View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Queensland University of Technology ePrints Archive

Queensland University of Technology
Brisbane Australia

This is the author's version of a work that was submitted/accepted for publication in the following source:

Liu, Shi Qiang & Kozan, Erhan
(2016)
New graph-based algorithms to efficiently solve large scale open pit mining optimisation problems.
Expert Systems With Applications, *43*, pp. 59-65.

This file was downloaded from: https://eprints.qut.edu.au/87292/

Notice: *Changes introduced as a result of publishing processes such as copy-editing and formatting may not be reflected in this document. For a definitive version of this work, please refer to the published source:*

https://doi.org/10.1016/j.eswa.2015.08.044

New graph-based algorithms to efficiently solve large scale open pit mining optimisation problems

Shi Qiang Liu and Erhan Kozan[*]

Decision Science Discipline, Mathematical Sciences School

Queensland University of Technology

2 George St GPO Box 2434, Brisbane Qld 4001 Australia

Abstract:

In the mining optimisation literature, most researchers focused on two strategic-level and tactical-level open-pit mine optimisation problems, which are respectively termed *ultimate pit limit* (UPIT) or *constrained pit limit* (CPIT). However, many researchers indicate that the substantial numbers of variables and constraints in real-world instances (e.g., with 50-1000 thousand blocks) make the CPIT's mixed integer programming (MIP) model intractable for use. Thus, it becomes a considerable challenge to solve the large scale CPIT instances without relying on exact MIP optimiser as well as the complicated MIP relaxation/decomposition methods. To take this challenge, two new graph-based algorithms based on network flow graph and conjunctive graph theory are developed by taking advantage of problem properties. The performance of our proposed algorithms is validated by testing recent large scale benchmark UPIT and CPIT instances' datasets of MineLib in 2013. In comparison to best known results from MineLib, it is shown that the proposed algorithms outperform other CPIT solution approaches existing in the literature. The proposed graph-based algorithms leads to a more competent mine scheduling optimisation expert system because the third-party MIP optimiser is no longer indispensable and random neighbourhood search is not necessary.

Keywords: Mine optimisation algorithms; planning and scheduling; mine block sequencing; ultimate pit limit; constrained pit limit; network flow graph.

[*] Corresponding author. Tel: +61 7 3138 1029; Fax: +61 7 3138 2310.
Email: e.kozan@qut.edu.au; sq.liu@qut.edu.au

1. Introduction

Modern mining is a complicated procedure that may sustain over several decades and necessitate huge investment in billions of dollars. To prepare for a feasibility study report at the exploration phase, a tentative strategic mine production plan/schedule should be optimised, that is: which part of orebody should be selected; and in which time period (when) the subset of blocks in this part should be extracted. The first of these questions is answered by the **Ultimate Pit Limit** (**UPIT**) problem in the mining literature. As pioneers, Lerchs and Grossmann (1965) presented to the mining community a dynamic programming method known as the Lerchs-Grossmann approach for UPIT. Caccetta and Giannini (1988) proposed several mathematical theorems to improve the Lerchs-Grossmann approach. Underwood and Tolwinski (1998) developed a dual simplex approach to solve the UPIT model. Hochbaum and Chen (2000) presented a push-relabel algorithm for UPIT based on the network flow graph theory. Nowadays, the UPIT problem has been well defined and computationally tractable to be solved even for the very large UPIT instances in today's computer technology.

After the determination of the ultimate pit contour, the next widely-studied mine optimisation problem type is to answer the second question: when the blocks should be extracted over time periods so that the total net present value is maximised. In the mining community, this problem type is called mine production scheduling Caccetta and Hill, 2003; Boland et al, 2009; Bley et al., 2010; Chicoisne et al., 2012), or open-pit block sequencing (Cullenbine, Wood and Newman 2011; Lambert et al., 2014), or constrained pit limit (Espinoza et al., 2013; MineLib, 2013). For convenience, we use the term **Constrained Pit Limit** (**CPIT**) to call this problem type in this paper. The following leading papers contribute to CPIT solution approaches in the mining optimisation literature. As a pioneer, Caccetta and Hill (2003) proposed a branch-and-cut algorithm embedded LP relaxation and MIP optimiser to solve CPIT. However, due to software commercialisation and confidentiality agreements, they only summarise some important features and the full details of all aspects of their proposed branch-and-cut algorithm are not provided in this paper. Ramazan (2007) proposed a "Fundamental Tree Algorithm" to aggregate the blocks for reducing the number of variables and constraints in the MIP model. Boland et al. (2009) developed a LP-based relaxation approach to solve large-size CPIT instances. Bley et al. (2010) improved the CPIT formulation by adding inequalities derived by generating the union predecessor set to replace the immediate predecessor set. Bienstock, D., & Zuckerberg (2010) developed linear programming (LP) relaxation approaches to solve CPIT efficiently. Cullenbine, Wood and Newman (2011) developed a sliding-time-window algorithm in which the relaxed CPIT formulation models are iteratively solved by MIP optimiser over divided time windows. Chicoisne et al. (2012) proposed a decomposition method to solve the relaxed CPIT formulation model period by period, in which there is a single capacity constraint per period. Then, the feasible solutions of more general CPIT formulation model with multiple capacity constraints are obtained by adding a rounding algorithm based on a topological sorting algorithm. Espinoza et al. (2013) presented a library (i.e., benchmark data and results of varied-size instances) of open-pit mining optimisation problems such as UPIT and CPIT to the mining community. Lambert and Newman (2014) employed a tailored Lagrangian relaxation to efficiently solve the CPIT formulation model. Lambert et al. (2014) concluded a tutorial of several CPIT mathematical formulation models developed in the literature.

The complexity of large scale CPIT problem and its variants led to development of numerous heuristic/metaheuristic algorithms. Kumral and Dowd (2005) developed a simulated annealing metaheuristic combined with Lagrangian relaxation. Ferland et al. (2007) developed a particle swam optimisation metaheuristic for solving CPIT. Myburgh and Deb (2010) reported an application of evolutionary algorithm to solve CPIT, in which an initial feasible sequence of blocks represented as a chromosome is iteratively improved by genetic operators such as crossover and mutation. Souza et al. (2010) developed a hybrid heuristic approach for a CPIT-type problem with the consideration of operational constraints such as truck allocations. Martinez and Newman (2011) developed a heuristic decomposition scheme to efficiently obtain satisfactory CPIT solutions in a real-world implementation. Lamghari and Dimitrakopoulos (2012) presented a tabu search to solve a CPIT-type problem with the consideration of metal uncertainty. Alonso-Ayuso et al. (2014) developed a heuristic approach to solve a stochastic CPIT model with the consideration of ore prices. Lamghari et al. (2015) developed a two-phase approach to solve the CPIT problem, in which the first phase is to generate the initial solution by a series of linear programming models and the second phase is to apply a variable neighbourhood search procedure to improve the initial solution. Shishvan and Sattarvand (2015) developed an Ant Colony Optimisation (ACO) metaheuristic to solve an extended CPIT-type problem applied in a Copper-Gold mine.

