GEOVIA WHITTLE

PSEUDOFLOW METHOD FOR PIT OPTIMIZATION

White Paper

ABSTRACT

GEOVIA Whittle has implemented a new pit optimization engine based on the pseudoflow algorithm. This pseudoflow algorithm creates the same optimal pits achieved using the traditional Lerchs-Grossmann algorithm (LG), but with far more time efficiency. The LG method of pit optimization has been the industry standard and it is understood that strategic mine planners will be reluctant to trust a new method. To address their concerns, this paper explains the mathematical concepts on which pseudoflow has been built and how this has been implemented to solve mining problems. A comparative study with LG is detailed, showing the improved performance of pit optimization using pseudoflow.

AUTHORS:

Victor Xiaoyu Bai
George Turczynski
Nathaniel Baxter
David Place
Hamish Sinclair-Ross
Shawn Ready

FROM LERCHS-GROSSMANN TO PSEUDOFLOW: A SHORT REVIEW

The general pit optimization procedure works based on two inputs: block values and pit slopes, where the slopes introduce constraints on removal precedence of the blocks. The output of optimization is essentially a selection of blocks representing pits of valid slopes that yield maximum profit. Investigations for solving the pit optimization problem using a computer algorithm started in the 1960s. The Lerchs-Grossmann algorithm [1] was published in 1965 and was one of the earliest methods to produce the optimal pit. In the 1980s, the first industrial package with the LG algorithm was implemented in Whittle Three-D. The LG method has become the industry standard for pit optimization and also part of the university syllabus for mining engineers. The main issue with the LG method is the significant amount of time that is required to determine the optimal pit as the block models and pits increase in size and scale.

After the publication of LG, finding the optimal pit was no longer a challenging task. In academia, significant effort has been focused on searching for more efficient pit optimization algorithms. Many promising alternatives have been delivered. In 1976, Picard proved that the pit optimization problem could be solved with more efficient maximum flow algorithms [2]. In 1988, Goldberg and Tarjan developed a highly efficient maximum flow algorithm called the Push-Relabel method [3]. Notably, in 2008, Hochbaum published a pseudoflow algorithm [4], which was demonstrated to be more efficient than the LG and other prevalent maximum flow algorithms, such as the Push-Relabel method [5], [6]. GEOVIA has recognized the power of the pseudoflow algorithm and has developed a unique version for GEOVIA Whittle.

MATHEMATICAL CONCEPTS BEHIND PSEUDOFLOW

Understanding how the pseudoflow algorithm works requires a deep knowledge of mathematics and computer science. It involves two layers of questions:

1) How to model pit optimization with mathematical concepts, such as "set" and "graph"

2) How the algorithm solves the mathematical (graph) problems

Understanding the first question requires an introduction to some "graph" concepts. The second question is a specialized question in operational research and will not be covered in this paper. More detailed explanations of these questions can be found in reference papers [4] and [6]. The following section addresses the first question.

Graph Concepts and Pit Optimization

The pit optimization process typically uses a block model with fixed block values as an input. The pit slopes requirement and mining sequence can be expressed by the dependencies among blocks. For example, in Figure 1-1, a simple 2D block model consists of 10 blocks indexed from "a" to "j" (the value is marked on the top-right corner of a block). To maintain 45 degree slopes, a typical block dependency is like that shown in Figure 1-2, i.e., to mine block "c", the blocks "g", "h", and "i" must be removed first. The optimization problem is to find a set of blocks that respects the block dependency constraints and gives the highest total block value.

