Essays on Sequence Optimization in Block Cave Mining and Inventory Policies with Two Delivery Sizes

by

Anita Frances Parkinson

B.Sc.Eng., Queen's University, 1991
M.S.C.E.P., Massachusetts Institute of Technology, 1994
M.Sc.B.A., The University of British Columbia, 2002

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

The Faculty of Graduate Studies

(Business Administration)

The University Of British Columbia

August, 2012

Abstract

Chapter 1 is an introductory chapter for this thesis work. It sets the scene by describing the motivation and industrial setting for each project.

In Chapter 2, "Optimal Inventory Replenishment with Two Delivery Sizes", we consider a periodic review inventory system where a retailer can order in multiples of a fixed quantity Q_1, or multiples of $Q_2 = 2Q_1$, where the per unit material cost is less for ordering Q_2. We extend results of Veinott, and of Fangruo Chen, to show that an optimal replenishment policy has a reorder point R, as well as a second parameter controlling when the last order should be for Q_1 instead of Q_2 under a linear cost structure.

In Chapter 3, "Sequence Optimization in Block Cave Mining" we investigate the use of integer programming models to aid the practitioner in the early planning stages of a Block Cave Mine. Given the footprint of the ore body divided into draw points or grid squares, sequence optimization determines which draw points to open in which period to meet the physical constraints of the mining process and maximize the total net present value of the mine. Traditionally done by trial and error by experts in the field, this is a first attempt to use modelling techniques to automate and optimize the process. We develop three integer programming models and discuss the challenges of formulating the problem in this framework. Two additional models are developed for comparison, one using the Column Generation technique and one using a greedy or myopic algorithm. All models are run on two data sets provided by our industrial partner, and the performance and results are compared. This work demonstrates that integer programming models can generate opening sequences but, like many "real life" problems, this one is complicated.

Table of Contents

List of Tables

List of Figures

Acknowledgements

Thank you to my thesis advisors Tom McCormick and Maurice Queyranne who guided and stuck with me through all the ups, downs and gaps in my thesis process. A big thank you to Tony Diering of Gemcom who was willing to take on this project with me and taught me all I know about block cave mining. Thanks also to Brian Graham, who taught me many, many things and kept me smiling. As is often said, it is not the arrival that is important, but how you get there.

Dedication

This thesis is dedicated to all my fellow students who have struggled with the Ph.D. experience, and to all the family, friends and humourists who help us through it.
If you read this let me know firstname.lastname at gmail

> "I see my light come shining
>
> From the west unto the east
>
> Any day now, any day now
>
> I shall be released"
>
> Bob Dylan

Chapter 1

Introduction

This thesis contains two chapters of work that are essentially unrelated to each other. In choosing to work on both of them I was motivated by real industrial problems. In a previous life, I worked as a chemical engineer, the perfect blend of theory in the office, and actual implementation in the chemical plant. There is nothing like staring at a three storey tall evaporator to realize that getting the temperature value that is essential for the computer model, will be problematic. Theory and models ARE great, but the art is making it work, or extracting the understanding that you can use to make it work. When I returned to school to study Operations Research I chose a master's program that was focused on an industrial project. With this background, it is not surprising that I sought out problems with an industrial basis for this thesis.

Behind the inventory policy work (Chapter 2 Optimal Inventory Replenishment with Two Delivery Sizes), is a shipping problem. A company produces a product in the southern hemisphere and transports it by using a fleet of tanker ships to customers in Europe and North America; how should they schedule their tankers to meet expected demand? The classic Economic Order Quantity (EOQ) model sets a target inventory position and prescribes ordering (or delivering) enough to meet that target in each period. The target trades off between inventory holding costs and fixed delivery charges. Unless demand is constant, this results in different order quantities in each time period. In the setting of a fleet of tanker ships, the size of the available ships may not match the desired order quantity. Also, it would be expected that the delivery costs would vary with the size of the ship, typically showing economies of scale. The chapter investigates an ordering policy for a very simple fleet — one with an unlimited number of ships of two sizes, one twice the size of the other.

The second project (Chapter 3 Sequence Optimization in Block Cave Mining) addresses a problem from the mining industry; given that you are going to mine a given ore body, where do you start and what direction do you proceed? The current practice is a trial and error process where options are generated using expert knowledge and evaluated by a set of criteria.

This is a classic opportunity where the tools of operations research can be applied to try to capture some of the experience and art of decision making and proved to be an inviting project.

The planning of open pit mines has long been aided by integer programming models, but these models do not apply to block cave mines. In block cave mining, the ore body is approached from underneath. Blasting is done to start the break up of the rock in a given area called a draw point. When the blasted rock is removed, a cave is created. Over time the rock continues to break off from the top of the cave and is removed for processing. Sequence optimization plans where to do the initial blasting in each period to create a single cave that matches processing capabilities, and maximizes the total Net Present Value of the mine. By assuming a fixed removal rate from each draw point, the three dimensional ore body is reduced to a two dimensional mining footprint.

In this work we develop three integer programming models. The main challenge that emerges is the constraint that the open draw points form a single contiguous cave. This proved difficult in the integer programming framework, though two of the models were able to meet it. In addition, the column generation technique is used to move the problem of feasible cave generation out of the integer programming framework.

All of the models share three basic constraints. Two are straightforward to frame in the integer programming framework. The *Start Once* constraint ensures that each draw point is opened once and only once, and the *Global Capacity* constraint ensures the number of active draw points does not exceed the downstream processing capacity. The third constraint, that the opened draw points must form a single, contiguous group, or cave, is the source of the model variations.

In the first model, the Basic model, contiguity is enforced along a two dimensional grid. Two formulations to achieve this are discussed and one is selected for the model. Unfortunately, two dimensions are not sufficient to guarantee a single cave, as is shown on the sample data. This model does prove useful on our sample data as it provides an upper bound on the Total Net Present Value of the mine. The model runs to completion in a reasonable time period on all the sample data sets.

The Malkin model comes from Malkin and Wolsey, whose contributions are much appreciated. Their model also incorporates the preference of planning experts for caves with a diamond shape, which is based on the rock dynamics. A centre point is chosen, which establishes two perpendicular axes in a two dimensional grid. Contiguity is achieved by only allowing a draw point to open if those draw points between it and each axis are open.

This model runs well on all sample data sets, producing the single, contiguous cave in each period as desired. The challenge comes in choosing the centre point. For the smaller sample data set it is feasible to have a run with each draw point as the centre point. This is not feasible for the larger sample data set.

Achieving a diamond shape cave is also the driving force behind the 2Cone model in which a cave is defined by the intersection of two cone or triangle shaped groups of draw points. Each cone is defined by a vertex, and grows by two draw points on each subsequent row of a two-dimensional grid. The location of each vertex is a decision variable so is chosen by the model to maximize the Total Net Present Value of the mine. This model does not complete on all the sample data sets, but when it does, it produces a single contiguous cave in each period.

The difficulties in achieving a single, contiguous cave in each period within the integer programming framework gave rise to the use of the Column Generation technique to create the ColGen model. Briefly, this technique iterates between generating a set of feasible caves for each period and using integer programming techniques to select a cave in each period that maximizes the objective and meets the constraints. The appeal of this technique is that the feasible caves can be generated outside of the integer programming framework. Unfortunately, the technique fails to generate valuable opening sequences. Additional algorithms are generated which ensure that each feasible cave added by the technique has caves in preceding and following periods that overlap to form a feasible sequence. This model did not complete on all data sets, but did produce a feasible solution on the smaller sets.

In this chapter we compare the performance of the models on two data sets. All models perform well on the smaller data set of 236 draw points. The Basic model returns a sequence with the highest value, though two caves are generated in the early periods. The Malkin model returns a value that is 98% of the Basic, and the 2Cone and ColGen models come in at 95% of the the Basic model value. On this data set, the Basic model completes in under 10 seconds while the others take over an hour. We suggest that this speed shows a usefulness of the Basic model in setting an upper bound on the value, and indicating high value draw points for initial periods. The second data set is much larger at 6510 draw points. On this data set, the ColGen and 2Cone models failed to complete in under a week. After 19 hours the Basic model returned a sequence which again had multiple caves in the early periods. It was not feasible to run the Malkin model on all draw points, but the highest valued of those selected reached 98.8% of the Basic model value.

The selection of individual Malkin models runs varied from 1 to 9 hours to complete. While these running times may seem prohibitively long to many researchers, in the workplace it may be feasible to run a model over night, or over a weekend if a single run produces useful results. Again, we suggest that the Basic model is useful in producing an upper bound against which the selection of Malkin runs can be evaluated.

Chapter 2

Optimal Inventory Replenishment with Two Delivery Sizes

2.1 Introduction

Consider a retailer facing uncertain demand who periodically replenishes her stock from a manufacturer (or distributor, or ...). Classic models assume that the quantity ordered can be any number of units. For simple version of such models, often one can find replenishment policies that are optimal, in the sense of minimizing long-run average cost. In the absence of fixed costs for ordering, such policies are often of the form of having an order "up to level", or reorder point, R. This means that if the retailer observes that her stock is below R at an ordering epoch, she orders up to R; otherwise, she orders nothing.

In reality, the order size is often limited to, e.g., multiples of a full truckload. Suppose that a full truckload is Q units, so that all orders must be integral multiples of Q. In this case, Veinott (1965) showed that, for an appropriate R, a variant of an order up to level policy is optimal: In each period, the retailer should order the smallest multiple of Q necessary to bring her inventory position to at least R. Later, Chen (2000) extended this result to a multi-echelon setting. In these cases the per-unit cost of items is always the same, so material cost can be ignored.

But in many practical situations, there is a variety of order sizes possible. For example, there could be both large and small trucks. In a consulting project we did with a bulk chemical supplier, there were several different sizes of ocean tankers available for shipping their product. This naturally raises the question of what optimal ordering policies would be in such cases.

This paper considers a stylized version of such a problem. We assume that there are two delivery sizes available, Q_1 (small truck), and Q_2 (big

truck), with $Q_2 = 2Q_1$. We defend this simple model by pointing out that even this simplest generalization turns out to be quite hard to solve. Denote the per-unit cost of one item on a big truck as c_2, and on a small truck as c_1, so that the total cost of ordering a big truckload is c_2Q_2. We make the reasonable assumption that $c_2 \leq c_1$, due to quantity discounts and/or economies of scale in transportation. This implies that our material ordering cost is no longer independent of our replenishment policy, since it depends on the mix of Q_2 and Q_1 that we order.

Since it is cheaper on a per-unit basis to order big truckloads, it will never pay to order more than one small truckload. This means that if we order a small truckload, we effectively pay a penalty of $(c_1 - c_2)Q_1$ for that delivery. Note that if $c_1 < c_2$ there is never any reason to order the large delivery size and this becomes Chen and Veinotts's setting.

It is natural to conjecture that some more elaborate version of a reorder-point policy will be optimal here. Thus there should be some value R such that the retailer should order enough Q_2's to bring her inventory just below R, and then choose whether her last order should be a Q_2 or a Q_1. This leads to the basic tradeoff in this model: If the last order is a Q_2, then we save on material cost, but at the expense of having a larger inventory, and hence larger holding costs. On the other hand, if the last order is a Q_1, then we pay more for the material, but pay lower holding costs as a result of having less inventory.

Our analysis extends that of Chen (2000).

2.1.1 The Model and Overview of Results

We assume that our retailer operates a single location using periodic review. Demand during each period is an integer number of units, and follows the stationary distribution function f. Demands in different periods are independent and identically distributed. If demand during a period is greater than inventory, then we backlog the unserved demand (at a cost).