According to the above literature review, the following five ways in solving large scale CPIT instances are concluded and categorised below:

i. reduce the full model size by aggregating the blocks and periods;
ii. relax the full model complexity by decreasing the number of variables or constraints;
iii. decompose the full model into several sub-models so that much less number of constraints and variables become tractable;
iv. embed heuristics within MIP optimiser to accelerate the solution procedure; and
v. develop metaheuristics with neighbourhood search.

However, the above first four ways still rely on the use of third-party MIP optimiser software and sophisticated relaxation/decomposition approaches. The fifth way by the development of metaheuristics such as generic algorithm (i.e., neighbourhood search with random diversification mechanisms) may be not advanced enough because there exist unexpected randomness in the solution procedure and the critical CPIT problem's structural properties are not utilised. Hence, the purpose of this study is to develop new graph-based algorithms to outperform the existing CPIT solution approaches in the literature.

The development of advanced mining optimisation approaches is an active research topic in expert and intelligent systems, especially for Australian mining industry. Main Australian mining companies such as BHP Billiton, Rio Tinto, Xstrata and OZ Minerals, are keen to adopt expert systems (e.g., commercialised mining software such as Whittle Gemcom's strategic planning software; XPAC's mine block sequencing software; and Modular's truck fleet dispatching software) for improving their mining management systems. However, in our recent visit to Australian mine sites, we observed that these commercialised expert systems still lack the advanced solution approaches in optimisation engine. As the CPIT problem is NP-hard, the required computational time of a MIP exact optimiser is increased exponentially. For solving large scale CPIT instances without any relaxation/decomposition schemes, a MIP exact optimiser such as IBM ILOG-CPLEX cannot be implemented due to memory overflow or unacceptable computational effort. Another practical reason is that the mining company is not willing to buy the third-party optimiser because the licence for commercial use is costly. To fill this gap, this study contributes to extend the boundaries of developing innovative numerical methods to solve large scale mining optimisation problems in a more efficient and effective way.

The remainder of this paper is outlined as follows. In Section 2, five lemmas on problem properties and the detailed procedures of two new algorithms are presented. In Section 3, the computational results of benchmark UPIT and CPIT instances obtained by the proposed algorithms are reported and compared to the best known results in MineLib. In the last section, we conclude the contribution and significance of this research in the last section.

2. New Algorithms

The following two fundamental mathematical programming models are given for showing the objective function and main constraints of the UPIT and CPIT problems respectively.

UPIT Model

Objective:

$$\text{Maximise:} \quad \sum_{b\in\mathcal{B}} x_b p_b \tag{1}$$

Subject to:

$$x_b \le x_{b'}, \quad \forall b \in \mathcal{B}; b' \in \Psi_b | \Psi_b \subset \mathcal{B} \tag{2}$$

$$x_b \in \{0,1\}, \quad \forall b \in \mathcal{B}; \tag{3}$$

where x_b is a binary decision variable that equals 1 if block b is selected; p_b is the value (positive or negative) if block b is to be mined; $\mathcal{B}$ is the set of total blocks for the whole orebody; Ψ_b is the subset of blocks that are the immediate predecessors of block b. Constraint (2) ensures that each block should be extracted after its predecessors. Constraint (3) defines that decision variables are binary.

CPIT Model

Objective:

$$\text{Maximise:} \quad \sum_{b\in\mathcal{B}} \sum_{t\in T} (y_{bt} - y_{b,t-1}) p_{bt} \tag{4}$$

Subject to:

$$y_{b,t-1} \le y_{bt}, \quad \forall b \in \mathcal{B}; t \in T \tag{5}$$

$$y_{bt} \le y_{b't}, \quad \forall b \in \mathcal{B}; b' \in \Psi_b | \Psi_b \subset \mathcal{B};\ t \in T \tag{6}$$

$$R_{rt}^{min} \le \sum_{b\in\mathcal{B}} (y_{bt} - y_{b,t-1}) u_{br} \le R_{rt}^{max}, \quad \forall t \in T; r \in \mathcal{R} \tag{7}$$

$$y_{bt} \in \{0,1\}, \quad \forall b \in \mathcal{B};\ t \in T \tag{8}$$

$$y_{b0} = 0, \quad \forall b \in \mathcal{B} \tag{9}$$

where p_{bt} is the discounted value of block b, which is calculated as $p_{bt} = \frac{p_b}{(1+\sigma)^t}$ and σ is the discount (interest) rate in the objection function (4). Constraint (5) ensures that each block is mined no more than once over time periods. Constraint (6) satisfies the precedence relationship. Constraint (7) requires that the minimum and maximum capacities/targets (R_{rt}^{min} and R_{rt}^{max}) of resource type r are satisfied in time period t, where u_{br} is the usage/feed of resource type r by block b. Constraint (8) states that decision variables are binary, i.e., y_{bt} that equals 1 if block b is mined **by** period t. Constraint (9) defines that each block should be ready by the start of the first period.

Unlike UPIT, CPIT is strongly NP-hard (Espinoza et al., 2013). Many researchers indicated that the substantial numbers of variables and constraints in industrial cases make the CPIT mathematical formulation model impossible for use in practice. Our computational experiments also justify the inapplicability of the CPIT formulation model in solving large scale benchmark CPIT instances. For example, for a benchmark CPIT instance called zuckSmall with 9400 blocks from MineLib, the number of constraints is 2,922,280 and the number of variables is 188,000 in the ILOG-CPLEX model of zuckSmall. As a result, no good solution can be obtained by IBM ILOG-CPLEX (Version 12.4 for academic use) after running over 24 hours on a desktop computer with Intel CORE i7-2600 (8 cores) at 3.40 GHz. Moreover, the ILOG-CPLEX model file (.lp format) of zuckSmall is too big (over 1GB) to be opened by Notepad++ or Microsoft Word.

Therefore, mainly based on network flow graph, machine scheduling and train scheduling theories, two new algorithms are developed to solve large scale CPIT instances in a more efficient and more effective way. For convenience, these two algorithms are called ***CPIT-Algorithm1*** and ***CPIT-Algorithm2*** in the paper.

The following notations will be used in property analysis and algorithm procedure.