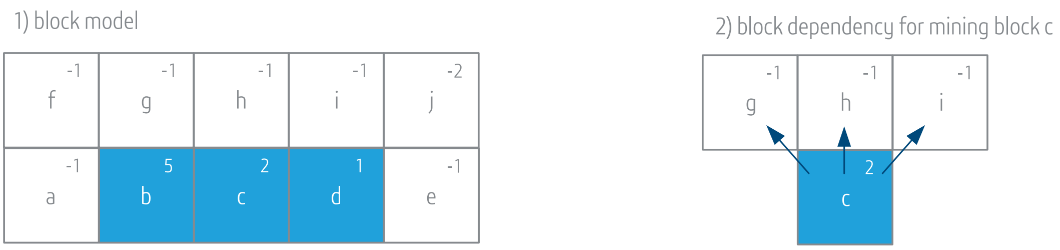

Figure 1. An example of block dependency in pit optimization

This problem is commonly represented by a mathematical concept called a "graph". A graph is a conceptual structure consisting of nodes and arcs. In the pit optimization case, a node represents a block, and an arc between two nodes represents the dependency relation of two blocks for the excavation sequence and slope constraint. A node can carry a weight value, to represent the value of the block. Figure 2 shows a graph representation of the pit optimization problem in Figure 1. We will use this example to demonstrate how to use the graph-based method to get the optimal pit. The concept is general and can be extended to more complex cases. Even for creating pits with a large 3D block model, the same process applies, but with an increased dimension and number of nodes and arcs.

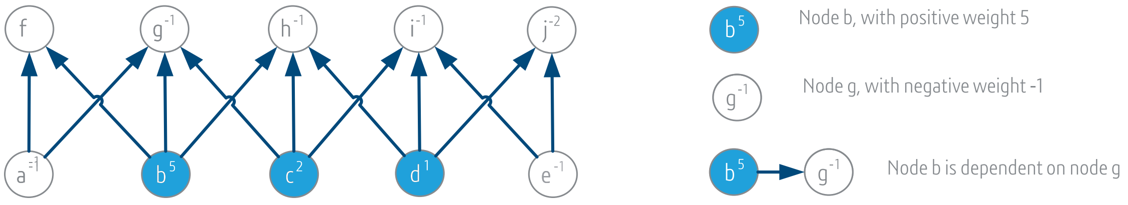

Figure 2. A graph representation of pit optimization problem in Figure 1

Maximum Closure and Optimal Pit

The precise definition of a pit with valid slopes is termed a "closed set" or "closure". It refers to a set of nodes that have no arcs out of the set. For example, in Figure 3, the set *{b, f, g, h}* is a closed set and *{d, h, i, j}* is another closed set; but set *{b, c}* is not closed, because *"b"* has available arcs to *"f", "g", "h";* and *"c"* has an available arc to *"g", "h", "i".*

A closed set of blocks is free to be removed and does not depend on the removal of other blocks. So, finding an optimal pit is the process of finding a closure with maximum total value. This problem is called a maximum closure problem. It is easy to observe that the optimal pit consists of block *{b, c, f, g, h, i}*, which gives total value 3. The Lerchs-Grossmann algorithm works by directly searching for the maximum closure.