At the beginning of the period, the retailer assesses her inventory position and places an order. For simplicity we assume zero lead time, and so the order arrives instantly. Over the period demand is realized and at the end of the period, one-period holding or backorder costs, independent of unit purchase costs, are assessed. We assume that the selling price of the good is fixed, and does not vary with demand. As a result of this assumption, price does not enter the model as all demand is met, and price does not affect demand.

The retailer can order any quantity of the form $aQ_2 + bQ_1$, where a is

a non-negative integer and where we can assess, without loss of generality (see Section 2.1), that b is zero or one.

We assume an infinite horizon and define an "optimal" policy as one that minimizes long-run average costs.

We follow Chen in deriving our results for general cost functions, assuming only some convexity properties of the one-period expected cost. As with Chen, we will be able to show that, e.g., linear holding and backorder costs lead to cost functions satisfying these properties.

Let's make an intuitive argument for what an optimal policy should look like. It certainly should have a reorder point R. As already mentioned, the retailer should order enough Q_2's to bring her inventory position into the interval $(R - Q_2, R]$, and then decide on her last order. If the all but last order brings inventory into $(R - Q_2, R - Q_1]$, then the last order must be for a large truck to bring the inventory above R. Otherwise inventory is (temporarily) in $(R - Q_1, R]$ and the last order can be either Q_1 or Q_2. If we define o_i as the amount to order from inventory position i, then a policy can be described as $[o_{R-Q_1+1}, o_{R-Q_1+2}, \ldots, o_R]$ with each $o_i = Q_1$ or Q_2.

Let $G(y)$ denote the expected one period cost given the inventory position is y after ordering. Inventory position is the sum of items on hand plus items on order. Chen proves that with one delivery size Q available and reorder point R, the stationary distribution of the Markov chain on inventory position is uniform on $(R, R + Q]$ and hence the long-run average cost is $\frac{1}{Q}\sum_{y=R+1}^{R+Q} G(y)$.

In our case the stationary distribution of inventory position under a two delivery size policy can be more complicated (see Section 2.3.4), so we must analyze the problem differently than Chen. We first describe the single delivery size case as a Markov Decision Process and show that the optimal policy has a set of recurrent or target states containing the Q smallest integer minimizers of $G(y)$.

Next we use similar techniques to show that the set of recurrent states for the two delivery size policy will include the integer minimizer of $G(y)$, plus the minimizer's Q_1 neighbours which are the next ordered integer minimizers. The set of recurrent states may also include the next Q_1 integer minimizers of $G(y)$.

We further provide some guidance on how R and the optimal policy can be computed in practice. Two approximations are provided as well as an algorithm for finding the optimal policy based on the Markov Decision process technique of Policy Iteration. It turns out, in the examples run, that the reorder points associated with the policies of always ordering just Q_2's, or always ordering just Q_1's, provide quite good starting approximations for

our optimal R.

2.2 Analytical Tools in the One Delivery Size Setting

2.2.1 Markov Decision Process

We will begin the analysis by reviewing the standard model of Veinott (1965) and the work of Chen (2000) to set the stage for our work. We will consider the setting as a periodic review inventory model at a single location. At the beginning of the period, inventory position s is observed and an order is placed, bringing the inventory position to y. To fix ideas, let us consider the simple case of one delivery size Q available for ordering. Demand in each period D, is assumed to be independent and identically distributed taking only integer values. Unmet demand in any period is backlogged. At the end of the period a holding charge is assessed on any excess inventory which is then held to the next period, and a penalty is paid for any backlogged demand. Let $G(y)$ be the expected one period holding and backorder cost given that the inventory position immediately after ordering is y. Since all delivery sizes have the same cost, and unmet demand is backlogged, item purchasing costs can be ignored. The optimal policy will minimize the long run average cost under an infinite time horizon. In Chen's work, he can prove that $G(y)$ is quasiconvex in y, but we assume that it is to make our analysis work. See Newman (1987) for a definition of a quasiconvex function.

Recognizing that inventory position is a Markov chain we follow Puterman's (1994) Markov decision process formulation by identifying the following:

Decision Epochs The beginning of the period when the ordering decision must be made

States The inventory position at the time the ordering decision is made. Define the set of inventory positions at the time the ordering decision is made by S, with lower bound $\underline{s}$ and upper bound $\overline{s}$, $S = \{\underline{s}, \underline{s}+1, \ldots, -1, 0, 1, \ldots, \overline{s}-1, \overline{s}\}$. Under reasonable demand distribution, i.e., one that is bounded below by zero, and above by some finite value, and a reasonable ordering policy, i.e., one that keeps inventory position bounded, the inventory position at the beginning of the period belongs to a finite set.

Actions Order some integer multiple a of delivery size Q. Define the set of actions for a given state s (the inventory position at the time of ordering), A_s, as $A_s = \{0, 1, 2, \ldots, \overline{a}\}$. Following the assumptions that lead to a bounded state space, we assume there is an upper limit, $\overline{a}$, to the size of the order placed.

Expected Rewards At the end of the period, after demand has been realized, holding and backorder costs are assessed. Define the cost for action a taken from inventory position s as $r(s, a) = G(s + aQ)$

Transition Probabilities Once the order has been made, the transition probabilities depend on the demand distribution. Denote the probability of demand equal to d as $P\{D = d\}$ and define the transition probability to inventory position j (before the next order is made), after taking action a from inventory position s as

$$p(j|s, a) = \begin{cases} P\{D = s + aQ - j\}, & j \leq s + aQ; \\ 0, & j > s + aQ. \end{cases}$$

2.2.2 Finding the Optimal Average Cost Policy

We seek an ordering policy π^*, consisting of actions a_s, one for each $s \in S$, which minimizes the long run average cost. For a model such as ours with stationary and bounded rewards, stationary transition probabilities and finite, discrete state space S, the resulting infinite-horizon Markov chain $\{Y_t : t = 1, 2, \ldots\}$, has average reward or gain of a policy π given as

$$g^{\pi}(y) = \lim_{N \to \infty} \frac{1}{N} E_y^{\pi} \{ \sum_{t=1}^{N} r(Y_t) \}$$

The policy's transition probability matrix $P_\pi = P_\pi(s, j) = (p(j|s, a_s))$ has a limiting matrix P_π^*. Denote the stationary distribution as q, the solution to the system of equations $q^T = q^T P_\pi$ subject to $\sum_{j \in S} q(s) = 1$. It is easy to show that $P_\pi^* = eq^T$ Puterman (1994) where e is the unit vector. Under the restrictions of our model, the average reward converges to

$$g^{\pi}(s) = P_\pi^* r(s).$$

The average cost in this form can be described as a weighted average of the expected costs of the states that are reached. The weights are the long run probability of spending a period in the state. Clearly, the average

cost is minimized under a policy that orders to the inventory positions that minimize $G(y)$, the expected one period costs.

Let y^* be an integer that minimizes $G(y)$. When $G(y)$ is quasiconvex, then the optimal policy is to order to y^* in each period. If $Q = 1$ this would be possible from all $s \in S$, and results in the class of order-up-to policies. For $Q > 1$ it is not always possible to reach y^*. From an inventory position $s < y^*$, if it is not possible to reach y^* by ordering multiples of Q, i.e., $(y^* - s) \bmod Q \neq 0$, then cost is minimized by ordering to $\arg\min\{G(s + \lfloor \frac{y^*-s}{Q} \rfloor Q), G(s + \lceil \frac{y^*-s}{Q} \rceil Q)\}$. That is, ordering either to the closest position above or below y^*, whichever has a lower expected one period cost. From inventory $s \geq y^*$, no orders should be placed. Thus, from any inventory position $s < y^*$ ordering enough deliveries to reach $y \in (y^* - Q, y^*]$, then for each inventory position $y \in [y^* - Q, y^*)$, ordering an additional delivery size if $G(y + Q) < G(y)$, is the optimal policy.

For ease of future notation, let $y_1 = y^*$ be the integer that minimizes $G(y)$. Similarly, let y_2 be the 'second' integer minimizer of $G(y)$, i.e., $y_2 \in \mathbb{Z}, G(y_1) \leq G(y_2) \leq G(y)\ \forall\ y \in \mathbb{Z}, y \neq y_1$. Since $G(y)$ is quasiconvex, $y_2 = y_1 - 1$ or $y_2 = y_1 + 1$, i.e., y_2 is a neighbour of y_1

Definition: Let $y_1, y_2, \ldots, y_Q$ be such that $G(y_1) \leq G(y_2) \leq \ldots \leq G(y_Q) \leq G(y)\ \forall y \in \mathbb{Z} \neq y_i\ i = 1, \ldots, Q$. Define $M_{G,Q} \equiv \{y_i\}, i = 1, \ldots, Q$, the ordered set of Q integer minimizers of $G(y)$. If $G(y)$ is quasiconvex then $y_i = \max_{j<i}\{y_j\} + 1$ or $y_i = \min_{j<i}\{y_j\} - 1$

If $G(y)$ is quasiconvex then the optimal policy is to order into the band of target inventory positions $(R, R + Q]$ containing $M_{G,Q}$, the Q integer minimizers of $G(y)$. The bottom of the band is $R = \min_{M_{G,Q}} -1$.

Under our optimal policy structure, the inventory position after ordering, y, forms a Markov chain $\{Y_t : t = 1, 2, \ldots\}$ over a finite set of inventory positions Y. As shown in the previous paragraph, Y contains Q inventory positions which form a contiguous band. In order to evaluate the average cost, we need the transition probability matrix P_π. As before, the reward function $r(y) = G(y)$.

The elements of the transition probability matrix p_{ij} are the probabilities of moving from $Y_t = i$ to $Y_{t+1} = j$ under the policy π, and are solely a function of the demand faced between orders. Before constructing P_π for our optimal policy structure we illustrate the possible transitions that can be made from target after-ordering inventory position $R + k$ for integer $k, 1 \leq k \leq Q$. Each line of Table 2.1 below shows the demand faced, the inventory position (IP) after the specified demand has be realized, the order placed to return to the set of target states Y, and the end state that is reached.

Starting State	Demand	IP after demand	Order	End State
$R+k$	0	$R+k$	0	$R+k$
	1	$R+k-1$	0	$R+k-1$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$k-1$	$R+1$	0	$R+1$
	k	R	Q	$R+Q$
	$k+1$	$R-1$	Q	$R+Q-1$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$Q-1$	$R+k-Q+1$	Q	$R+k+1$
	Q	$R+k-Q$	Q	$R+k$
	$Q+1$	$R+k-Q-1$	Q	$R+k-1$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$Q+k-1$	$R-Q+1$	Q	$R+1$
	$Q+k$	$R-Q$	$2Q$	$R+Q$
	$Q+k+1$	$R-Q-1$	$2Q$	$R+Q-1$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 2.1: Possible Transitions from After-Ordering Inventory Position $R+k$

From this it is easy to construct the probability transition matrix

$$P_\pi = \begin{array}{c|cccc} \text{To State} \rightarrow & R+1 & \dots & R+Q-1 & R+Q \\ \text{From State} \downarrow & & & & \\ \hline R+1 & \sum_{j=0}^{\infty} P\{D=jQ\} & \dots & \sum_{j=0}^{\infty} P\{D=jQ+2\} & \sum_{j=0}^{\infty} P\{D=jQ+1\} \\ \vdots & & & & \\ R+Q-1 & \sum_{j=0}^{\infty} P\{D=jQ+Q-2\} & \dots & \sum_{j=0}^{\infty} P\{D=jQ\} & \sum_{j=0}^{\infty} P\{D=jQ+2\} \\ R+Q & \sum_{j=0}^{\infty} P\{D=jQ+Q-1\} & \dots & \sum_{j=0}^{\infty} P\{D=jQ+1\} & \sum_{j=0}^{\infty} P\{D=jQ\} \end{array}$$

Inspection reveals that each row of the probability matrix has the same entries as the row above, with the values each shifted one column to the right. In addition to P_π being stochastic, the columns of P_π sum to one. As a result of this symmetry, the solution to $q^T = q^T P_\pi$ is

$$q(R+Q) = q(R+Q-1) = \ldots = q(R+1) = \frac{1}{Q}$$

and the long run average cost is

$$\frac{1}{Q}\sum_{i=1}^{Q} G(R+1+i)$$

as shown by Chen (2000).