Notations:

$\mathcal{B}^{UPIT}$	the set of total selected blocks in a UPIT result.
$\lvert\mathcal{B}^{UPIT}\rvert$	number of total selected blocks in a UPIT result.
$\mathcal{B}^{CPIT}$	the set of total decided blocks in a CPIT result, $\mathcal{B}^{CPIT} \subseteq \mathcal{B}^{UPIT}$.
$\lvert\mathcal{B}^{CPIT}\rvert$	number of total decided blocks in a CPIT result, $\lvert\mathcal{B}^{CPIT}\rvert \leq \lvert\mathcal{B}^{UPIT}\rvert$.

$\lvert T \rvert$	number of periods.
$\lvert \mathcal{R} \rvert$	number of resource types.
x_{bt}	equals 1 if block b is mined **in** period t; 0, otherwise. It is important to note the difference between x_{bt} and y_{bt}. For example, if block b is mined **by** the second period during a horizon of 4 periods, then $y_{b0} = 0$; $y_{b1} = 0$; $y_{b2} = 1$; $y_{b3} = 1$; and $y_{b4} = 1$. In the same case, the values of x_{bt} are $x_{b0} = 0$; $x_{b1} = 1$; $x_{b2} = 0$; $x_{b3} = 0$.
R_{rt}^{max}	maximum capacity of resource type r in period t.
R_{rt}^{-}	remaining capacity of resource type r in period t.
p_{bt}	discounted value of block b in period t; if $p_{bt} \geq 0$, then block b is called *a positive block*; else, block b is called *a negative block*.
S_b	a subset of blocks that contains an undecided positive block b with its all undecided (immediate and non-immediate) predecessors at the current stage. For convenient, S_b is called *a positive subset* associated with a positive block b.
U_{br}	usage of resource type r for S_b.
δ_{bt}	equals 1 if S_b can be assigned to period t under the condition that their resource usage is not more than the current remaining capacity for each recourse type; 0, otherwise.
S_{bt}	a positive subset of blocks associated with a positive block b that are assigned in period t.
B_t	a subset of blocks assigned to period t, $B_t = \bigcup_b S_{bt}$ and $\mathcal{B}^{CPIT} = \bigcup_t B_t$.

The proposed algorithms are based on the following properties of UPIT and CPIT.

Property 1:
The fundamental UPIT problem is equivalent to be a special type of the maximum-flow and minimum-cut network flow problem.

Proof: The UPIT problem can be transformed and modelled by a network flow graph $G_{UPIT} = (N1, A1)$, where $N1$ contains the set of total given blocks (vertices) associated with weights (net values of blocks) and $A1$ is the set of directed edges that represent the total given precedence relationship. Thus, the decision on which blocks to be selected to maximise the total value is equivalent to finding a maximum-weight closure set of vertices. For more analysis, please refer to Lerchs and Grossmann (1965), Hochbaum and Chen (2000) and Hochbaum (2001).

Property 2:
To solve CPIT more efficiently, the optimal UPIT result can be used to cut the subset of redundant vertices and edges in the construction of the corresponding CPIT graph.

Proof: The optimal UPIT and CPIT objective value can be respectively obtained by $\sum_{b\in\mathcal{B}^{UPIT}} x_b p_b$ and $\sum_{b\in\mathcal{B}^{CPIT}} x_{bt}\frac{p_b}{(1+\sigma)^t}$. Assuming that $S_{UPIT}|S_{UPIT} \subset \mathcal{B}^{UPIT}$ is the set of unselected blocks in an optimal UPIT solution, the summation of value $\sum_{b\in S_{UPIT}} p_b$ is negative. Thus, in the construction of a CPIT solution, the contribution from S_{UPIT} to the objective value of CPIT is also negative because the discount rate σ is always positive, implying that S_{UPIT} can be eliminated in the construction of CPIT graph without affecting the optimality of CPIT.

Property 3:
To obtain the optimal CPIT solution, the positive blocks should be mined as early as possible while the negative blocks should be mined as late as possible.

Proof: In a CPIT solution, the discounted value of block b is calculated by $\frac{p_b}{(1+\sigma)^t}$ if it is decided to be mined in period t. Therefore, to maximise the CPIT objective value determined by $\sum_{b\in\mathcal{B}^{CPIT}} x_{bt}\frac{p_b}{(1+\sigma)^t}$, the period index t should be as small as possible for the positive blocks and as large as possible for the negative blocks.

Property 4:
In the construction of a partial optimal CPIT solution in a period that also guarantees global optimality, it is enough to consider a concise subset of blocks in the precedence-satisfaction sequence.

Proof: Because the CPIT objective value is determined by $\sum_{b\in\cup_t B_t} x_{bt}\frac{p_b}{(1+\sigma)^t}$ and there exists such a requirement: $b' \in B_{t'}$ should be the predecessor of $b \in B_{t|t\geq t'}$ under resource capacity constraints, it is enough to consider a subset of blocks $B_{t'}$ in an earlier period t' in terms of precedence-satisfaction sequence.

Property 5:
To obtain an optimal CPIT solution, a positive subset with the largest value in terms of precedence-satisfaction sequence should be mined as early as possible.

Proof: According to equation ($\sum_{b\in S_{bt}} x_{bt} \frac{p_b}{(1+\sigma)^t}$) to determine the objective value of CPIT, the value $\sum_{b\in S_{bt}} \frac{p_b}{(1+\sigma)^t}$ is maximised under two conditions, that is, $\sum_{b\in S_{bt}} p_b$ should be as large as possible while $(1+\sigma)^t$ should be as small as possible.

Property 6:
A positive subset is not allowed to be mined in a period if there exists a bottleneck resource type under the condition: $r^* = arg_r(\bigcup_{b\in B_t} U_{br} > R_{rt}^{max})$.

Proof: The usage of a resource type r in a period t can be determined by $\bigcup_{b\in B_t} x_{bt} U_{br}$. In the construction of a CPIT solution in a period, the conditions $\bigcup_{b\in B_t} U_{br} \leq R_{rt}^{max} | \forall r$ should be satisfied. In this case, the bottleneck resource type is determined by $r^* = arg_r(\bigcup_{b\in B_t} U_{br} > R_{rt}^{max})$.

In the following, the main steps of CPIT-Algorithm1 and CPIT-Algorithm2 are presented.

CPIT-Algorithm1

Step 1: Set the graph model of UPIT: $G_{UPIT} = (N1, A1)$, where $N1$ contains the set of total given blocks (vertices) and $A1$ is the set of directed edges that satisfy the total given precedence relationship among vertices.

Step 2: Based on $G_{UPIT} = (N1, A1)$, apply the maximum-flow minimum-cut network flow algorithm to obtain the optimal UPIT result. For conciseness, please refer to Hochbaum and Chen (2000) and Hochbaum (2001) about the details of the well-known maximum-flow minimum-cut network flow algorithm.