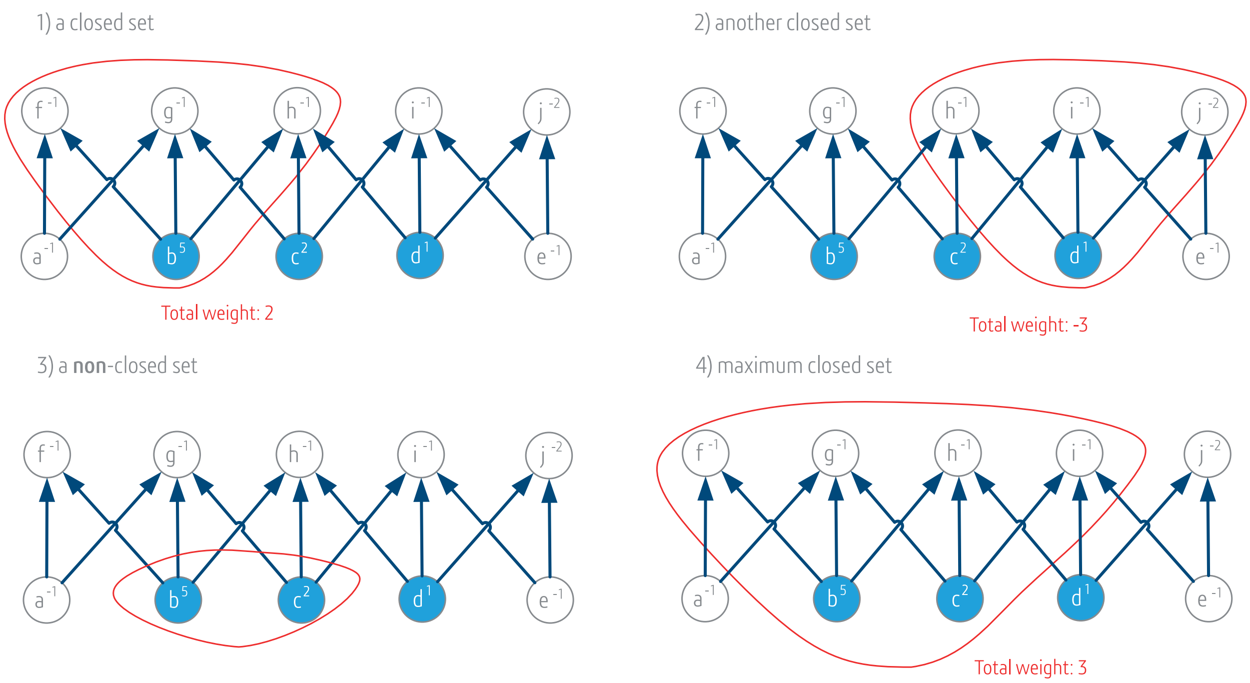

Figure 3. Closure and maximum closure in a graph

Maximum Flow Method—An Alternative to Generating an Optimal Pit

Research has proven that searching directly for the maximum closure is not the most efficient method of finding it. A more efficient method has been proven that involves solving a variant version of a graph, i.e., flow graph or flow network. For ease of understanding, an example of a flow graph would be a network of pipes for sending water from one city to another. A flow graph contains two additional special nodes, the source node (where the flow starts) and the sink node (where the flow finishes). Also, each arc, like a pipe, has a capacity property and allows a flow, up to the capacity limit, to pass through. The flow and capacity along an arc must be positive. The nodes represent a joining of pipes, so the amount of flow into a node must equal the total flow out of the node, which is called the flow balance criteria. In this network, searching for a flow distribution with maximum total flows that move into the sink node (or equally go out of the source node) is termed the maximum flow problem. It has been proved that the maximum flow problem is equivalent to the maximum closure problem [2].

To get a flow graph, we need to make a few changes to the graph in Figure 2:

- Add two special nodes: source node and sink node
- For all the existing arcs (blue), assign infinite capacities

- Add links from source to all positive nodes, with the capacities equal to the weight of the nodes
- Add links from negative nodes to sink, with the capacities equal to the absolute weight value of the nodes
- Remove the weights on nodes

The converted flow graph is shown in Figure 4-1. We also give an example of arbitrarily assigned flows that satisfy the flow balance, shown in Figure 4-2.

The relation between the flow and mining concepts is not as straightforward as the relation between a closure and a pit. One way to describe this is to consider the ore as the water stored in a source city that as much as possible needs to be sent to a destination city through a pipe network. The source station connects all the ore blocks, and the destination connects all the wastes. In the network, the economic value of a block is not reflected on a node, but is measured by the capacity of the pipe (arc) that connects it with the source or the destination city. Since the pipes representing block dependency have unlimited capacity, the bottlenecks of the networks are the pipes connected to the source or destination. Three types of pipes can be identified: "waste-to-destination", "source-to-ore", and "block-to-block". The flow assignment in each type of pipe can be interpreted in different mining senses, respectively:

1) Pushing flow from a "waste node" to a destination is similar to using the underlying ore value to pay for the waste block. When a flow saturates a pipe that links to the destination (flow amount equals the capacity), it means that the corresponding waste block can be paid off by its underlying ores, for example, the node *"f"*, "*g"* and *"h"* in Figure 4-2. Otherwise, if a "waste-to-destination pipe" is not saturated, then the waste block is not paid off by the ores, such as node *"a", "i", "j"* and *"e"* in Figure 4-2. The flow balance criteria imply that the flow that goes into a "waste node" cannot exceed its out-pipe capacity (absolute block value), thus guaranteeing that the waste block is not paid for multiple times.

2) Pushing flows from a source into an "ore node" means passing the ore value down, within the capacity limits, to pay for the necessary wastes. If a "source-to-ore" pipe is not saturated while at the same time the flow is balanced and no more can be pushed downstream, such as pipe *"s-b"* in Figure 4-2, this means that the ore is sufficient to pay off the overlying wastes and has residual value left. Otherwise, if an ore block is not high enough to pay for the linked wastes, then the pipe connected to the source would be saturated, such as pipe *"s-d"* in Figure 4-2.