Again, we note that if $G(\cdot)$ is quasiconvex, the long run average cost $g = \frac{1}{Q}\sum_{i=1}^{Q} G(R+1+i)$ is minimized by $M_{G,Q}$. The sequential order of the integer minimizers will vary with $G(\cdot)$, but the quasiconvexity of $G(\cdot)$ ensures that they form a contiguous band of inventory positions as shown in Figure 2.1.

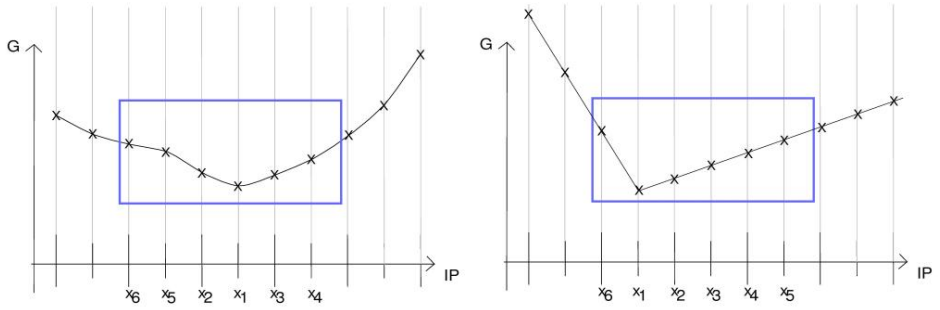


Figure 2.1: Two Examples of Set of Q Integer Minimizers of $G(\cdot)$ for $Q = 6$

These figures also illustrate that for $Q = i, Q' = j$ with $i < j$ the target inventory positions for the setting in which multiples of delivery size Q are ordered are a subset of the target inventory positions for the setting in which multiples of delivery size Q' are ordered. This is because $M_{G,Q} \subset M_{G,Q'}$.

In this section we have shown that the optimal policy under delivery size ordering of size Q is to order into the band $(R, R+Q]$, commonly known as the (R, nQ) policy. The long run average cost of this policy is $\frac{1}{Q}\sum_{i=1}^{Q} G(R+1+i)$. This result is a replication of the earlier work of Veinott and Chen which is described below. We also demonstrated that the band $(R, R+Q]$ contains the set of Q integer minimizers of $G(x)$, the one period holding and backorder costs.

2.2.3 Previous Work

Two approaches to this single delivery size problem have been presented in the literature. In 1965 Veinott showed the optimality of the (R, nQ) policy, which he called a (k, Q) policy, for the n-period and infinite period models. In his setting, at the beginning of a period inventory is reviewed and an order, consisting of integer multiples of delivery size Q, placed which will be delivered λ periods later. Over the period previous orders are delivered and new demand arrives. At the end of the period holding and penalty costs are assessed on the inventory on hand. Ordering costs of unit cost c per item, are assessed when the order is made, but are discounted to the end of the period with one period discount factor α, $0 \leq \alpha \leq 1$. Under the (k, Q) policy, if the inventory position at the beginning of the period is less than k, order the smallest integral multiple of Q that will bring the inventory position to at least k, otherwise, do not order. Given the inventory position after ordering is y the cost function is the sum of the ordering cost, and $L(y)$, the expected future holding and penalty costs. $G(y) = (1-\alpha)cy + L(y)$. $G(y)$ is assumed to be unimodal and minimized at $\overline{y}$. The policy is optimal if it minimizes the expected discounted cost. The minimizer k is chosen from all values l such that $l \leq \overline{y} \leq l + Q$.

The policy is first proved optimal for the n-period model by comparing the period-by-period inventory position under the (k, Q) policy to that under any other policy. Since unmet demand is backlogged, the difference in inventory positions after ordering under the two policies will be an integral multiple of Q in each period. In any given period i, the after-ordering inventory position under the (k, Q) policy, y_i', will satisfy $k \leq y_i' \leq k + Q$ and the after-ordering inventory position under the alternate policy y_i will satisfy either $y_i = y_i'$ or $|y_i - y_i'| \geq Q$. In both cases $G(y_i) \geq G(y_i')$. Under the (k, Q) policy, $y_i' \geq k + Q$ can only occur if $y_1' \geq k + Q$ and by period i there has not been enough demand to bring y_i' below $k + Q$. In this situation, $y_i \geq y_i'$ $i = 1 \ldots j$ where j is the first period where $y_i' < k + Q$, hence $G(y_i) \geq G(y_i')$ for these periods. Since this policy minimizes $G(y_i)$ it minimizes the expected cost in the n-period model and also the infinite period model.

Chen looks at the infinite horizon, non-discounted problem where $G(y)$ is the expected cost (holding and backorder) in a period given the after-ordering inventory position is y. Chen shows that if band $[R + 1, R + Q]$ contains the integer minimizers of $G(y + nQ)$ for any y and integer n, then this R minimizes $\sum_{x=1}^{Q} G(y + x)$. Since the demands in each period are independent, the after-ordering inventory position under the (R, nQ) policy

will follow a Markov chain. Under some mild conditions on the demand distribution, the steady-state distribution of the Markov chain is uniform. Thus the long run average single period cost is $\frac{1}{Q}\sum_{y=R+1}^{R+Q} G(y)$. This result is more general than Veinott's as Chen assumes that $\sum_{x=1}^{Q} G(y+x)$ is quasiconvex whereas Veinott assumes $G(\cdot)$ is quasiconvex. Also, Chen extends to a multi-echelon setting.

2.3 Extensions to Two Delivery Sizes

In this section, the setting is extended to one where two delivery sizes are available for ordering. We show that this complicates the policy determination over the single delivery size, illustrate a method of determining the optimal policy and finally show computationally easy upper and lower bounds on the long run average cost.

2.3.1 Formal Model

We start with the single delivery size model and extend it to multiple delivery sizes. We remind the reader of the formal model in which at the beginning of the period, inventory position s is determined and an order is placed. Demand is realized over the period with any unmet demand backlogged to a later period. Assume that demands in each period are independent and identically distributed taking only integer values, and that demand is bounded. At the end of the period a penalty is paid for any backlogged demand and any inventory held over to the next period. Define $G(y)$ as the single period expected cost given the inventory position after ordering is y.

Now we extend the model to allow for orders made up from two delivery sizes. Consider the specific case of a small delivery size Q_1 and a larger delivery size $Q_2 = 2Q_1$. Any order can consist of an integer multiple of each delivery size. There are some economies of scale so the unit cost of the small delivery size is higher than that of the larger delivery size. If c_i is the unit cost of delivery size Q_i, then $c_1 > c_2$. In this case, it is never economical to order more than one of the small delivery sizes and orders will be of the form $aQ_2 + bQ_1$ for integers $a \geq 0, b = 0, 1$. Since all demand is met in the model, all reasonable policies will order the same number of items over the long run. We can consider the material or purchase cost at the lower unit cost c_2 a sunk cost and we can account for differences in unit cost by charging a cost of $Q_1(c_1 - c_2)$ for each small delivery size ordered.

These changes result in the following changes to the definition of the Markov decision process formulation given in Section 2.2.1 above:

Actions Place an order $a_sQ_2 + b_sQ_1$

$A_s = \{(a_s, b_s)\}, a_s \in \{0, 1, 2, \ldots, \overline{a}\}, b_s \in \{0, 1\}$.

Expected Rewards At the end of the period, after demand has been realized, holding and backorder costs are assessed. $r(s, (a_s, b_s)) = G(s + a_sQ_2 + b_sQ_1) + b_sQ_1(c_1 - c_2)$

Transition Probabilities

$$p(j|s, (a_s, b_s)) = \begin{cases} Prob\{D = s + a_sQ_2 + b_sQ_1 - j\}, & j \leq s + a_sQ_2 + b_sQ_1 \\ 0, & j > s + a_sQ_2 + b_sQ_1 \end{cases}$$

As for the single order quantity described in the previous section, since the rewards (costs) are stationary and bounded and the transition probabilities stationary, a stationary policy over an infinite horizon will minimize the average cost.

2.3.2 Intuitive Starting Point

Intuitively, the two delivery size case should be structurally similar to the single delivery size case. Consider that we start with a single delivery size case, Q_2. We know that we are trying to reach the optimal reorder point in each period after ordering, but are often prevented from doing so by delivery size. Now we introduce the smaller delivery size $Q_1 = \frac{1}{2}Q_2$. With this smaller delivery size we should have a better chance at reaching the optimal reorder point, but at added cost. The smaller delivery sizes come at a greater unit cost and so faces the incremental material cost $(c_1 - c_2)Q_1$. With this in mind we hope to find a stationary policy, with a reorder point R, and a finite band of recurrent target states. Since the starting point is only ordering large delivery sizes, there is reasonable expectation that the target states will be a subset of the Q_2 target states in the single delivery size case. The decision to order the small delivery size compares the additional material cost to the savings from reaching a lower cost inventory position. The retailer orders enough large delivery sizes to reach the band $(R - Q_2, R]$, and then decides if a small or large delivery size should be ordered to bring the inventory position above R. In the band $(R - Q_2, R - Q_1]$ there is no decision to be made as only the addition of a large delivery size will cross R. This leaves the band $(R - Q_1, R]$ in which the decision between the two delivery sizes can be made, or 2^{Q_1} possible policies.

Due to the quasiconvexity of the cost function, we conjectured in earlier work, Parkinson (2005), that there would either be a single point in the band $(R - Q_1, R]$ where the optimal decision would switch from the large delivery size to the smaller, or that no small delivery sizes would be ordered. That is, moving up in inventory position from $R - Q_1 - 1$, if it became cost effective at inventory position α to order the smaller delivery size, it would also be cost effective in positions $\alpha + 1, \ldots, R$. If this were true, then there would be only $Q_1 + 1$ candidates for the optimal policy once R is established. This policy was given the name the alpha-policy.

In the next section we will show that there are 2^{Q_1} candidates for optimal policy, but that the stationary distribution may change with each candidate policy making the choice of when to make the tradeoff to the small delivery size difficult. As a result, the alpha-policy argument does not hold and all 2^{Q_1} candidate policies must be considered. In Section 2.3.5 we show a counterexample showing that the alpha-policy conjecture is false.

2.3.3 Structure of Optimal Policy

The analysis of the two delivery size case follows that of the single delivery size case, but differs significantly enough to be demonstrated in detail.