Step 3: Based on the obtained UPIT solution, cut the redundant vertices and edges in G_{UPIT} for building the refined graph model for CPIT: $G_{CPIT} = (N2, A2)$, where $N2$ and $A2$ are the condensed set of vertices and edges respectively.

Step 4: Based on the $G_{CPIT} = (N2, A2)$, apply the topological sequencing algorithm that is based on conjunctive graph theory for machine scheduling (Liu and Ong, 2002 and 2004) to determine the precedence-satisfaction sequence of selected blocks, which are also in the non-increasing order of blocks' values.

4.1 Based on $G_{CPIT} = (N2, A2)$, set the list of current immediate predecessors and successors of each vertex.

4.2 Compute the in-count value (i.e. the number of immediate predecessors) of each vertex in the graph.

4.3 Decrease the in-count value for each of the immediate successor of the selected vertex by one after determining the current topological vertex.

4.4 If none of the undetermined vertices have a zero in-count value, stop running algorithm as the given graph model is cyclic (infeasible); else, select any of the undetermined vertices having a zero in-count value and put this vertex as the next candidate in the topological order.

4.5 Repeat Steps 4.1 and 4.4 until all vertices are determined.

Step 5: According to the achieved precedence-satisfaction sequence in Step 4, assign the positive subset of blocks period by period while the resources in each period are utilised as much as possible.

```
5.1 While $t < |T|$
5.2     Initialise the remaining capacity of each resource type $r$ in period $t$: $R^{-}_{rt} \leftarrow R^{max}_{rt}$.
5.3     for $b \leftarrow 1$ to $|\mathcal{B}^{UPIT}|$ in the precedence-satisfaction sequence obtained in Step 4
5.4         if $p_{bt} > 0$, then set the positive subset $S_b$ by determining all undecided predecessors of block $b$ at the current stage based on $G_{CPIT} = (N2, A2)$ and the current decision status of blocks; and calculate the usage ($U_{br}$) of each resource type $r$ for $S_b$; and initialise the resource-capacity condition: $\delta_{bt} \leftarrow 1$.
5.5         for $r \leftarrow 1$ to $|\mathcal{R}|$
```

```
5.6          if $U_{br} > R_{rt}^{-}$, then set the resource-capacity condition: $\delta_{bt} \leftarrow 0$ and
             break.
5.7        if $\delta_{bt} = 1$, then assign $S_b$ to period t; update the remaining capacity:
           $R_{rt}^{-} \leftarrow R_{rt}^{max} - U_{br} | \forall r$; and update the decision status of blocks in $S_b$.
5.8        Else, then go Step 5.9.
5.9      Update the result in period $t$.
5.10     Consider the next time period: $t \leftarrow t + 1$.
```

CPIT-Algorithm2

The CPIT-Algorithm2 is an improved version of CPIT-Algorithm1, with the purpose of improving the optimality performance. However, the computational effort of CPIT-Algorithm2 is much more than that of CPIT-Algorithm1. The main difference between CPIT-Algorithm1 and CPIT-Algorithm2 is only in Step 5. In CPIT-Algorithm1, the selection of positive blocks is according to the precedence-satisfaction sequence. In comparison, in CPIT-Algorithm2, the selection of positive blocks is determined both by the precedence-satisfaction sequence and the best positive set found by a local search to find the best accessible positive subset.

Pseudo-codes of Step 5 in CPIT-Algorithm2

```
5.1  While1 $t < |T|$
5.2    Initialise the result in period $t$ such as resource capacity: $R_{rt}^{-} \leftarrow R_{rt}^{max}$.
5.3    While a best positive subset can be found in period $t$
5.4      Initialise the list of positive subsets as empty.
5.5      for each undecided positive block $b$ in the precedence-satisfaction sequence
5.6        if block $b$ is not the successor of any blocks in the current list of positive
           subsets.
5.7          then set the positive subset $S_b$ by determining all undecided predecessors
             of block $b$ at the current stage based on $G_{CPIT} = (N2, A2)$ and the current
             decision status of blocks; add $S_b$ to the current list of positive subsets.
5.8      for each positive subset $S_b$ in the current list of positive subsets
5.9        determine the best positive subset $S_b^*$ that leads to the maximum value and
           also satisfies the resource capacity.
```

```
5.10      If the best positive subset is found, then assign the best positive subset $S_b^*$ to
          period t; update the current status such as remaining capacity in this period and
          the current decision status of assigned blocks.
5.11    Update the result in period $t$.
5.12    Consider the next time period: $t \leftarrow t + 1$.
```

It is noted that the data structure used for coding the above two algorithms plays an important role in computational efforts and CPU times. For example, an efficient data structure should be able to directly locate a block's full information and current status to avoid unnecessary indexing loops in the algorithm procedure. Otherwise, millions of unnecessary block-indexing loops may be generated in a function for a large scale CPIT instance, in which the number of blocks is usually over 10 thousand and a block regularly have over 30 immediate predecessors (e.g., see the benchmark mclaughlin_limit CPIT instance in MineLib).

3. Computational Results

In this section, we conduct extensive computational experiments and report computational results of benchmark large scale UPIT and CPIT instances, which are based on benchmark data files recently from MineLib (2013). The proposed algorithms were coded in C# programming language, compiled under Microsoft Visual Studio 2010 and run on a desktop computer with Intel Core i7 CPU (8 processors) at 3.4 GHz and 8 GB RAM under 64-bit Windows 7 Operating System.

The full data files of benchmark CPIT instances can be downloaded from the website: http://mansci-web.uai.cl/minelib/. For the description of data format, please refer to the recent paper about MineLib by Espinoza et.al (2013). Our computational experiments on benchmark MineLib UPIT and CPIT instances are concluded in Table 1.

<<Insert Table 1 here>>

In Table 1, the first column (*Instance Name*) displays the instance name from MineLib. The next three columns (*#B., #P., #R.*) indicate the problem size of each instance in terms of three main attributes, that is, the number of blocks, the number of periods and the number of resource types. The fifth column gives the optimal UIPT objective values obtained in CPIT-Algorithm1 and CPIT-Algorithm2 that are exactly same as the benchmark solutions in MineLib. The sixth column (*LP Upper Bound*) gives the LP upper bound of CPIT objective value from MineLib, which is determined by the LP relaxation model. The seventh column (*Best Know Solution*) provides the objective value of the best known CPIT solution from MineLib. In the eighth (*New Solution1*) and the ninth (*Time1*) columns, the objective values of these benchmark CPIT instances obtained by CPIT-Algorithm1 with their CPU times are presented. In the same format, the results of CPIT-Algorithm2 are shown in the last two columns (*New Solution2* and *Time2*).

