considered, and let d_{max} represent the number of possible destinations to which a block can be sent. Let r_{max} represent the number of resources required to extract a block. Assume that in order to extract block b it is necessary to use q_{br} amount of resource r. Let $\underline{R}_{rt}$ and $\overline{R}_{rt}$ represent the minimum and maximum amount of resource r which can be used in time period t. Finally, let A represent an arbitrary

matrix with $n \times t_{max} \times d_{max}$ columns and m rows, and let q represent a vector having m rows.

These three problem classes are the Ultimate Pit Problem (UPIT), the Constrained Pit Limit Problem (CPIT) and the more general Precedence Constrained Production Scheduling Problem (PCPSP):

UPIT: This problem is also known as the maximum-weight closure problem [1]. Consider a binary variable x_b for each block $b \in B$ indicating if block b should be included in the ultimate-pit. For each block b let p_b represent the profit obtained from including block b in the pit. Note that p_b can be positive or negative. The Ultimate Pit Problem consists in solving:

$$\begin{aligned} \max & \sum_{b \in B} p_b x_b & \\ \text{s.t. } & x_a \leq x_b & \forall a \in B, \forall b \in P(a) \\ & x_b \in \{0,1\} & \forall b \in B \end{aligned}$$

CPIT: Consider a binary variable x_{bt} for each block $b \in B$ and each time period $t \in 1,\ldots,T$ indicating if block b should be extracted in time period t. For each variable x_{bt} let p_{bt} represent the profit obtained from extracting block b in time period t. Note that p_{bt} can be positive or negative. The Constrained Pit Limit Problem consists in solving:

$$\max \sum_{b \in B} \sum_{t=1}^{t_{max}} p_{bt} x_{bt} \tag{1}$$

$$\text{s.t. } \sum_{\tau=1}^{t} x_{a\tau} \leq \sum_{\tau=1}^{t} x_{b\tau} \qquad \forall a \in B, \forall b \in P(a)\ 1 \leq t \leq t_{max} \tag{2}$$

$$\sum_{t=1}^{t_{max}} x_{bt} \leq 1 \qquad 1 \leq t \leq t_{max} \tag{3}$$

$$\underline{R}_{rt} \leq \sum_{b \in B} q_{br} x_{bt} \leq \overline{R}_{rt} \qquad 1 \leq t \leq t_{max},\ 1 \leq r \leq r_{max} \tag{4}$$

$$x_{bt} \in \{0,1\} \qquad \forall b \in B, 1 \leq t \leq t_{max} \tag{5}$$

Constraints (2) impose the precedence constraints. That is, if a is an immediate predecessor of b, then a must be extracted before or in the same time period as b. Constraints (3) impose that each block can be extracted at most once. Constraints (4) impose that the minimum and maximum resource constraints are satisfied each period.

PCPSP: Consider the same binary variables x_{bt} defined in CPIT. In addition, consider continuous variables y_{bdt} representing how much of block b to send to destination d in time period t. The Precedence Constrained Production Scheduling Problem consists in solving:

$$\max \sum_{b\in B}\sum_{d=1}^{d_{max}}\sum_{t=1}^{t_{max}} p_{bdt}\, y_{bdt} \tag{6}$$

$$\text{s.t.} \sum_{\tau=1}^{t} x_{a\tau} \leq \sum_{\tau=1}^{t} x_{bt} \qquad \forall a \in B, \forall b \in P(a),\ 1 \leq t \leq t_{max} \tag{7}$$

$$x_{bt} = \sum_{d=1}^{d_{max}} y_{bdt} \qquad \forall b \in B, 1 \leq d \leq d_{max},\ 1 \leq t \leq t_{max} \tag{8}$$

$$\sum_{t=1}^{t_{max}} x_{bt} \leq 1 \qquad 1 \leq t \leq t_{max} \tag{9}$$

$$\underline{R}_{rt} \leq \sum_{b\in B} q_{bdr} y_{bdt} \leq \overline{R}_{rt} \qquad 1 \leq t \leq t_{max},\ 1 \leq r \leq r_{max} \tag{10}$$

$$\underline{a} \leq Ay \leq \overline{a} \tag{11}$$

$$y_{bdt} \in [0,1] \qquad \forall b \in B,\ 1 \leq d \leq d_{max},\ 1 \leq t \leq t_{max} \tag{12}$$

$$x_{bt} \in \{0,1\} \qquad \forall b \in B,\ 1 \leq t \leq t_{max} \tag{13}$$

Constraints (7), (9), and (10) are as before. Constraints (8) impose that blocks are broken up and sent to their destinations appropriately and constraints (11) represent general side-constraints.

5.2 The file format

5.2.1 `NAME`: <*str s*>

Identifies the data file.

5.2.2 `TYPE`: <*str s*>

Specifies the problem type. The value of *s* must be *UPIT*, *CPIT*, or *PCPSP*.

5.2.3 `NBLOCKS`: <*int n*>

Gives the number of blocks in the problem.

5.2.4 NPERIODS: <*int* t_{max}>

Identifies the number of time periods for the problem; this field is valid for formulating problems of type $(CPIT)$ and $(PCPSP)$.