Recall that we are seeking a policy that minimizes $v(s) = \min_{(a,b)\in A_s}\{r(s,(a,b)) + \lambda \sum_{j\in S} p(j|s,(a,b))v(j)\}$. For a given policy $\pi = \{(a_s, b_s)\} \in A_s$, let $\overline{G}(s + a_sQ_2 + b_sQ_1) = G(s + a_sQ_2 + b_sQ_1) + \lambda \sum_{j\in S} p(j|s,(a,b))v(j)$ and let y^* be an integer that minimizes $\overline{G}(y)$. We want to minimize $\widetilde{G}(s + a_sQ_2 + b_sQ_1) = \overline{G}(s + a_sQ_2 + b_sQ_1) + b_sQ_1(c_1 - c_2)$

It should be clear that if $s + a_sQ_2 + b_sQ_1 = y^*$ and $b_s = 0$, then $\widetilde{G}(s + a_sQ_2 + b_sQ_1)$ is minimized at y^*. Intuitively this makes sense: if it is possible to reach the minimum cost inventory position with the cheaper, larger delivery sizes, this should be done. Also, as with the single delivery size model, no orders should be placed from inventory positions above y^*.

If b_s is zero for all s, then $\widetilde{G}(s + a_sQ_2 + b_sQ_1) = \overline{G}(s + a_sQ_2)$ and the problem is the single delivery size case. The smaller delivery size allows for the tradeoff of paying the incremental cost of ordering to arrive at an inventory position with a lower one period cost. That is, the tradeoff should be considered if $\overline{G}(s + a_sQ_2 + Q_1) < \overline{G}(s + a_sQ_2)$ or $\overline{G}(s + (a_s - 1)Q_2 + Q_1) < \overline{G}(s + a_sQ_2)$. In the first, the additional delivery size raises the inventory position by Q_1 and the second reduces the inventory position by Q_1. This leads us to conclude that if the Q_1 integer minimizers of $\overline{G}(y)$ can be reached by ordering Q_2, no savings can be realized by considering the smaller delivery size.

Recall the definition of the Q_2 integer minimizers of $\overline{G}(y)$ which would be the target states if only the large delivery size is ordered, and divide them in to two groups: the first Q_1 indices $\{y_i\}, i = 1, \ldots, Q_1$, and the second Q_1 indices $\{y_i\}, i = Q_1 + 1, \ldots, Q_2$. We have shown that $\{y_i\}, i = 1, Q_1$ will be in the target inventory positions under the optimal policy. This leaves the possibility that the tradeoff will be worthwhile from the Q_1 targets $\{y_i\}, i = Q_1 + 1, Q_2$.

Since $\overline{G}(s)$ contains the future expected costs, and (as we will demonstrate) the stationary probabilities are difficult to compute, it is not simple to evaluate this tradeoff.

We can summarize the results above as follows:

Proposition 2.1. *For the setting with two available delivery sizes Q_1 and $Q_2 = 2Q_1$ with the unit costs $c_1 > c_2$, the set of recurrent states under the optimal ordering policy will contain the set of Q_1 integer minimizers of $\overline{G}(y)$.*

Proposition 2.2. *For the setting with two available delivery sizes Q_1 and $Q_2 = 2Q_1$ with the unit costs $c_1 > c_2$, there are 2^{Q_1} candidates for an optimal policy. These policies order into some subset of the set of Q_2 integer minimizers of $\overline{G}(s)$.*

We are in the position of having the structure of the optimal policy, but it is not clear how to calculate it. For the single delivery size, calculation was simplified by realizing that the uniform stationary distribution implies a long run average cost that is fairly easy to compute. In the next section we will show that this property does not hold for the two delivery size case, making it difficult to evaluate when the small delivery size should be ordered.

2.3.4 Stationary Distribution of a Two Delivery Size Policy

Let us now look at the long run average cost of policies of the format described above, to see if this setting provides insight into determining which tradeoffs are necessary to arrive at the optimal policy.

As discussed in the previous section, for a given policy π that orders into a finite band of inventory positions, the average expected reward is

$$g^{\pi}(s) = P_{\pi}^{*} r(s).$$

In order to track the incremental cost of ordering the smaller delivery size, we need to differentiate when a state is reached by the small delivery size,

and so we will extend the state space to (s, b). The long run average cost becomes

$$g^{\pi}(s, b) = P_{\pi}^{*} r(s, b)$$

where $r(s, b) = G(s) + bQ_1(c_1 - c_2)$.

If the stationary distributions of all the candidate policies happened to be uniform, the evaluation of each of the possible tradeoffs would be the comparison of $G(s)$ to $G(s \pm Q_1) + Q_1(c_1 - c_2)$. Unfortunately this is not the case.

Only under the policy that only Q_2 is ordered is the stationary distribution guaranteed to be uniform. For the other $2^{Q_1} - 1$ policies, the stationary distribution depends on the demand distribution.

Starting from the Q_2 only policy, there are $2^{Q_1} - 1$ policy options to consider, including ordering Q_1 for any one of the $y_{Q_1+1}, \ldots, y_{Q_2}$ targets, or a subset of the targets. Each of these tradeoffs affects the stationary distribution of the recurrent target states. The following illustrates how making even one of the possible tradeoffs may result in a non-uniform stationary distribution.

Let us consider the candidate policy of ordering only the large delivery sizes. The set of target states will be the Q_2 integer minimizers of $\overline{G}(y)$. Let y_m be the largest of the target states. Table 2.2 is the transition probability matrix for this policy.

$$P = \begin{array}{c||cccccc||} \text{To State} \rightarrow & y_m-(Q_2-1) & \dots & y_m-1-Q_1 & \dots & y_m-1 & y_m \\ \text{From State} \downarrow & & & & & & \\ \hline\hline y_m-(Q_2-1) & \sum_{j=0}^{\infty} P\{D=jQ_2\} & \dots & \sum_{j=1}^{\infty} P\{D=jQ_2+2\} & \dots & \sum_{j=0}^{\infty} P\{D=jQ_2+2\} & \sum_{j=0}^{\infty} P\{D=jQ_2+1\} \\ \vdots & & & & & & \\ y_m-1-Q_1 & \sum_{j=0}^{\infty} P\{D=jQ_2+Q_1-2\} & \dots & \sum_{j=0}^{\infty} P\{D=jQ_2\} & \dots & \sum_{j=0}^{\infty} P\{D=jQ_2=Q_1\} & \sum_{j=0}^{\infty} P\{D=jQ_2+Q_1-1\} \\ \vdots & & & & & & \\ y_m-1 & \sum_{j=0}^{\infty} P\{D=jQ_2+Q_2-2\} & \dots & \sum_{j=0}^{\infty} P\{D=jQ_2+Q_1\} & \dots & \sum_{j=0}^{\infty} P\{D=jQ_2\} & \sum_{j=0}^{\infty} P\{D=jQ_2+Q_2-1\} \\ y_m & \sum_{j=0}^{\infty} P\{D=jQ_2+Q_2-1\} & \dots & \sum_{j=0}^{\infty} P\{D=jQ_2+Q_1+1\} & \dots & \sum_{j=0}^{\infty} P\{D=jQ_2+1\} & \sum_{j=0}^{\infty} P\{D=jQ_2\} \\ \hline\hline \end{array}$$

Table 2.2: Transition Probability Matrix for Only Ordering Large Delivery Sizes

Let us now assume that $y_m - 1$ is one of the $y_{Q_1+1}, \ldots, y_{Q_2}$ integer minimizers of $\overline{G}(y)$ so we want to consider ordering the small delivery size to move to the target inventory position $y_m - 1 - Q_1$ which is one of the Q_1 integer minimizers of $\overline{G}(y)$. In order to track the incremental cost of ordering the smaller delivery size, we need to differentiate when a state is reached by the small delivery size and we will use $\langle Q_1 \rangle$ to denote a target reached by the small delivery size. Thus when $y_m - 1 - Q_1$ is reached by a small delivery size it will be referred to as state $y_m - 1 - Q_1 \;\; \langle Q_1 \rangle$. With this one change in ordering policy the probability transition matrix becomes that shown in Table 2.3

To State → / From State ↓	$y_m-(Q_2-1)$	$\cdots$	y_m-1-Q_1	$y_m-1-Q_1\ \ \langle Q_1\rangle$	$\cdots$	y_m
$y_m-(Q_2-1)$	$\sum_{j=0}^{\infty} P\{D=jQ_2\}$	$\cdots$	$\sum_{j=0}^{\infty} P\{D=jQ_2+Q_2+2\}$	$\sum_{j=0}^{\infty} P\{D=jQ_2+2\}$	$\cdots$	$\sum_{j=0}^{\infty} P\{D=jQ_2+1\}$
$\vdots$						
y_m-1-Q_1	$\sum_{j=0}^{\infty} P\{D=jQ_2+Q_1-2\}$	$\cdots$	$\sum_{j=0}^{\infty} P\{D=jQ_2\}$	$\sum_{j=0}^{\infty} P\{D=jQ_2+Q_1\}$	$\cdots$	$\sum_{j=0}^{\infty} P\{D=jQ_2+Q_1-1\}$
$y_m-1-Q_1\ \ \langle Q_1\rangle$	$\sum_{j=0}^{\infty} P\{D=jQ_2+Q_1-2\}$	$\cdots$	$\sum_{j=0}^{\infty} P\{D=jQ_2\}$	$\sum_{j=0}^{\infty} P\{D=jQ_2+Q_1\}$	$\cdots$	$\sum_{j=0}^{\infty} P\{D=jQ_2+Q_1-1\}$
$\vdots$						
y_m	$\sum_{j=0}^{\infty} P\{D=jQ_2+Q_2-1\}$	$\cdots$	$\sum_{j=0}^{\infty} P\{D=jQ_2+Q_1+1\}$	$\sum_{j=0}^{\infty} P\{D=jQ_2+1\}$	$\cdots$	$\sum_{j=0}^{\infty} P\{D=jQ_2\}$

Table 2.3: Transition Probability Matrix for Ordering a Small Delivery Size at Only One Inventory Position

To summarize the changes in the transition matrix resulting from the change in the policy

- state $y_m - 1 - Q_1 \ \langle Q_1 \rangle$ has been added. The entries of this row will be exactly the same as those of the row for state $y_m - 1 - Q_1$ since the probability transitions are independent of how the originating state was reached.
- the values in the column for state $y_m - 1$ have been moved to the column $y_m - 1 - Q_1 \ \langle Q_1 \rangle$ since the latter is now reached instead of the former
- the values in the column for state $y_m - 1$ have been set to zero as there is now no way to reach this state under the revised policy.

In the new transition matrix, the probabilities in each row still sum to one – with probability one leaving any single state will end up at one of the other states, however, the probabilities in each column might not. More importantly, the probabilities in each column do not sum to the same value. The solution to $q^T P_\pi = q$ now depends on the demand distribution and we do not see any general closed-form expression for stationary distributions here. The cost for this policy is $\sum_{i=0, i\neq 1}^{Q_2-1} q(y_m - i)G(y_m - i) + q(y_m - 1 - Q_1 \ \langle Q_1 \rangle)(G(y_m - 1 - Q_1) + Q_1(c_1 - c_2))$ and the tradeoff between holding and material costs is not transparent. Clearly, the stationary distribution will only be uniform under very specific demand distributions.

Let us now consider the policy when the small delivery size is ordered whenever necessary to reach the Q_1 integer minimizers of $G(\cdot)$. In this case, while each column of the probability transition matrix does not sum to one, the sum of each pair of columns $y_i, y_i \ \langle Q_1 \rangle$ does sum to one and the result is that for each $i = 1, \ldots, Q_1 \ q(x_i) + q(x_i \ \langle Q_1 \rangle) = \frac{1}{Q_1}$. The long run average cost for this policy is $\frac{1}{Q_1} \sum_{i=1}^{Q_1} G(x_i) + \sum_{i=1}^{Q_1} q(x_i \ \langle Q_1 \rangle) Q_1(c_1 - c_2)$. This will be used to create upper and lower bounds on the optimal cost in Section 2.3.7.