The detailed results of benchmark CPIT instances obtained by CPIT-Algorithm1 and CPIT-Algorithm2 were exported by Microsoft Excel 2010. Each CPIT solution represented in an Excel Workbook file consists of several worksheets, each of which is regarded as a part of the full CPIT solution. The format of a complete CPIT solution using Instance newman1 is described in Appendix.

In comparison with the best known feasible solution in the seventh column (*Best Know Solution*), CPIT-Algorithm1 is able to obtain the better solutions of two instances (i.e., newman1 and zuck_medium). For other eight instances, CPIT-Algorithm1 can find the high-quality feasible solutions in a shorter time. For example, the exact MIP solver such as ILOG-CPLEX can only exactly solve one CPIT instance's MIP model (i.e., newman1 with 1060 blocks, 6 periods and 2 resource types) but with the CPU time of 27.55 seconds; in comparison, the CPU time of CPIT-Algorithm1 is 0.11 seconds. In Furthermore, if the computational time is allowed to be longer, the CPIT-Algorithm2 is a triumph as the better solutions of eight benchmark CPIT instances have been achieved. It is also observed that for only two benchmark CPIT instances (i.e., zuck_large and Mclaughin_limit), the better solutions cannot be found by CPIT-Algorithm2 due to complexity of precedence relationship in these two large scale instances. For example, the size of a data file about precedence relationship in Instance Mclaughin_limit is over 500 megabytes, because most blocks (among 112,687 blocks) have over 37 immediate predecessors. To optimally solve such two extremely complicated instances, it is essential to take much more computational efforts and develop more globally-diversified search methods. We remain it as a future challenge to us and other researchers in mine optimisation academic community.

4. Conclusion

In this paper, we proposed two new algorithms mainly based on our expertise of network flow and scheduling theory (e.g., Liu and Kozan, 2009, 2011a, 2011b and 2012; Kozan and Liu, 2011, 2012) in order to solve the large scale CPIT instances in a more efficient and effective way. Computational results based on the recent benchmark CPIT instances from MineLib (Espinoza et al., 2013) are conducted. In comparison with the best known solutions from MineLib, the superiority of proposed algorithms is validated. By taking advantage of UPIT and CPIT's properties, our proposed graph-based numerical algorithms can lead to the better (more efficient and applicable) mine scheduling optimisation expert systems, in which the third-party MIP optimiser is no longer indispensable and the random neighbourhood search under the mechanisms of any metaheuristics is also not required. The proposed algorithms have extended the boundaries of advanced mining optimisation approaches and would be promising to bring significant benefits for mine planning/scheduling engineers.

In comparison to other CPIT solution approaches in the literature, the strengths and weaknesses of our proposed algorithms are discussed as follows. Firstly, most CPIT solution techniques in the literature heavily relied on the use of third-party MIP optimiser software such as IBM ILOG-CPLEX with sophisticated relaxation/decomposition approaches (see Caccetta and Hill, 2003; Ramazan, 2007; Boland et al. 2009; Bley et al., 2010; Cullenbine et al., 2011; Chicoisne et al., 2012; Lambert et al., 2014; etc.). In comparison, the third-party MIP optimiser software is not required in the procedure of our proposed algorithms. Secondly, the development of metaheuristics such as Simulated Annealing, Genetic Algorithm or Ant Colony Optimisation algorithms for CPIT are not cutting-edge enough because of unexpected randomness in their neighbourhood search procedure (see Kumral and Dowd, 2005; Myburgh and Deb, 2010; Shishvan and Sattarvand, 2015). Instead of metaheuristic framework and neighbourhood structure, our proposed algorithms implement network flow graph solution techniques and utilise several fast decision rules through analysing the CPIT's critical problem properties (see the details in Section 2). On the other hand, there are two main challenges which are regarded the so-called weaknesses (or difficulties). One challenge is that unique expertise in the areas of scheduling theory and network flow theory are requisite in the algorithm development. Moreover, to guarantee the algorithm performance due to the problem size of a CPIT instance (e.g., over 100,000 block units), a state-of-the-art data structure is essential because of need to quickly identify indices of each block unit and its current immediate predecessors (e.g., over 37 predecessors) in the algorithm design. Our proposed graph-based algorithms outperform other CPIT solution approaches in the literature (as validated by computational experiments in Section 3), because the efficiency and effectiveness of our proposed algorithm are balanced in a better way.

Regarding the future research directions, we will further improve CPIT-Algorithm2 in order to find the better solutions of two unconquered instances (i.e., zuck_large and mclaughlin_limit). In addition, the proposed algorithms will be enhanced to solve the CPIT with multiple destinations also called PCPSP (Precedence Constrained Production Scheduling Problem) in MineLib. Moreover, the proposed CPIT/PCPSP model will be extended by including more practical constraints and attributes such as grade control, blending, rehandling and stockpiling costs. Finally, the proposed algorithms will be incorporated with a short-term mine equipment timetabling model (Kozan et al., 2013) within a demand-responsive expert system.

Glossaries:

UPIT	Ultimate Pit Limit
CPIT	Constrained Pit Limit
PCPSP	Precedence Constrained Production Scheduling Problem
MineLib	A public online library of benchmark instances' data files and best known results of mine optimisation problems including UPIT, CPIT and PCPSP

Acknowledgements

The authors would like to acknowledge the support of CRC ORE, established and supported by the Australian Government's Cooperative Research Centres Programme.

References

Alonso-Ayuso, A., Carvallo, F., Escudero, L.F., Guignard, M., Pi, J., Puranmalka, R., & Weintraub, A. (2014). Medium range optimization of copper extraction planning under uncertainty in future copper prices. European Journal of Operational Research. 233 (3), 711-726.

Bley, A., Boland, N., Fricke, C., & Froyland, G. (2010). A strengthened formulation and cutting planes for the open pit mine production scheduling problem. Computers & Operations Research, 37, 1641-1647.

Boland, N., Dumitrescu, L., Froyland, G., & Gleixner, A. M. (2009). LP-based disaggregation approaches to solving the open pit mining production scheduling problem with block processing selectivity. Computers & Operations Research, 36, 1064-1089.

Bienstock, D., & Zuckerberg, M. (2010). Solving LP relaxations of large-scale precedence constrained problems. In 14th International Conference IPCO (Integer programming and Combinatorial Optimisation) 2010 Lausanne, Switzerland Proceedings, Springer.

Caccetta, L., & Giannini, L. M. (1988). An application of discrete mathematics in the design of an open pit mine. Discrete Applied Mathematics, 21, 1-19.

Cullenbine, C., Wood, R. K., & Newman, A. M. (2011). A sliding time window heuristic for open pit mine block sequencing. Optimisation Letters, 5, 365-377.