5.2.5 NDESTINATIONS: <*int* d_{max}>

Specifies the number of possible processing alternatives for each block; this field is valid for formulating problems of type $(PCPSP)$.

5.2.6 NRESOURCE_SIDE_CONSTRAINTS: <*int* r_{max}>

Identifies the number of operational resource constraints per time period; this field is valid for problems of type $(CPIT)$ and $(PCPSP)$.

5.2.7 NGENERAL_SIDE_CONSTRAINTS: <*int* m>

Identifies the number of general side-constraints for the problem, or equivalently, the number of rows in matrix A. This field is valid for problems of type $(PCPSP)$.

5.2.8 DISCOUNT_RATE: <*dbl* α>

This specifies the discount rate used in computing the objective function. That is, $p_{bt} = \frac{p_b}{(1+\alpha)^t}$ and $p_{bdt} = \frac{p_{bd}}{(1+\alpha)^t}$, where p_b and p_{bd} are quantities defined subsequently in the file.

5.2.9 OBJECTIVE_FUNCTION:

The objective function is given by one row for each block. Thus, this section has NBLOCKS lines. If the problem-type is either $(UPIT)$ or $(CPIT)$, the number of destinations is assumed to be one, i.e., NDESTINATIONS=1. In this case, $p_b = p_{b1}$. Each line is of the form:

<*int* b> <*dbl* p_{b1}> ... <*dbl* $p_{bd_{max}}$>

That is, the first value (b) defines the block, and the next d_{max} values describe the objective function values associated with each destination. No two lines can begin with the same identifier.

5.2.10 RESOURCE_CONSTRAINT_COEFFICIENTS:

Here, we define the coefficients q_{br} and q_{brd}, corresponding to constraints (4) and (10) in $(CPIT)$ and $(PCPSP)$. This entry consists of n lines, where n is at most the total number of non-zero coefficients in the aforementioned constraints. Specifically, each of these lines has the form:

<int b> <int r> <dbl v>

or

<int b> <int d> <int r> <dbl v>

The values of b, d, and r indicate the block, the destination, and the operational resource, respectively. The value of v represents the coefficient q_{br} or q_{brd}. All coefficients that are not defined in this way have value zero.

5.2.11 RESOURCE_CONSTRAINT_LIMITS:

Here, we define the limits $\underline{R}_{rt}$ and $\bar{R}_{rt}$ corresponding to constraints (4) and (10) in $(CPIT)$ and $(PCPSP)$, respectively.
This entry consists of NRESOURCE_CONSTRAINTS lines, each having the form:

<int r> <int t> <char c> <dbl v_1>

or

<int r> <int t> <char c> <dbl v_1> <dbl v_2>

The value of r indicates the operational resource and the value of t indicates the time period in which the operational resource constraint holds. The value of c can be L (less-than-or-equal-to), G (greater-than-or-equal-to) or I (within an interval). If c has value L, then $\underline{R}_{rt} = -\infty$ and $\bar{R}_{rt}$ is equal to the value of v_1. In this case, v_2 is not defined. If c has value G, then $\bar{R}_{rt} = \infty$ and the value of $\underline{R}_{rt}$ is equal to v_1. In this case, v_2 is not defined. If c has value I, then v_1 has value $\underline{R}_{rt}$ and v_2 has value $\bar{R}_{rt}$. No default value is assumed for these limits. Thus, if an operational resource constraint has no specific type and limits, the instance is not well defined.

5.2.12 GENERAL_CONSTRAINT_COEFFICIENTS:

Here, we define the coefficients A_{bdtj} of matrix A, corresponding to constraints (11) in $(PCPSP)$, where b is a block identifier, d a destination identifier, t a time period, and j a number between 0 and $m-1$, where m is the number of rows in A. This entry consists of n lines, where n is at most the total number of non-zero coefficients in matrix A. Specifically, each of these lines has the form:

<int b> <int d> <int t> <int j> <dbl v>

The values of b, d, t, j, and v indicate the block, the destination, the time period, the row, and the coefficient A_{bdrj} in the matrix A, respectively. All coefficients that are not defined in this way have value zero.

5.2.13 GENERAL_CONSTRAINT_LIMITS:

Here, we define the limits corresponding to constraints (11) in (*PCPSP*). This entry consists of NGENERAL_SIDE_CONSTRAINTS lines, each having the form:

<int m> <char c> <dbl v_1>

or

<int m> <char c> <dbl v_1> <dbl v_2>

The value of m indicates the row number of A. The value of c can be L, G or I. If c has value L, then $\underline{a}_m = -\infty$ and $\bar{a}_m$ is equal to the value of v_1. In this case, v_2 is not defined. If c has value G, then $\underline{a}_m = \infty$ and the value of $\bar{a}_m$ is equal to v_1. In this case, v_2 is not defined. If c has value I, then v_1 has value $\underline{a}_m$ and v_2 has value $\bar{a}_m$. No default value is assumed for these limits. Thus, if an operational resource constraint has no specific type and limits, the instance is not well defined.

References

[1] R. AHUJA, T. MAGNANTI, AND J. ORLIN, *Network Flows*, Prentice Hall, 1993.