3) The unlimited capacity pipes that link block-to-block guide the flow to the related "waste nodes", and allow the ore value passing through freely to pay for all the overlying wastes.

When the maximum flow is found, it ensures that all the ores have been utilized to pay for the necessary wastes. As an opposite example, in Figure 4-2, the node *"c"* is not considered to be passing any flow through. So the total flow is 4 and has not reached the maximum flow of 5 (shown in the following section), therefore additional distribution is needed to reach the maximum flow.

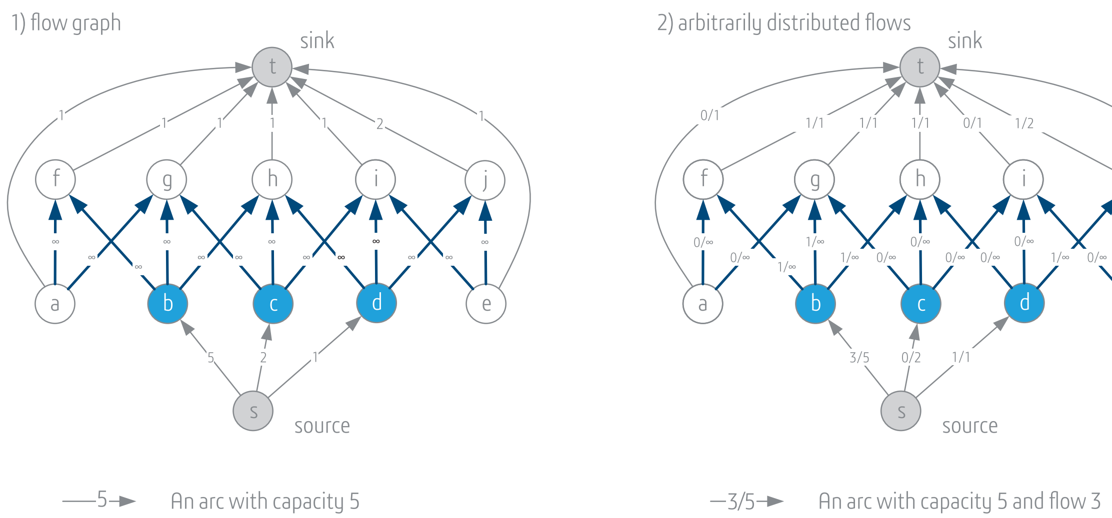

Figure 4. (1) Flow graph converted from the graph in Figure 2; (2) Arbitrarily distributed flows on Graph 4-1 that satisfies the flow balance criteria

Assume that a maximum flow solution is found as shown in Figure 5-1, by any possible method, say pseudoflow. To get the optimal pit, we still need to convert the maximum flow solution to a maximum closure. To do this, we first break the saturated arcs (blue dashed arcs in Figure 5-2). This separates the sink node from the waste blocks that can be paid off by the underlying ore blocks (such as blocks *"f", "g", "h" and "i"*), and also cuts the paths from the source node to the ore blocks that cannot afford the overlying wastes values (such as block *"d"*). Then we search all the nodes that can be reached by the source node, as shown in Figure 5-2. Those nodes (except the source) are the maximum closure set. The reason for doing this relates to the "max-flow min-cut theorem" [7] and the proof of equivalency of maximum flow and maximum closure in the research paper [2].