Caccetta, L., & Hill, S. P. (2003). An application of branch and cut to open pit mine scheduling. Journal of Global Optimisation, 21, 1-19.

Chicoisne, R., Espinoza, D., Goycoolea, M., Moreno, E., & Rubio, E. (2012). A new algorithm for the open-pit mine production scheduling problem. Operations Research, 60(3), 517-528.

Epstein, R., Goic, M., Weintraub, A., Catalan, J., Santibanez, P., Urrutia, R., Aguayo, A. (2012). Optimizing long-term production plans in underground and open-pit copper mines. Operations Research, 60(1), 4-17.

Espinoza, D., Goycoolea, M., Moreno, E., & Newman, A. M. (2013). MineLib: a library of open pit mining problems. Annals of Operations Research 206, 93-1114.

Ferland, J.A., Amaya, J., & Djuimo, M.S. (2007). Application of a particle swarm algorithm to the capacitated open pit mining problem. Studies in Computational Intelligence, 76, 127-133.

Hochbaum, D. S., & Chen, A. (2000). Performance analysis and best implementations of old and new algorithms for the open-pit mining problem. Operations Research, 48(6), 894-914.

Hochbaum, D. S. (2001). A new-old algorithm for minimum-cut and maximum-flow in closure graphs. Networks, 37(4), 171-193.

Kozan, E., & Liu, S. Q. (2011). Operations research for mining: A classification and literature review. ASOR Bulletin, 30(1), 2-23.

Kozan, E., & Liu, S. Q. (2012). A demand-responsive decision support system for coal transportation. Decision Support Systems, 54, 665-680.

Kozan, E., Liu, S. Q., & Wolff, R. (2013). A short-term production scheduling methodology for open-pit mines. Paper presented at International Symposium on the 36th Applications of Computers and Operations Research in the Mineral Industry (APCOM), Brazil.

Lambert, W. B., Brickey, A., Newman, A. M., & Eurek, K. (2014). Open-pit block-sequencing formulations: a tutorial. Interfaces, 44(2), 127-142.

Lambert, W. B., & Newman, A. W. (2014). Tailored lagrangian relaxation for the open pit block sequencing problem. Annals of Operations Research, 222, 419-438.

Lamghari, A., & Dimitrakopoulos, R. (2012). A diversified tabu search approach for the open-pit mine production scheduling problem with metal uncertainty. European Journal of Operational Research, 222, 642-652.

Lamghari, A., Dimitrakopoulos, R., & Ferland, J. A. (2015). A hybrid method based on linear programming and variable neighborhood descent for scheduling production in open-pit mines. Journal of Global Optimisation, DOI 10.1007/s10898-10014-10185-z.

Lerchs, H., & Grossmann, I. F. (1965). Optimum design of open-pit mines. Transactions on CIM, LXVIII, 17-24.

Liu, S. Q., & Ong, H. L. (2002). A comparative study of algorithms for the flow shop scheduling problem. Asia-Pacific Journal of Operational Research, 19, 205-222.

Liu, S. Q., & Ong, H. L. (2004). Metaheuristics for the mixed shop scheduling problem. Asia-Pacific Journal of Operational Research, 21(4), 97-115.

Liu, S. Q., & Kozan, E. (2009). Scheduling a flow-shop with combined buffer conditions. International Journal of Production Economics, 117, 371-380.

Liu, S. Q., & Kozan, E. (2011a). Optimising a coal rail network under capacity constraints. Flexible Service and Manufacturing Journal, 23, 90-110.

Liu, S. Q., & Kozan, E. (2011b). Scheduling trains with priorities: a no-wait blocking parallel-machine job-shop scheduling model. Transportation Science, 45(2), 175-198.

Liu, S. Q., & Kozan, E. (2012). A hybrid shifting bottleneck procedure algorithm for the parallel-machine job-shop scheduling problem. Journal of the Operational Research Society, 63(2), 168-182.

Kumral, M., & Dowd, P. A. (2005). A simulated annealing approach to mine production scheduling. Journal of the Operational Research Society, 56, 922-930.

Martinez, M. A., & Newman, A. M. (2011). A solution approach for optimizing long- and short-term production scheduling at LKAB's Kiruna mine. European Journal of Operational Research, 211, 184-197.

MineLib (2013) http://mansci-web.uai.cl/minelib/.

Myburgh, C., & Deb, K. (2010). Evolutionary algorithms in large scale open pit mine scheduling. Paper presented at the GECCO'10, Portland, Oregon, USA.

Ramazan, S. (2007). The new Fundamental Tree Algorithm for production scheduling of open pit mines. European Journal of Operational Research, 177, 1153-1166.

Shishvan, M. S., & Sattarvand, J. (2015). Long term production planning of open pit mines by ant colony optimization. European Journal of Operational Research, 240, 825-836.

Souza, M. J. F., Coelho, I. M., Ribas, S., Santos, H. G., & Merschmann, L. H. C. (2010). A hybrid heuristic algorithm for the open-pit-mining operational planning problem. European Journal of Operational Research, 207, 1041-1051.

Underwood, R., & Tolwinski, B. (1998). A mathematical programming viewpoint for solving the ultimate pit problem. European Journal of Operational Research, 107(1), 96-107.

Table 1: Computational results of benchmark MineLib UPIT and CPIT instances

Instance Name	Problem Size			MineLib UPIT, Algorithms 1, 2*	MineLib CPIT		CPIT-Algorithm1**		CPIT-Algorithm2***	
	#B.	#P.	#R.	Optimal Solution	LP Upper Bound	Best Known Solution	New Solution1	Time1 (s)	New Solution2	Time2 (s)
newman1	1060	6	2	**26,086,899**	24,486,184	23,483,671	**23,899,187**	0.11	**24,147,400**	0.57
zuck_small	9400	20	2	**1,422,726,898**	854,182,396	788,652,600	759,266,499	4.43	**828,767,123**	23.69
kd	14153	12	1	**652,195,037**	409,498,555	396,858,193	378,434,056	7.52	**400,172,931**	153.65
zuck_medium	29277	15	2	**1,075,124,490**	710,641,410	615,411,415	**627,406,624**	68.21	**669,884,356**	5812.13
p4hd	40947	10	2	**293,373,256**	247,415,730	246,138,696	210,912,953	49.97	**247,206,003**	1622.31
marvin	53271	20	2	**1,415,655,436**	863,916,131	820,726,048	772,144,782	59.62	**849,609,951**	1032.06
w23	74260	12	3	**510,973,998**	400,653,199	392,226,063	307,995,791	126.48	**399,909,961**	5261.93
zuck_large	96821	30	2	**122,220,280**	57,389,094	56,777,190	47,964,706	227.05	50,495,599	10870.76
sm2	99014	30	2	**2,743,603,730**	1,648,051,083	1,645,242,774	1,310,832,727	182.66	**1,646,392,582**	2004.59
Mclaughlin_l.	112687	15	1	**1,495,726,474**	1,078,979,501	1,073,327,197	851,487,048	332.06	9,814,511,41	55731.09

*: the optimal UPIT solutions of all instances obtained by Algorithms 1 and 2 are exactly same as the optimal UPIT solutions in MineLib.