Clearly the stationary distribution is not guaranteed to be uniform and computation (solving $q = q^T P$) is necessary to determine the long run average cost of each of the 2^{Q_1} candidate policies.

2.3.5 Counterexamples to Optimality of Alpha-Policy

The previous section suggests that the optimal policy will not always be an alpha-policy. Here we give two examples demonstrating that alpha-policies

are not always optimal. In the first we show that the selection of R can change the appeal of the alpha-policies, and in the second we show a case where even the selection of R does not make the alpha-policy optimal.

For the first counterexample, consider the following instance:

- $Q_1 = 3$
- $h = 25$, $b = 50$
- $c_1 - c_2 = 5$
- demand is uniformly distributed over the integers $[47, 56]$, $U(47, 56)$, i.e., $P\{D = x\} = \begin{cases} 0.1 & \text{for } x = 47, 48, \ldots, 56 \\ 0 & \text{otherwise} \end{cases}$

To compute the optimal policy we must identify the Q_2 minimizers of the cost function then evaluate the 2^{Q_1} candidate policies. In this example we use the following analysis to identify the minimizers.

The one period expected holding cost given that the inventory position after ordering is y is a combination of the expected holding and backorder costs to be paid at the end of the period after demand D is realized. Under linear costs, a unit holding cost of h is applied to each item held, and unit backorder cost of b is applied to each item in shortfall. Using $(y)^+ = \max\{y, 0\}$ and $E[y]$ as the expected value of y, the cost function is given by

$$G(y) = E[h(y - D)^+ + b(D - y)^+]$$

$$G(y) = (h + b)(yP\{D \leq y\} - \sum_{d=0}^{y} dP\{D = d\}) - b(y - E[D])$$

In this analysis we will use the following:

$$\begin{aligned}
&G(y) - G(y-1) \\
&= (h+b)(yP\{D \leq y\} - \sum_{d=0}^{y} dP\{D = d\}) + b(E[D] - y) \\
&- [(h+b)((y-1)P\{D \leq (y-1)\} - \sum_{d=0}^{y-1} dP\{D = d\}) + b(E[D] - (y-1))] \\
&= (h+b)(yP\{D \leq y\} - (y-1)P\{D \leq (y-1)\} - yP\{D = y\}) - b \\
&= (h+b)(yP\{D = y\} + P\{D \leq y-1\} - yP\{D = y\}) - b \\
&= (h+b)P\{D \leq y-1\} - b
\end{aligned}$$

If y^* is a minimizer then $G(y^*) < G(y^*+1)$ and $G(y^*) < G(y^*-1)$. Starting with the first,

$$\begin{aligned} G(y^*) - G(y^*+1) &< 0 \\ (h+b)P\{D \le y^*\} - b &> 0 \\ P\{D \le y^*\} &> \frac{b}{h+b} \end{aligned}$$

And the second

$$\begin{aligned} G(y^*) - G(y^*-1) &< 0 \\ (h+b)P\{D \le y^*-1\} - b &< 0 \\ P\{D \le y^*-1\}) &< \frac{b}{h+b} \end{aligned}$$

So if $G(y)$ minimized at y^* then

$$P\{D \le y^*-1\} < \frac{b}{h+b} < P\{D \le y^*\}$$

In our example, $P\{D \le 52\} = 0.6$, $\frac{b}{h+b} = 0.66$, and $P\{D \le 53\} = .7$ and we identify 53 as the minimizer. Evaluating $G(y)$ in the vicinity of $y = 53$ we see $\begin{array}{l} G(50) = 120 \\ G(51) = 100 \\ G(52) = 87.5 \\ G(53 = y^*) = 82.5 \\ G(54) = 85 \\ G(55) = 95 \\ G(56) = 112.5 \\ G(57) = 137.5 \end{array}$ and establish our Q_2 minimizers are $[51, 56]$, $G(53) < G(54) < G(52) < G(55) < G(51) < G(56)$, and $R = 50$.

From the demand distribution, the entries in the probability transition matrices are

$$\sum_{i=0}^{\infty} P\{D = iQ_2\} = 0.2 \qquad \sum_{i=0}^{\infty} P\{D = 1 + iQ_2\} = 0.2$$
$$\sum_{i=0}^{\infty} P\{D = 2 + iQ_2\} = 0.2 \qquad \sum_{i=0}^{\infty} P\{D = 3 + iQ_2\} = 0.1$$
$$\sum_{i=0}^{\infty} P\{D = 4 + iQ_2\} = 0.1 \qquad \sum_{i=0}^{\infty} P\{D = 5 + iQ_2\} = 0.2$$

We have $2^3 = 8$ candidates for the optimal policy as listed in Table 2.4

	Target Inventory Positions								
Policy	51	52	53	54	55	56	53 $\langle Q_1 \rangle$	54$\langle Q_1 \rangle$	52$\langle Q_1 \rangle$
1	1	1	1	1	1	1			
2	1	1	1		1	1	1		
3	1	1	1	1		1		1	
4	1	1	1	1	1				1
5	1	1	1			1	1	1	
6	1	1	1		1		1		1
7	1	1	1	1				1	1
8	1	1	1				1	1	1

Table 2.4: Candidates for Optimal Policy: Counterexample 1

Across the top are first listed the Q_2 inventory positions that minimize the cost function, followed by the Q_1 inventory positions that can be reached by ordering the smaller delivery size. In this example inventory positions 51 through 56 are the 6 minimizers, then 53, 54, 52 can also be reached by ordering small delivery sizes. In the first policy only large delivery sizes are ordered and all inventory positions 51 through 56 are reached. In the second policy, the choice has been made to order a small delivery size to reach inventory position 53 instead of reaching 54 with the large delivery size. The remaining policies enumerate the combination of choices for small delivery size.

Note that the alpha-policies allow a switch from the large delivery size to the small delivery size at or below inventory position R, in the band $(R - Q_1, R]$. In this case, only two of the $Q_1 = 3$ minimizers can be reached by ordering a small delivery size from at or below inventory position $R = 50$. The alpha-policies are the subset of the candidates shown in Table 2.5

	Target Inventory Positions								
Policy	51	52	53	54	55	56	53 $\langle Q_1 \rangle$	54$\langle Q_1 \rangle$	52$\langle Q_1 \rangle$
1	1	1	1	1	1	1			
2	1	1	1		1	1	1		
6	1	1	1		1		1		1

Table 2.5: Alpha Policy Candidates: Counterexample 1

The stationary distributions for each policy, and the average cost are given in the Table 2.6 .

	Target Inventory Positions									
Policy	51	52	53	54	55	56	53 $\langle Q_1 \rangle$	54 $\langle Q_1 \rangle$	52 $\langle Q_1 \rangle$	Avg Cost
1	0.1667	0.1667	0.1667	0.1667	0.1667	0.1667	0	0	0	93.77
2	0.1687	0.1852	0.1831	0.1646	0.1481	0	0.1502	0	0	90.89
3	0	0.1687	0.1852	0.1831	0.1646	0.1481	0	0.1502	0	92.92
4	0.1852	0.1831	0.1646	0.1481	0	0.1687	0	0	0.1502	95.08
5	0	0.1852	0.2	0.1815	0.1481	0	0.1333	0.1519	0	**90.39**
6	0.1852	0.2	0.1815	0.1481	0	0	0.1519	0	0.1333	92.06
7	0	0.1852	0.1852	0.1667	0	0.1481	0	0.1667	0.1481	94.17
8	0	0.2	0.2	0.1667	0	0	0.1333	0.1667	0.1333	91.50

Table 2.6: Stationary Distribution and Average Cost of Candidate Policies: Counterexample 1

Policy 5 has the lowest cost at 90.93, and we conclude that it is optimal to order the large delivery size at every inventory position at or below 49, and to order the small delivery size at inventory positions 50 and 51.

The example is illustrated in Figure 2.2 below which shows $G(y)$ for this counterexample 1. The $Q_1 = 3$ minimizers are shown as solid circles $\{52, 53, 54\}$ and the remaining 3 minimizers are shown as open circles $\{51, 55, 56\}$. R is indicated with the vertical line at inventory position 50. Along the bottom are indicators of the delivery size ordered from an inventory position, the open triangles and squares, and the delivery size ordered to arrive at a target state, closed triangles and squares. The open triangles show the $Q_1 = 3$ inventory positions from which Q_1 can be ordered to reach one of the Q_1 minimizers, and the open squares show the $Q_1 = 3$ inventory positions from which Q_2 can be ordered to reach one of the Q_1 minimizers. In this run, Q_2 was ordered from inventory positions at or below 49 to reach target states $\{52, \ldots, 54, 55\}$, and Q_1 is ordered from inventory positions $\{50, 51\}$ to reach target state $\{53, 54\}$. We see that at inventory position 49 the large batch is ordered and an inventory position outside of the set of Q_1 minimizers is reached.

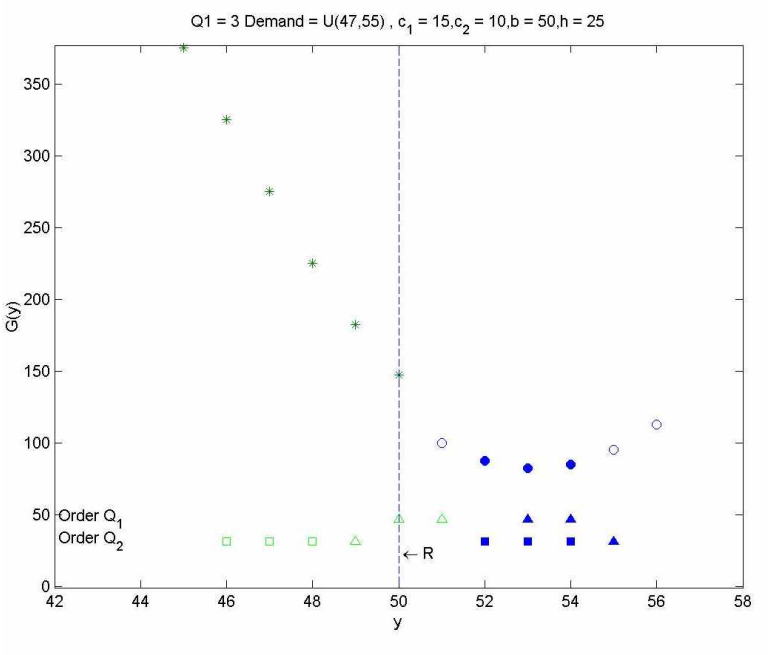


Figure 2.2: Graphical Representation of Optimal Policy. The $Q_1 = 3$ minimizers are shown as solid circles $\{52, 53, 54\}$ and the remaining 3 minimizers are shown as open circles $\{51, 55, 56\}$. The open triangles and squares show the inventory positions where the last order is placed. The triangles indicate where Q_1 can be ordered to reach a Q_1 minimizer, and the squares indicate where Q_2 can be ordered to reach a Q_1 minimizer. The solid triangles and squares show the inventory positions in the set of recurrent states for this policy.

While one may argue that instead of being a counterexample, the example above instead makes a case for somehow tweaking the definition of the alpha-policy, the next example will show that the optimal policy can have gaps that no tweaking can cover.

For the second counterexample, consider the following instance:

- $Q_1 = 3$
- $h = 25$, $b = 50$
- $c_1 - c_2 = 5$
- Demand distribution, $P\{D = x\} = 0$ except for the cases listed below.