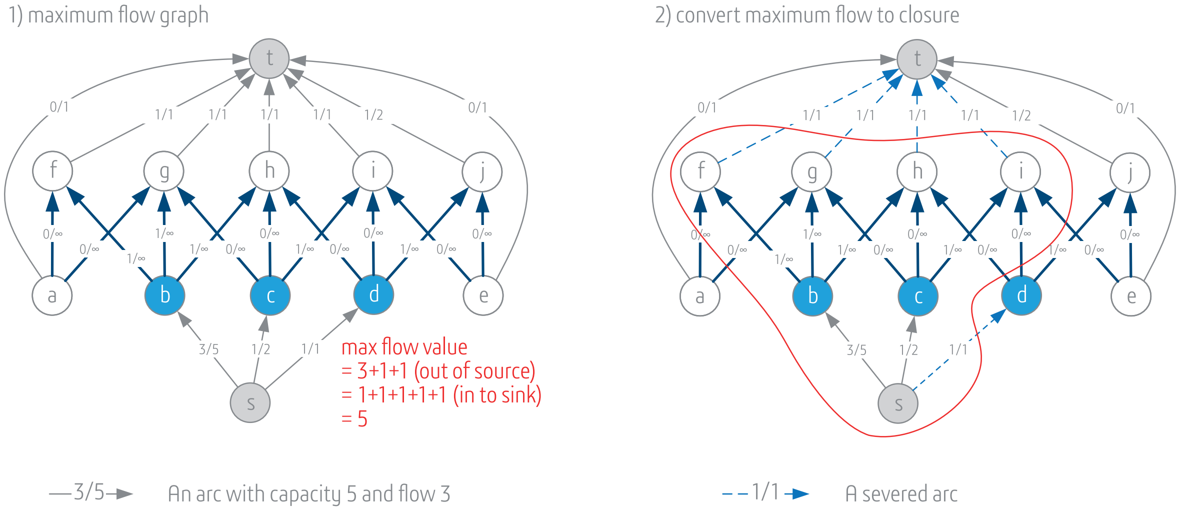

Figure 5. Finding optimal pits by in maximum flow graph

Maximum Flow Problem and Pseudoflow Algorithm

In the description above, we introduced how to model a pit optimization problem with the graph concept, specifically using maximum closure and maximum flow representations. We also noted that LG is a method to solve the maximum closure problem. The remaining question is, what is the procedure to find a maximum flow solution? In general, the procedure is to iteratively change the flows along the paths until the maximum flow is found. There are many maximum flow algorithms, and each algorithm uses different ways to distribute the flow with varying efficiency. The pseudoflow algorithm has been demonstrated to be one of the most efficient methods to date with respect to the time used to solve a defined problem set.

Understanding the procedure of pseudoflow and the reason for its outstanding efficiency needs very specialized mathematical knowledge. This document is not designed to be an exhaustive explanation of the pseudoflow algorithm, and more details are available in reference [4].

Pseudoflow Engine in GEOVIA Whittle

GEOVIA Whittle is powered by a new implementation of the pseudoflow algorithm with an optimized data structure. The new engine significantly speeds up the pit generation process compared to our traditional LG engine. A computation comparison of pseudoflow vs. LG is discussed below.

COMPUTATION COMPARISON AND APPLICATION CONCERNS

TESTING DATA AND PARAMETERS

To demonstrate the computation speed of the Whittle pseudoflow engine, a series of testing block models were used (see Table 1 and Figure 6). The 45 degree slope angle is adopted for all cases. The number of arcs created for this slope setting is listed in Table 1. Note that with Whittle, the actual number of blocks and arcs used in the optimization is called active blocks and active arcs. (Active blocks represent the blocks that contain parcels, and all the precedent blocks need to be removed to access the blocks with parcels).

Tests were done for two scenarios: one scenario uses one revenue factor (RF) to generate one pit shell; the other uses nine RFs to create nine pit shells. The creation of multiple pit shells is basically a repetition using the pseudoflow/LG across multiple RFs.

The test computer was a laptop with Intel Core i7 2.7GHz CPU and 32 GB RAM.

	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Model dimension	122x120x34	183x180x51	240x240x68	305x300x85	366x360x102	427x420x119
Number of Blocks	497,760	1,679,940	3,916,800	7,777,500	13,439,520	21,341,460
Number of Arcs	7,358,228	26,043,348	63,194,216	125,157,272	218,278,956	348,905,708
Number of Active Blocks	108,704	372,780	891,941	1,749,722	3,032,680	4,822,114
Number of Active Arcs	1,464,517	5,438,090	13,525,933	27,153,793	55,638,248	76,794,400