**: the better CPIT solutions of two instances (newman1and zuck_medium) are found by CPIT-Algorithm1 in a shorter CPU time.

***: the better CPIT solutions of eight instances are found by CPIT-Algorithm2 but in a longer CPU time.

Appendix

A complete CPIT solution of Instance newman1 in terms of Microsoft-Excel format is described in the below. Each CPIT solution represented in an Excel Workbook file consists of the following worksheets, each of which is regarded as a part of the full CPIT solution.

Worksheet 1: "General"

Worksheet 1 is named as "General", in which the instance name, problem size, CPU time, objective value, and number of decided (positive) blocks are given in this Excel worksheet. For example, the "General" worksheet of newman1 instance is shown in Table 2.

Table 2: The "General" worksheet of newman1 CPIT solution

Instance Name	Problem Size	CPU Time (s)	Objective Value of CPIT	Number of Total Decided Blocks	Number of Total Decided Positive Blocks
1_newman1_CPIT	1060 blocks; 6 periods; 2 resources	0.11	23,899,187	1059	545

Worksheet 2: "GeneralResultOfPeriods"

Worksheet 2 is named as "GeneralResultOfPeriods", in which the given resources' capacities, resources' usage rates, total (discounted) value, number of (positive) blocks assigned in each period are given in this Excel worksheet. For example, the "GeneralResultOfPeriods" worksheet of newman1 instance is presented in Table 3.

Table 3: The “Periods” worksheet of newman1 instance*

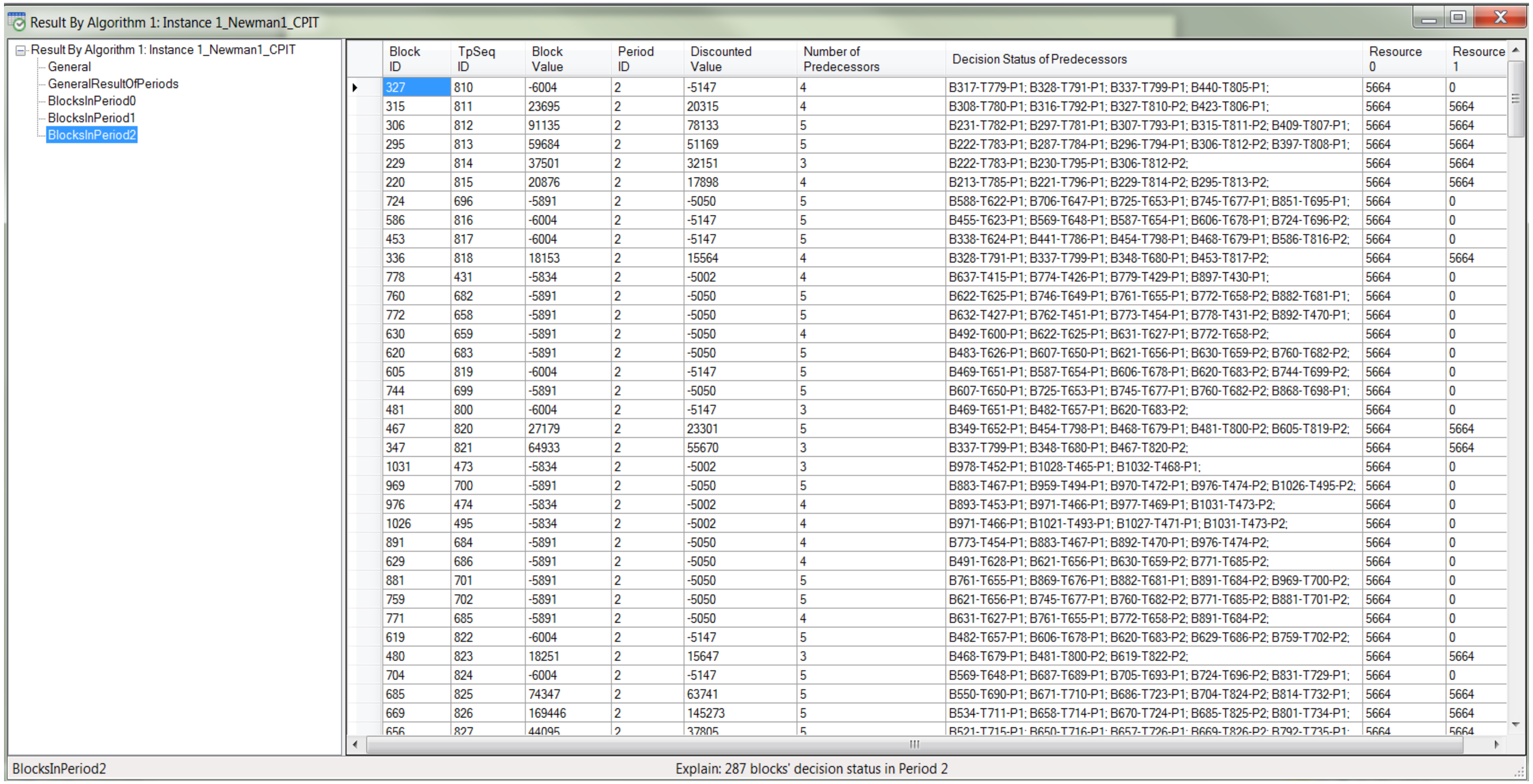
Period ID	Resources Capacities	Resources’ Usage Rate	Remaining Unused Capacities	Total Value	Total Discounted Value	#Blocks	#PBlocks
0	R0:2000000; R1:1100000;	R0:1998480 (99.92%); R1:1086583 (98.78%);	R0:1520; R1:13417;	6106288	6106288	416	217
1	R0:2000000; R1:1100000;	R0:1992924 (99.65%); R1:850938 (77.36%);	R0:7076; R1:249062;	9662825	8947060	356	144
2	R0:2000000; R1:1100000;	R0:1624608 (81.23%); R1:1081824 (98.35%);	R0:375392; R1:18176;	10317786	8845838	287	184

*#Blocks: number of blocks assigned in a period; #PBlock: number of positive blocks assigned in a period.