$P\{D = 23\} = 0.09667$	$P\{D = 29\} = 0.064$	$P\{D = 53\} = 0.10267$
$P\{D = 27\} = 0.08467$	$P\{D = 48\} = \frac{1}{6}$	$P\{D = 57\} = 0.082$
$P\{D = 28\} = 0.08467$	$P\{D = 49\} = \frac{1}{6}$	$P\{D = 58\} = 0.082$

Following the steps from the previous example, we first identify the minimizer of $G(y)$, $y^* = 49$.

$$P\{D \leq 48\} < \frac{50}{75} = \frac{2}{3} < P\{D \leq 49\}$$

Next we calculate the single period expected holding and back order costs of given inventory position after ordering around the minimizer.
$G(47) = 348$
$G(48) = 328$
$G(49) = 320.5$
$G(50) = 325.5$
$G(51) = 330.5$
$G(52) = 335.5$
$G(53) = 340.5$
$G(54) = 353.2$
From the above list, the cost function is minimized at 49. We can order the minimizers as follows: $G(49) < G(50) < G(48) < G(51) < G(52) < G(53)$

From the demand distribution, the entries in the probability transition matrices are

$$\sum_{i=0}^{\infty} P\{D = iQ_2\} = 0.1667 \qquad \sum_{i=0}^{\infty} P\{D = 1 + iQ_2\} = 0.1667$$

$$\sum_{i=0}^{\infty} P\{D = 2 + iQ_2\} = 0.07 \qquad \sum_{i=0}^{\infty} P\{D = 3 + iQ_2\} = 0.1667$$

$$\sum_{i=0}^{\infty} P\{D = 4 + iQ_2\} = 0.1667 \qquad \sum_{i=0}^{\infty} P\{D = 5 + iQ_2\} = 0.2633$$

We again have $2^3 = 8$ candidates for the optimal policy as shown in Table 2.7

	Target Inventory Positions								
Policy	48	49	50	51	52	53	48 $\langle Q_1 \rangle$	49$\langle Q_1 \rangle$	50$\langle Q_1 \rangle$
1	1	1	1	1	1	1			
2	1	1	1		1	1	1		
3	1	1	1	1		1		1	
4	1	1	1	1	1				1
5	1	1	1			1	1	1	
6	1	1	1		1		1		1
7	1	1	1	1				1	1
8	1	1	1				1	1	1

Table 2.7: Candidate Policies: Counterexample 2

The stationary distributions for each policy, and the average cost are given in Table 2.8.

	Target Inventory Positions									
Policy	48	49	50	51	52	53	48 $\langle Q_1 \rangle$	49$\langle Q_1 \rangle$	50$\langle Q_1 \rangle$	Avg Cost
1	0.1667	0.1667	0.1667	0.1667	0.1667	0.1667				330.083
2	0.1655	0.1989	0.1729		0.1344	0.1604	0.1679			331.608
3	0.1604	0.1655	0.1989	0.1729		0.1344		0.1679		**329.634**
4	0.1344	0.1604	0.1655	0.1989	0.1729				0.1679	330.275
5	0.1604	0.1989	0.1989			0.1344	0.1729	0.1344		331.293
6	0.1344	0.1989	0.1729		0.1344		0.1989		0.1604	332.073
7	0.1344	0.1604	0.1989	0.1989				0.1729	0.1344	329.774
8	0.1344	0.1989	0.1989				0.1989	0.1344	0.1344	331.683

Table 2.8: Stationary Distribution and Average Costs: Counterexample 2

While the difference is small, policy 3 is the lowest cost policy. It qualifies as a non-alpha-policy since one should order the small delivery size at inventory position $52 - 6 = 46$, but at inventory position 47 the large delivery size should be ordered. This creates a gap in the set of recurrent states at inventory position 46. This example shows the intuition behind the alpha-policy that once a transition to the small delivery size is made it is not reversed, does always not hold, thus the alpha-policy conjecture is false.

The example is illustrated in Figure 2.3 below which shows $G(y)$ for counterexample 2. The $Q_1 = 3$ minimizers are shown as solid circles $\{48, 49, 50\}$ and the remaining 3 minimizers are shown as open circles $\{51, 52, 53\}$. R is indicated with the vertical line at inventory position 47. Along the bottom are indicators of the delivery size ordered from an inventory position, the open triangles and squares, and the delivery size ordered to arrive at a target state, closed triangles and squares. In this run, Q_2 was ordered from inventory positions $\{42, \ldots, 45, 47\}$ to reach target states $\{48, \ldots, 51, 53\}$, and Q_1 is ordered from inventory positions 46 to reach target state 49.

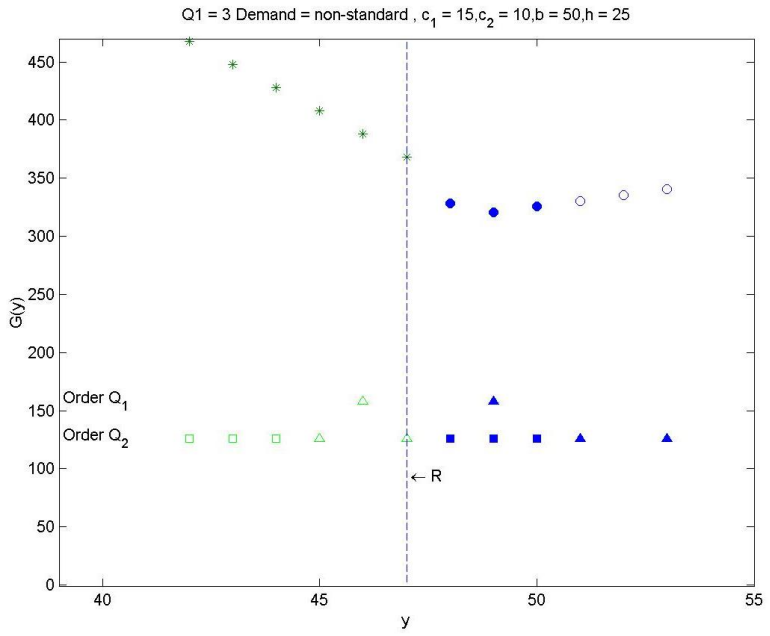


Figure 2.3: Graphical representation of optimal policy. The $Q_1 = 3$ minimizers are shown as solid circles and the remaining 3 minimizers are shown as open circles. The open triangles and squares show the inventory positions where the last order is placed. The triangles indicate where Q_1 can be ordered to reach a Q_1 minimizer, and the squares indicate where Q_2 can be ordered to reach a Q_1 minimizer. The solid triangles and squares show the inventory positions in the set of recurrent states for this policy. Note that at inventory position 46 a small delivery size is ordered, but at inventory position 47 the large delivery size is ordered.

2.3.6 An Algorithm for Computing Exact Solutions

As demonstrated, the stationary distribution of a policy depends on the demand distribution. We have shown that there are 2^{Q_1} possible candidates for optimal policy, and each candidate can be evaluated by finding the stationary distribution and evaluating the long run average cost. Let us denote the 2^{Q_1} possible candidates for optimal policy as D^*. For small values of Q_1, evaluating each candidate policy may be feasible, but as Q_1 grows, this quickly becomes infeasible. For $Q_1 = 10$ there are 1024 candidate policies.

Ideally one would look at each of the Q_1 possible inventory positions where the small delivery size can be ordered, and decide if the tradeoff is warranted. Instead, we turn to conventional methods for solving Markov decision processes. Puterman (1994) describes some algorithms used for solving Markov decision processes. The basis for these algorithms is the Bellman or optimality equations. Since we have a finite set of candidate policies, we choose the policy iteration of Howard (1960). Each step of policy iteration uses a modified form of the Bellman equations to search through candidate policies for one that has an improvement in value. We also show that the structure of policy iteration provides a separation of the problem that allows for evaluation of each of the Q_1 tradeoffs individually.

In a Markov process states are classified by the expected traffic to the states. The traffic to a state s is described with two random variables, the number of visits, v_s and the time of the first visit τ_s. If the chain starts at state s then τ_s represents the time of the first return visit. A state is recurrent if it is expected that it will be visited repeatedly. That is, $E[\tau_s] < \infty$. Otherwise, if $E[\tau_s] = \infty$ the state is transient and the state will rarely be visited. Using the number of visits, a state is recurrent if and only if $E[v_s] = \infty$, that is, there is nothing that stops visits to the state. Recurrent states are then grouped into sets based on accessibility between the states. Two states are accessible if there is a non-zero probability of transitioning between them in any number of periods. Formally, state i is accessible from state j if $p^n(i|j) > 0$ for some $n \geq 0$. For a finite state space where all states are recurrent and accessible, the Markov chain is referred to as unichain.

Since the probability matrix for our problem consists of a single recurrent class for each of the candidate policies, the model can be classified as unichain. The Unichain Policy Iteration Algorithm provided by Puterman converges to an optimal solution in a finite number of iterations for a unichain model. The algorithm is started with an arbitrary decision rule/policy d, for which the stationary distribution P_d^* and average reward

$g = P_d^* r_d$ are computed. Next

$$0 = r_d - ge + (P_{d_n} - I)h_n \text{ and } P_{d_n}^* h_n = 0$$

are solved to obtain the vector h, referred to as the bias. The bias is the expected difference between the reward in each state and the average reward under a given policy. In the policy improvement step

$$d_{n+1} \in \arg\max_{d \ inD} \{r_d + P_d h_n\}$$

is computed. In this step, the bias from the previous iteration is used to estimate the average reward of the other candidate policies, and the best is selected. The algorithm terminates when there is no improvement possible. Since the algorithm only proceeds if there is an improvement in average value, no candidate policy is evaluated more than once and the algorithm is guaranteed to converge to the optimal solution if there is a finite number of policies.

To start the Unichain Policy Iteration Algorithm a decision rule must be chosen and the set of target states determined. We have already shown that the optimal policy will only target some subset of the Q_2 integer minimizers of $G(\cdot)$. Since Q_1 of these may be reached both by ordering Q_2 or by ordering the additional Q_1, there are $Q_2 + Q_1$ target states if we also include in the state definition the number of small delivery sizes that must be ordered to reach it. These $3Q_1$ states are the

- the Q_2 integer minimizers of $G(\cdot)$, $\{x_1, \ldots, x_{Q_2}\}$, and
- the Q_1 states reached by ordering a small delivery size $\{x_1 \ \langle Q_1 \rangle, \ldots, x_{Q_1} \ \langle Q_1 \rangle\}$

By choosing the policy in which no small delivery sizes are ordered, the calculations in the first iterations can be simplified. First, $g = \sum_{i=1}^{Q_2} G(x_i)$ is easily calculated, and second, the stationary distribution P^* is the square matrix with all entries in the first Q_2 columns $\frac{1}{Q_2}$ and zero in all remaining entries.

The reward $r_{d\,i}$ is the cost for spending a period in state i, under policy d. In this model, the cost for spending the period at one of the Q_2 integer minimizers after ordering a large delivery size is policy independent and $r_{d\,x_i} = r_{x_i} = G(x_i)$. Similarly, for an inventory position reached by ordering a small delivery size, $r_{d\,x_i \ \langle Q_1 \rangle} = r_{x_i \ \langle Q_1 \rangle} = G(x_i) + Q_1(c_1 - c_2)$.

The calculation of h is further simplified by the observation that since $r_{x_j \ \langle Q_1 \rangle} = r_{x_j} + Q_1(c_1 - c_2)$, it follows that $h_{n\,x_j \ \langle Q_1 \rangle} = h_{n\,x_j} + Q_1(c_1 - c_2)$ for $j = 1, \ldots, Q_1$.