Table 1. Testing Data descriptions

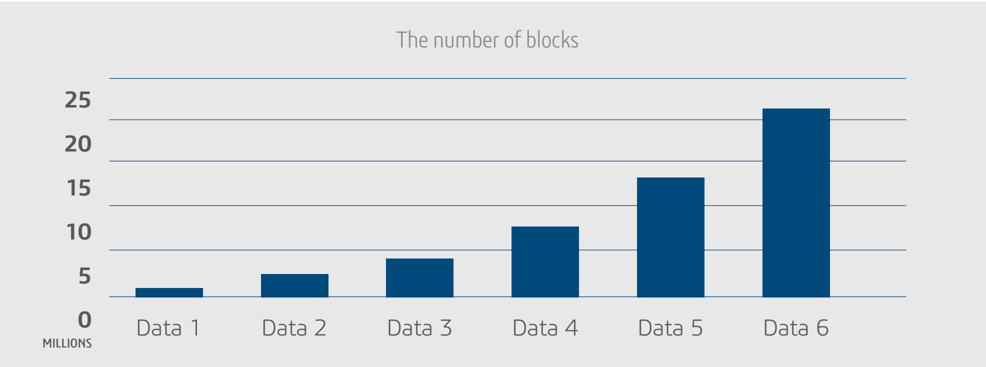

Figure 6. The size of testing data

Testing Results

The computation time for the datasets and parameters are plotted in Figures 7 and 8, and listed in Table 2. Note that the collected computation time reflects the overall process of pit optimization with Whittle, including reading and writing data, as well as pseudoflow/LG process. The pseudoflow engine is faster than LG in all cases. The boost of speed is more significant as the block model becomes larger, especially for the 21.3 million block case, with the time reduced from 15 hours to 12 minutes using pseudoflow. Also, when creating nine pit shells, the speed gain from pseudoflow for creating each single pit shell accumulates and shows an even more remarkable overall improvement.