Worksheets 3-5: “BlocksInPeriod_t” (t=0, 1, 2)

For the solution of newman1 CPIT instance, all of 1059 blocks are decided to be mined in three periods (i.e., Periods 0, 1 and 2). In this case, Worksheets 3-5 are named as “BlocksInPeriod0”, “BlocksInPeriod1” and “BlocksInPeriod2” respectively. For example, Table 4 (Worksheet 5) shows the decision status of blocks assigned in Period 2. Due to the page limit, only a sample output using the graphical user interface of our developed mining optimisation software is illustrated in Table 4, in which the block index, assigned in-period index (e.g., Period 2), precedence-satisfaction sequence index, block value, discount block value, number of immediate predecessors, decision status of immediate predecessors and resource usages of each block are presented. As shown in Table 4, there are 287 blocks assigned in Period 2 and each row represents one block’s decision status with its immediate predecessors’ decision status.

Table 4: A sample output of “BlocksInPeriod2” worksheet of newman1 CPIT instance


Result By Algorithm 1: Instance 1_Newman1_CPIT
Result By Algorithm 1: Instance 1_Newman1_CPIT
General
GeneralResultOfPeriods
BlocksInPeriod0
BlocksInPeriod1
BlocksInPeriod2
BlocksInPeriod2

Block ID	TpSeq ID	Block Value	Period ID	Discounted Value	Number of Predecessors	Decision Status of Predecessors	Resource 0	Resource 1
327	810	-6004	2	-5147	4	B317-T779-P1; B328-T791-P1; B337-T799-P1; B440-T805-P1;	5664	0
315	811	23695	2	20315	4	B308-T780-P1; B316-T792-P1; B327-T810-P2; B423-T806-P1;	5664	5664
306	812	91135	2	78133	5	B231-T782-P1; B297-T781-P1; B307-T793-P1; B315-T811-P2; B409-T807-P1;	5664	5664
295	813	59684	2	51169	5	B222-T783-P1; B287-T784-P1; B296-T794-P1; B306-T812-P2; B397-T808-P1;	5664	5664
229	814	37501	2	32151	3	B222-T783-P1; B230-T795-P1; B306-T812-P2;	5664	5664
220	815	20876	2	17898	4	B213-T785-P1; B221-T796-P1; B229-T814-P2; B295-T813-P2;	5664	5664
724	696	-5891	2	-5050	5	B588-T622-P1; B706-T647-P1; B725-T653-P1; B745-T677-P1; B851-T695-P1;	5664	0
586	816	-6004	2	-5147	5	B455-T623-P1; B569-T648-P1; B587-T654-P1; B606-T678-P1; B724-T696-P2;	5664	0
453	817	-6004	2	-5147	5	B338-T624-P1; B441-T786-P1; B454-T798-P1; B468-T679-P1; B586-T816-P2;	5664	0
336	818	18153	2	15564	4	B328-T791-P1; B337-T799-P1; B348-T680-P1; B453-T817-P2;	5664	5664
778	431	-5834	2	-5002	4	B637-T415-P1; B774-T426-P1; B779-T429-P1; B897-T430-P1;	5664	0
760	682	-5891	2	-5050	5	B622-T625-P1; B746-T649-P1; B761-T655-P1; B772-T658-P2; B882-T681-P1;	5664	0
772	658	-5891	2	-5050	5	B632-T427-P1; B762-T451-P1; B773-T454-P1; B778-T431-P2; B892-T470-P1;	5664	0
630	659	-5891	2	-5050	4	B492-T600-P1; B622-T625-P1; B631-T627-P1; B772-T658-P2;	5664	0
620	683	-5891	2	-5050	5	B483-T626-P1; B607-T650-P1; B621-T656-P1; B630-T659-P2; B760-T682-P2;	5664	0
605	819	-6004	2	-5147	5	B469-T651-P1; B587-T654-P1; B606-T678-P1; B620-T683-P2; B744-T699-P2;	5664	0
744	699	-5891	2	-5050	5	B607-T650-P1; B725-T653-P1; B745-T677-P1; B760-T682-P2; B868-T698-P1;	5664	0
481	800	-6004	2	-5147	3	B469-T651-P1; B482-T657-P1; B620-T683-P2;	5664	0
467	820	27179	2	23301	5	B349-T652-P1; B454-T798-P1; B468-T679-P1; B481-T800-P2; B605-T819-P2;	5664	5664
347	821	64933	2	55670	3	B337-T799-P1; B348-T680-P1; B467-T820-P2;	5664	5664
1031	473	-5834	2	-5002	3	B978-T452-P1; B1028-T465-P1; B1032-T468-P1;	5664	0
969	700	-5891	2	-5050	5	B883-T467-P1; B959-T494-P1; B970-T472-P1; B976-T474-P2; B1026-T495-P2;	5664	0
976	474	-5834	2	-5002	4	B893-T453-P1; B971-T466-P1; B977-T469-P1; B1031-T473-P2;	5664	0
1026	495	-5834	2	-5002	4	B971-T466-P1; B1021-T493-P1; B1027-T471-P1; B1031-T473-P2;	5664	0
891	684	-5891	2	-5050	4	B773-T454-P1; B883-T467-P1; B892-T470-P1; B976-T474-P2;	5664	0
629	686	-5891	2	-5050	4	B491-T628-P1; B621-T656-P1; B630-T659-P2; B771-T685-P2;	5664	0
881	701	-5891	2	-5050	5	B761-T655-P1; B869-T676-P1; B882-T681-P1; B891-T684-P2; B969-T700-P2;	5664	0
759	702	-5891	2	-5050	5	B621-T656-P1; B745-T677-P1; B760-T682-P2; B771-T685-P2; B881-T701-P2;	5664	0
771	685	-5891	2	-5050	4	B631-T627-P1; B761-T655-P1; B772-T658-P2; B891-T684-P2;	5664	0
619	822	-6004	2	-5147	5	B482-T657-P1; B606-T678-P1; B620-T683-P2; B629-T686-P2; B759-T702-P2;	5664	0
480	823	18251	2	15647	3	B468-T679-P1; B481-T800-P2; B619-T822-P2;	5664	5664
704	824	-6004	2	-5147	5	B569-T648-P1; B687-T689-P1; B705-T693-P1; B724-T696-P2; B831-T729-P1;	5664	0
685	825	74347	2	63741	5	B550-T690-P1; B671-T710-P1; B686-T723-P1; B704-T824-P2; B814-T732-P1;	5664	5664
669	826	169446	2	145273	5	B534-T711-P1; B658-T714-P1; B670-T724-P1; B685-T825-P2; B801-T734-P1;	5664	5664
656	827	44095	2	37805	5	B521-T715-P1; B650-T716-P1; B657-T726-P1; B669-T826-P2; B792-T735-P1;	5664	5664

Explain: 287 blocks' decision status in Period 2