Once h is calculated, the policy improvement step is next, and here is where each tradeoff can be individually evaluated, thus reducing the need to evaluate all 2^{Q_1} policies.

Since r_{di} does not change with policy, policy improvement is reduced to inspecting $d_{n+1} \in \arg\max_{d \in D}\{P_d h_n\}$. For each state i, the improvement of policy d over d_n is

$$\sum_{j=1}^{Q_2}(p_{d\,i,x_j} - p_{d_n\,i,x_j})h_{n\,x_j} + \sum_{j=1}^{Q_1}(p_{d\,i,x_j\ \langle Q_1\rangle} - p_{d_n\,i,x_j\ \langle Q_1\rangle})h_{n\,x_j\ \langle Q_1\rangle}.$$

Nominally, this is only $3Q_1$ comparisons, instead of the $2^{Q_1}-1$ other policies in D^*.

Fortunately a few simplifications can be made to reduce the number of comparisons down to Q_1 as a result of the structure of the problem. Recall the description of the construction of the transition matrices for policies where small delivery sizes are ordered, described above, using the transition matrix for the ordering only large delivery size policy as a starting point. For replacement of x_j with state $x_i\ \langle Q_1\rangle$, a row is added with the same probabilities as x_i. Next, a new column taking the entries of the x_j column is added and the entries in column x_j set to zero. This construction highlights the following:

- for any policy $d \in D^*$, $p_{d\,i,x_j} = p_{d_n\,i,x_j}$ for $j = 1, \ldots, Q_1$. The transition probabilities from any state to one of the Q_1 minimizers does not change with policy since these columns are unchanged. Now, for each state i, the improvement of policy d over d_n is

  $$\sum_{j=Q_1+1}^{Q_2}(p_{d\,i,x_j} - p_{d_n\,i,x_j})h_{n\,x_j} + \sum_{j=1}^{Q_1}(p_{d\,i,x_j\ \langle Q_1\rangle} - p_{d_n\,i,x_j\ \langle Q_1\rangle})h_{n\,x_j\ \langle Q_1\rangle}.$$

- for each pair of states $x_j\ \langle Q_1\rangle = x_k \pm Q_1$, $j = 1, \ldots, Q_1, k = Q_1 + 1, \ldots, Q_2$, only one can be in the final set of recurrent states. When comparing policy d to d_n

  either x_k in d but the switch to $x_j\ \langle Q_1\rangle$ has been made in policy d_n. In this case $p_{d_n\,i,x_j\ \langle Q_1\rangle} = p_{d\,i,x_k}$ and $p_{d\,i,x_j\ \langle Q_1\rangle} = p_{d_n\,i,x_k} = 0$ The contribution of this pair of states to potential improvement of policy d_n is

  $$\begin{aligned}(p_{d\,i,x_k} - p_{d_n\,i,x_k})h_{n\,x_k} + (p_{d\,i,x_j\ \langle Q_1\rangle} - p_{d_n\,i,x_j\ \langle Q_1\rangle})h_{n\,x_j\ \langle Q_1\rangle} \\ = p_{d\,i,x_k}h_{n\,x_k} - p_{d_n\,i,x_j\ \langle Q_1\rangle}h_{n\,x_j\ \langle Q_1\rangle} \\ = p_{d\,i,x_k}(h_{n\,x_k} - h_{n\,x_j\ \langle Q_1\rangle})\end{aligned}$$

or $x_j \ \langle Q_1 \rangle$ in d but the switch to back to x_k has been made in policy d_n. In this case $p_{d_n\, i,x_k} = p_{d\, i,x_j\ \langle Q_1 \rangle}$ and $p_{d\, i,x_k} = p_{d_n\, i,x_j\ \langle Q_1 \rangle} = 0$. The contribution of this state pair to policy improvement becomes $p_{d\, i,x_j\ \langle Q_1 \rangle}(-h_{n\, x_k} + h_{n\, x_j\ \langle Q_1 \rangle})$

or x_j in both $p_{d_n\, i,x_j\ \langle Q_1 \rangle} = p_{d_n\, i,x_k} = 0$ and $p_{d_n\, i,x_k} = p_{d_0\, i,x_k}$. With no change in policy for these states there is potential improvement.

Combining these, policy improvement at the end of the first iteration, can be reduced to looking at the Q_1 differences $h_{n\, x_{j<Q_1>}} - h_{n\, x_k}\ j \in [Q_1 + 1, Q_2]$. If $h_{n\, x_{j<Q_1>}} - h_{n\, x_k} < 0$ then the tradeoff should be made to take the small delivery size when at state x_k as it will result in a reduction in the long run cost of the policy. In further iterations, evaluation of the potential contribution for each pair of points as above, determines if there is a policy with lower costs.

This simplification is illustrated using the following example in which $Q_1 = 2$ and a cost function $G(\cdot)$ such that $G(20) < G(21) < G(22) < G(19) < G(y)\ y \in Z, y \neq \{21, 22, 23, 24\}$. In this example there will be 4 candidate policies with the following probability transition matrices.

Policy: Order all Q_2

	19	20	21	22	20$\langle Q_1 \rangle$	21$\langle Q_1 \rangle$
19	0.1	0.2	0.3	0.4	0	0
20	0.4	0.1	0.2	0.3	0	0
21	0.3	0.4	0.1	0.2	0	0
22	0.2	0.3	0.4	0.1	0	0
20 $\langle Q_1 \rangle$	0.4	0.1	0.2	0.3	0	0
21$\langle Q_1 \rangle$	0.3	0.4	0.1	0.2	0	0

Policy: Order Q_1 at 19

	19	20	21	22	20$\langle Q_1 \rangle$	21$\langle Q_1 \rangle$
19	0	0.2	0.3	0.4	0	0.1
20	0	0.1	0.2	0.3	0	0.4
21	0	0.4	0.1	0.2	0	0.3
22	0	0.3	0.4	0.1	0	0.2
20 $\langle Q_1 \rangle$	0	0.1	0.2	0.3	0	0.4
21$\langle Q_1 \rangle$	0	0.4	0.1	0.2	0	0.3

Policy: Order Q_1 at 21

	19	20	21	22	20$\langle Q_1 \rangle$	21$\langle Q_1 \rangle$
19	0.1	0.2	0.3	0	0.4	0
20	0.4	0.1	0.2	0	0.3	0
21	0.3	0.4	0.1	0	0.2	0
22	0.2	0.3	0.4	0	0.1	0
20 $\langle Q_1 \rangle$	0.4	0.1	0.2	0	0.3	0
21$\langle Q_1 \rangle$	0.3	0.4	0.1	0	0.2	0

Policy: Order all Q_1

	19	20	21	22	20$\langle Q_1 \rangle$	21$\langle Q_1 \rangle$
19	0	0.2	0.3	0	0.4	0.1
20	0	0.1	0.2	0	0.3	0.4
21	0	0.4	0.1	0	0.2	0.3
22	0	0.3	0.4	0	0.1	0.2
20 $\langle Q_1 \rangle$	0	0.1	0.2	0	0.3	0.4
21$\langle Q_1 \rangle$	0	0.4	0.1	0	0.2	0.3

If we start the algorithm with the policy of ordering all Q_2 as d_0 we can compare the improvement of the other policies as $(P_d - P_{d_0})h_0$. Let us use h_{0_s} to indicate the state s component of the bias vector from the

initialization step. The improvement for ordering the small delivery size at inventory position 19 is

$$\begin{bmatrix} -.1 \\ -.4 \\ -.3 \\ -.2 \\ -.4 \\ -.3 \end{bmatrix} (h_{0_{19}} - h_{0_{21\langle Q_1 \rangle}}),$$

for ordering the small delivery size at inventory position 22

$$\begin{bmatrix} -.4 \\ -.3 \\ -.2 \\ -.1 \\ -.3 \\ -.2 \end{bmatrix} (h_{0_{22}} - h_{0_{20\langle Q_1 \rangle}}),$$

and for always ordering small delivery sizes

$$\begin{bmatrix} -.1 \\ -.4 \\ -.3 \\ -.2 \\ -.4 \\ -.3 \end{bmatrix} (h_{0_{19}} - h_{0_{21\langle Q_1 \rangle}}) + \begin{bmatrix} -.4 \\ -.3 \\ -.2 \\ -.1 \\ -.3 \\ -.2 \end{bmatrix} (h_{0_{22}} - h_{0_{20\langle Q_1 \rangle}}).$$

Clearly, if $h_{0_{19}} - h_{0_{21\langle Q_1 \rangle}} < 0$ then the tradeoff to the smaller delivery size at inventory position 19 should be made for the next iteration. Similarly, if $h_{0_{22}} - h_{0_{20\langle Q_1 \rangle}} < 0$ then the tradeoff to ordering the smaller delivery size to reach inventory position 22 should be made for the next iteration. By evaluating the difference in bias between pairs of states for each opportunity to order the smaller delivery size, the policy that satisfies $d_{n+1} \in \arg\max_{d\ in D}\{r_d + P_d h_n\}$ is quickly found. The improvement for each tradeoff consideration is additive and the policy improvement step is reduced to the consideration of Q_1 tradeoffs.

While the evaluation of each step is fast, the consideration of Q_1 tradeoffs, it may be necessary for the algorithm to evaluate all of the 2^{Q_1} possible candidates for optimal policy before completion. In practice, however, this algorithm converges quickly. In the eighteen examples worked to date with

linear cost functions, 4 converge in one step and the others converge in at most 4 steps. These examples are discussed in Section 2.4

In this section we have shown that one pass of the Unichain Policy Iteration Algorithm can easily provide a method of evaluating the tradeoff of ordering a small delivery size at each of the inventory positions where it may be considered. Unfortunately we can't prove that it will converge in one pass of the algorithm. If convergence is not reached in a small number of iterations, then it may be sufficient to bound the results. The following section provides easily calculated upper and lower bounds on the value of the optimal policy.

2.3.7 Upper and Lower Bounds

Faced with the potential evaluation of many iterations of the Unichain Policy Iteration Algorithm or finding the stationary distribution of 2^{Q_1} policies to find the optimal policy, we have developed upper and lower bounds on the cost of optimal policies. This section describes the bounds and demonstrates that the gap between the bounds is small for reasonable examples.