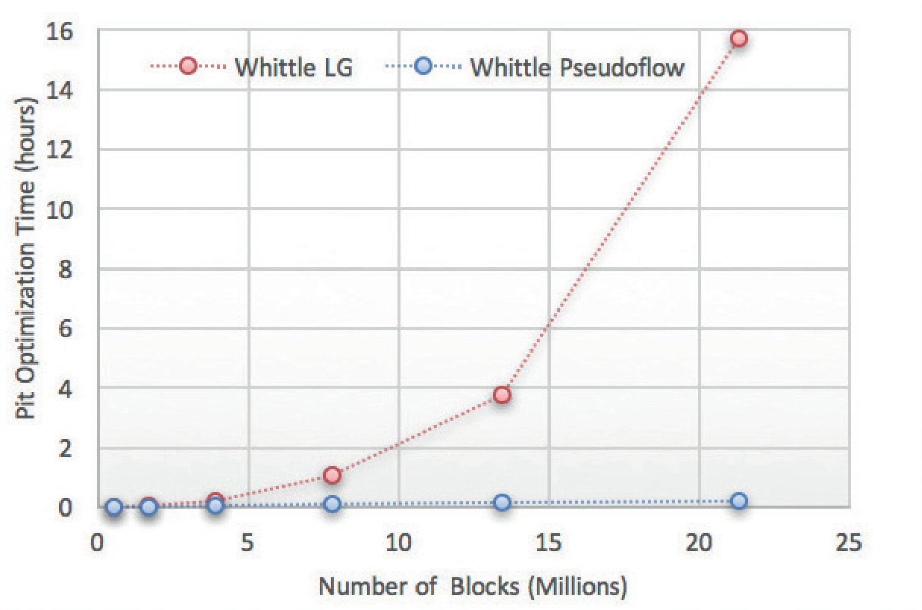

Figure 7. Computation time comparison for one revenue factor

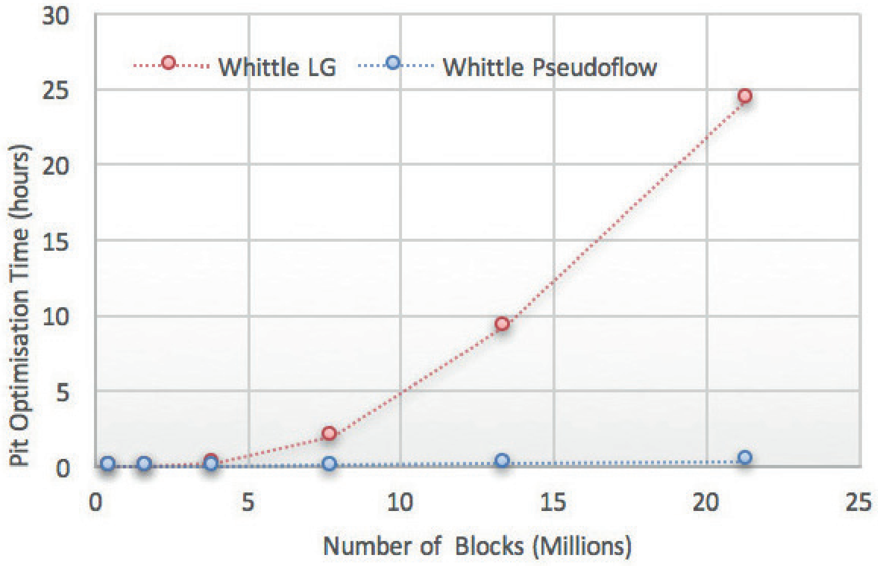

Figure 8. Computation time comparison for 9 revenue factors

		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
1RF	**Whittle LG**	17s	2m14s	12m55s	1h4m	4h45m	15h42m
	Whittle pseudoflow	16s	57s	1m17s	4m27s	7m52s	12m41s
	Time reduced	6%	57%	82%	93%	97%	99%
9RFs	**Whittle LG**	21s	1m49s	14m	2h1m	9h21m	24h21m
	Whittle pseudoflow	19s	1m4s	2m39s	5m26s	9m56s	20m
	Time reduced	10%	42%	82%	96%	98%	99%

Table 2. Computation time of pseudoflow vs. LG

Factors Impacting the Speed of Optimization

In general, the computation time of the optimization process can be impacted by a variety of factors including:

- The number of blocks
- The number of arcs, related to slope setting and block size
- Distribution of block values
- Computer hardware and system

The testing done here is not exhaustive for all factors, but focuses on showing the time comparison over different sized block models, which is usually the dominant factor. For some small or intermediate block models, the pseudoflow engine may not produce significant speed improvement over LG. The reason is that the LG engine is already very fast in solving these cases, and the majority of processing time is taken up by data reading/writing instead of optimization. But for larger sized block models, the speed improvements are significant.

Precision Question

With the recent exposure of pseudoflow in the mining industry, one common question continues to be raised: "Does pseudoflow always produce exactly the same result as LG?"

The answer is "Yes" and "No". Mathematically, "Yes", it has been proven that the pseudoflow algorithm and LG generate the same result. When it relates to software implementation, it is not always true. The reason is that the Whittle pseudoflow engine approximates the value of blocks as integers, while LG deals with them as floating point numbers. In both cases, using floating point numbers or integers, the block value encoding will introduce imprecision in the block value. The pseudoflow engine neglected the value in the scale of cents, which is usually marginal to the block value. In some rare cases, this approximation can result in a pit slightly different from the LG result. However, even if different pits occur, the pit values should be very close. In the context of strategic mine planning, considering that the actual block values have much greater uncertainty when comparing to the marginal value neglected here, the approximation of value hardly impacts on the NPV report and is definitely tolerable.

Memory Requirements

The pseudoflow engine utilizes more physical memory via RAM than LG does. For some large cases, the pseudoflow engine may reach the memory limit of the computer. In general, the memory usage grows almost linearly with the number of active arcs, as shown in Figure 9. The information of active arcs is reported in the pit optimization message tab for both LG and pseudoflow. Table 3 lists typical memory requirements to efficiently solve problems of different sizes (measured by the number of active arcs). Here, the term "efficiently" means "processing without using virtual memory". Using virtual memory can drastically slow down the optimization and therefore add significant time to the overall optimization process. On the other hand, a slightly larger case can also be solved by using a small amount of virtual memory, with a trade-off of speed. For example, with a 32 GB RAM computer, a problem with less than 509 million active arcs can be efficiently solved; and a problem of 550 million active arcs is still solvable by using virtual memory.

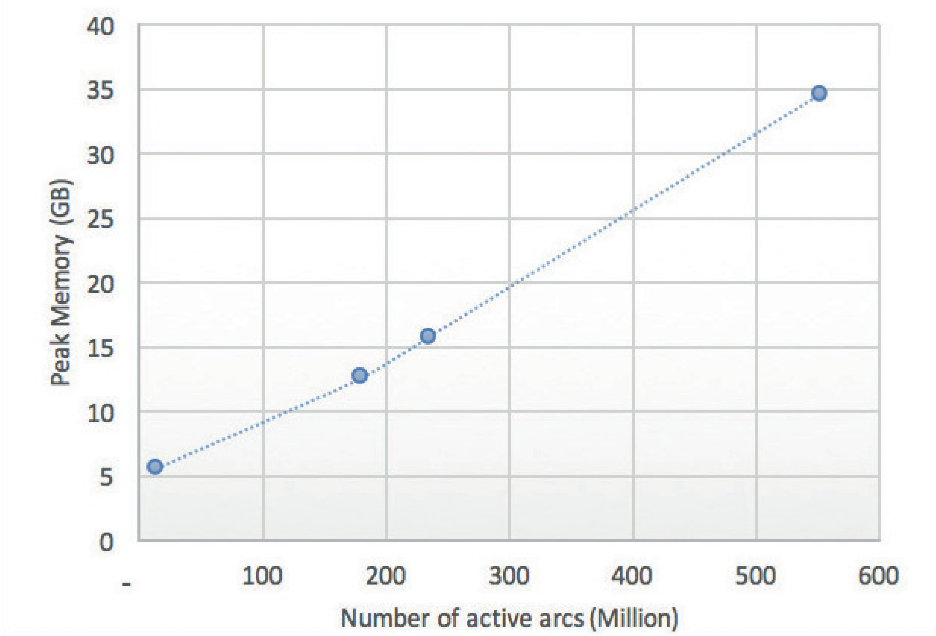

Figure 9. Memory consumption of pseudoflow engine for the cases of different number of active arcs

RAM (GB)	8	16	32	48	64
Active arcs (Millions)	103	238	509	779	1,049

Table 3. The pseudoflow process limit (in number of arcs) for computers of different memories

CONCLUSION

The pseudoflow algorithm is a fast new vehicle for delivering optimal pit solutions. In Whittle, the pseudoflow engine has inherited the same usability as the entrusted Pit Optimization Engine, which allows users to configure comprehensive practical slope settings for a variety of geotechnical needs, and achieves identical results to the LG method. Speed improvements open up the opportunity to solve problems that were previously too large for Whittle and the traditional LG engine. Furthermore, the pseudoflow algorithm also enables some interesting collaboration possibilities.

Dassault Systèmes is now connecting GEOVIA's offerings to the likes of SIMULIA® Process Automation and Simulation technologies on the **3D**EXPERIENCE® platform. This enables running hundreds to thousands of "What if?" scenarios and analyzing them within the same timeframe that it took to run a handful in the past. With the **3D**EXPERIENCE platform, it is possible to even further automate and improve the performance seen with GEOVIA Whittle, and GEOVIA continues searching for faster, practical and easy-to-use strategic mine planning solutions for the future.

REFERENCES

[1] H. Lerchs and I. F. Grossmann, "Optimum design of open-pit mines," Transactions CIM, vol. LXVIII, pp. 17–24, 1965.
[2] J. C. Picard, "Maximal closure of a graph and applications to combinatorial problems," Management Science, vol. 22, no. 11, pp. 1268–1272, 1976.
[3] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum-flow problem," Journal of the Association for Computing Machinery, vol. 35, no. 4, pp. 921–940, Oct. 1988.
[4] D. S. Hochbaum, "The Pseudoflow Algorithm: A New Algorithm for the Maximum-Flow Problem," Operations Research, vol. 56, no. 4, pp. 992–1009, Aug. 2008.
[5] B. G. Chandran and D. S. Hochbaum, "A Computational Study of the Pseudoflow and Push-Relabel Algorithms for the Maximum Flow Problem," Operations Research, vol. 57, no. 2, pp. 358–376, Jan. 2009.
[6] D. C. W. Muir, "Pseudoflow, New Life for Lerchs-Grossmann Pit Optimisation," in Orebody Modelling and Strategic Mine Planning Conference, Perth, Australia, 2004, vol. 14, pp. 97–104.
[7] C. H. Papadimitriou and K. Steiglitz, "The Max-Flow, Min-Cut Theorem,"
in Combinatorial Optimization: Algorithms and Complexity, Unabridged edition., Mineola, N.Y: Dover Publications, 1998, pp. 120–128.

By Victor Xiaoyu BAI, George TURCZYNSKI, Nathaniel BAXTER, David PLACE, Hamish SINCLAIR-ROSS, Shawn READY