Proposition 2.3. *For the setting with two available delivery sizes Q_1 and $Q_2 = 2Q_1$ with the unit costs $c_1 > c_2$, $\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i)$ is a lower bound on the long run average cost of the optimal policy.*

Proof. When only Q_1 is available to be ordered, the optimal policy is to order to the set of Q_1 integer minimizers of $G(\cdot)$ resulting in a long run average cost of $\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i)+$ ordering. Adding the possibility of ordering Q_2 allows for the tradeoff of higher holding cost, for a savings in ordering cost. While there is not a policy with cost $\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i)$, it is a lower bound on the policy. □

Proposition 2.4. *For the setting with two available delivery sizes Q_1 and $Q_2 = 2Q_1$ with the unit costs $c_1 > c_2$, $\min\{\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i) + Q_1(c_1 - c_2), \frac{1}{Q_2}\sum_{i=1}^{Q_2} G(x_i)\}$ is an upper bound on the long run average cost of the optimal policy.*

Proof. The long run average cost of the optimal policy when only the large delivery size is ordered is $\frac{1}{Q_2}\sum_{i=1}^{Q_2} G(x_i)$, thus this is a feasible solution and is an upper bound. When Q_1 is ordered so that the set of Q_1 integer minimizers of $G(\cdot)$ are always reached, the long run average cost is $\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i) + \sum_{i=1}^{Q_1} q(x_i \quad \langle Q_1\rangle)Q_1(c_1 - c_2)$ for $0 \leq q(x_i \quad \langle Q_1\rangle) \leq \frac{1}{Q_1}$ as shown at the end of Section 2.3.4. Since $\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i) + \sum_{i=1}^{Q_1} q(x_i \ \langle Q_1\rangle)Q_1(c_1 - c_2) \leq$

$\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i) + Q_1(c_1 - c_2)$ this is an upper bound to this policy. Thus, the minimum of the two upper bounds is an upper bound on the optimal policy. □

The gap between the upper and lower bounds is

$$\min\{\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i)+Q_1(c_1-c_2)-\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i), \frac{1}{Q_2}\sum_{i=1}^{Q_2} G(x_i)-\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i)\}$$

$$= \min\{Q_1(c_1 - c_2), \frac{1}{Q_2}(\sum_{i=Q_1+1}^{Q_2} G(x_i) - \sum_{i=1}^{Q_1} G(x_i))\}.$$

The gap is the minimum of the incremental material cost of the small delivery size, and the average incremental one period costs under the large delivery size. This is the fundamental tradeoff and the extremes are the basis for the bounds. Unfortunately, this gap could be arbitrarily bad if $\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i)$ is close to zero. However, it seems unlikely that $\frac{1}{Q_2}\sum_{i=Q_1+1}^{Q_2} G(x_i)$ would be excessively large if $\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i)$ is close to zero. The following numerical study provides some evidence that the gap is can be small in practice, although some instances had a gap 32% of the optimal cost.

2.4 Numerical Study

An numerical study was conducted to demonstrate the magnitude of the gap between the upper and lower bounds and compare the solution techniques.

A linear holding and backorder model was considered where the one period expected cost after ordering is given by $G(y) = hE[y-d]^+ + bE[d-y]^+$. The demand was assumed to follow either a normal or uniform distribution.

The parameters for the runs were chosen to create instances that varied not only with delivery size, but also with relative costs and demand. Small delivery size ranged from 3 to a maximum of 20 and the incremental cost of ordering the small delivery size, $c_2 - c_1$ was either 1,3 or 5. Two levels of holding and back order costs were used; low, approximately 5, and high, 20 to 30. The spread of the demand distribution relative to the delivery size was also varied.

Ten instances were selected and run with uniformly distributed demand for Runs 1 through 10, then with normally distributed demand for Runs 11 through 20.

In each run, we computed the upper and lower bounds and solved the instance using the following three algorithms

1. Computation of stationary distributions and total cost of each of the 2^{Q_1} candidates for optimal policy

2. Unichain Policy Iteration Algorithm (UPIA) as described in Section 2.3.6

3. Computation of stationary distributions and total cost of the up to $Q_1 + 1$ alpha-policy candidates

Table 2.9 shows the results for runs with uniform demand distribution and Table 2.10 the results for the runs with normal demand distribution. Matlab was used for the computations.

We use the following notation in the presentation of the results.

g^*	is the average cost of the optimal policy (obtained through enumeration)
UB Q_1	$\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i) + Q_1(c_1 - c_2)$
UB Q_2	$\frac{1}{Q_2}\sum_{i=1}^{Q_2} G(x_i)$
UB	min{ UB Q_1, UB Q_2}
LB	$\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i)$
g^{P*}	the average cost of the policy found through the UPIA
g^{P_1}	is the average cost of the policy from one iteration of the UPIA Note: The 0^{th} iteration will have value UB Q_2
# It	is the number of iterations required for the UPIA to converge Note: Starting with ordering all Q_2 is considered the 0^{th} iteration. One additional iteration is required to confirm convergence.
g^{α}	is the average cost of the policy found from evaluating only at most $Q_1 + 1$ alpha-policies

In the cases run, the gap between the upper and lower bound ranges between 0.12% and 32% of the optimal solution and did not confirm the proposition that the gap between bounds is small.

The success of policy iteration is also demonstrated with very few iterations required for the algorithm to converge. Of the instances run, some converged after one iteration and the maximum number of iterations required was 4. In the instance with Q_1 of 20, UPIA converged in 3 iterations and we were unable to complete the $2^{20} = 1\ 048\ 476$ evaluations necessary for full enumeration.

Run	1	2	3	4	5	6	7	8	9
Q_1	3	5	7	3	15	5	10	15	20
c_1	20	20	15	20	20	20	20	20	20
c_2	19	19	12	15	19	19	19	19	19
h	30	30	22	20	30	5	5	5	5
b	30	30	33	20	30	3	3	3	3
Demand Uniform	(65,135)	(90,110)	(77,123)	(85,115)	(90,110)	(40,60)	(40,60)	(40,60)	(40,60)
g^*	533.60	162.38	319.22	156.88	192.38	21.21	24.81	30.37	
UB Q_1	535.68	165.00	333.45	170.27	198.81	25.05	31.21	38.22	45.96
UB Q_2	533.73	169.29	319.59	156.88	261.67	21.21	25.96	32.99	41.17
UB	533.73	165.00	319.59	156.88	198.81	21.21	25.96	32.99	41.17
LB	532.68	160.00	312.45	155.27	183.81	20.05	21.21	23.22	25.96
UB-LB	1.06	5.00	7.14	1.61	15.00	1.17	4.74	9.77	15.21
$\frac{UB-LB}{g^*}$	0.2%	3.08%	2.24%	1.03%	7.8%	5.5%	19.1%	32.2%	
$\frac{UB-LB}{LB}$	0.2%	3.13%	2.28%	1.04%	8.16%	5.82%	22.4%	42.1%	58.6%
g^{P*}	533.60	162.38	319.22	156.88	192.38	21.21	24.81	30.37	35.39
g^{P_1}	533.60	162.38	319.22	156.88	193.29	21.21	24.81	30.47	35.70
$g^{P_1} - g^*$	0	0	0	0	0.914	0	0.000	0.093	
# It	2	2	2	1	4	1	2	3	3
g^α	533.60	167.38	319.50	156.88	242.12	21.21	25.31	31.62	
$g^\alpha - g^*$	0	5.0	0.275	0	49.74	0	0.505	1.24	
Solve Time (s)									
UPIA	0	0	0.016	0	0.031	0	0.015	0.031	0.063
all 2^{Q_1}	0.016	0.015	0.141	0	531.20	0.016	2.032	529.05	
α	0	0.016	0.016	0	0.047	0	0.015	0.062	

Table 2.9: Runs with Uniform Distribution

The UPIA proved most effective for the large values of Q_1 where complete enumeration was time consuming or not possible (Runs 9 and 18). These runs also demonstrate that the minimum value alpha-policy is not always optimal, with the largest gap of 25% from optimality. Note that these runs provide further counterexamples to the alpha-policy conjecture beyond those in Section 2.3.5

Table 2.11 illustrates the first step of the UPIA for three instances that converged in one iteration. Recall that the 0^{th} iteration assumes only the

Run	10	11	12	13	14	15	16	17	18
Q_1	3	5	7	3	15	5	10	15	20
c_1	20	20	15	20	20	20	20	20	20
c_2	19	19	12	15	19	19	19	19	19
h	30	30	22	20	30	5	5	5	5
b	30	30	33	20	30	3	3	3	3
Demand Normal	(100,35)	(100,10)	(100,23)	(100,15)	(100,10)	(50,10)	(50,10)	(50,10)	(50,10)
g^*	739.95	243.95	477.50	240.51	268.69	31.57	34.83	38.95	
UB Q_1	742.09	246.95	492.78	254.44	276.29	35.64	41.57	48.07	55.13
UB Q_2	739.96	249.02	477.57	240.51	320.29	31.57	35.13	40.54	47.24
UB	739.96	246.95	477.57	240.51	276.29	31.57	35.13	40.54	47.24
LB	739.09	241.95	471.78	239.44	261.29	30.64	31.57	33.07	35.13
UB-LB	0.87	5.00	5.79	1.07	15.00	0.94	3.56	7.47	12.12
$\frac{UB-LB}{g^*}$	0.12%	2.05%	1.21%	0.44%	5.58%	2.96%	10.2%	19.2%	
$\frac{UB-LB}{LB}$	0.12%	2.07%	1.23%	0.45%	5.74%	3.05%	11.3%	22.6%	34.5%
g^{P*}	739.95	243.95	477.50	240.51	268.69	31.57	34.83	38.95	43.56
g^{P_1}	739.95	243.95	477.50	240.51	268.69	31.57	34.83	38.96	43.63
$g^{P_1} - g^*$	0	0	0	0	0	0	0	0.013	
# It	2	2	2	1	2	1	2	3	3
g^α	739.95	246.49	477.57	240.51	296.88	31.57	35.06	39.67	
$g^\alpha - g^*$	0	2.538	0.0747	0	28.19	0	0.222	0.71	
Solve Time (s)									
UPIA	0.015	0	0.016	0	0.032	0	0.015	0.031	0.062
all 2^{Q_1}	0	0.032	0.141	0.016	546.3	0.016	2.313	530.4	
α	0	0	0.015	0	0.109	0	0.015	0.078	

Table 2.10: Runs with Normal Distribution

large delivery size is ordered and has uniform stationary distribution of $\frac{1}{Q_2}$ for each state. The table shows the h values for each of the integer minimizers where the decision to order the small delivery size can be made, and compares it to the h value at its tradeoff pair. Note y_j denotes the states that are candidates for tradeoffs, i.e., $\{y_j\} = \{x_i\}\ i \in [Q_1 + 1, Q_2]$ Where the difference in h value, i.e., $\Delta = h_{x_i\ \langle Q_1 \rangle} - h_{y1}$ is negative, a switch to ordering the small delivery size is made. The lower section of the table enumerates the 2^{Q_1} candidate policies and value, identifying them by the

which tradeoffs, if any, to the small delivery size have been made. In the three instances shown, the first iteration policies match the optimal policies found through enumeration.

Run	1		4				
Normal	(100,35)		Normal	(100,15)		Uniform	(91, 105)
$Q_1 = 3$, $c_1 = 20$, $c_2 = 19$			$Q_1 = 3$, $c_1 = 20$, $c_2 = 15$			$Q_1 = 3$, $c_1 = 20$, $c_2 = 19$	
$h = 30$, $b = 30$			$h = 20$, $b = 20$			$h = 30$, $b = 30$	
h_{y_1}	0.552		h_{y_1}	1.75		h_{y_1}	11.02	
$h_{x_i\ \langle Q_1 \rangle}$	2.151		$h_{x_i\ \langle Q_1 \rangle}$	13.59		$h_{x_i\ \langle Q_1 \rangle}$	-2.68	
Δ	1.6		Δ	11.84		Δ	**-13.70**	order Q_1
h_{y_2}	0.163		h_{y_2}	-0.34		h_{y_2}	2.13	
$h_{x_i\ \langle Q_1 \rangle}$	2.344		$h_{x_i\ \langle Q_1 \rangle}$	14.63		$h_{x_i\ \langle Q_1 \rangle}$	-1.8	
Δ	2.18		Δ	14.97		Δ	**-3.93**	order Q_1
h_{y_3}	1.913		h_{y_3}	1.913		h_{y_3}	0.74	
$h_{x_i\ \langle Q_1 \rangle}$	1.877		$h_{x_i\ \langle Q_1 \rangle}$	13.58		$h_{x_i\ \langle Q_1 \rangle}$	-0.41	
Δ	**-0.036**	order Q_1	Δ	11.67		Δ	**-1.15**	order Q_1
Tradeoff to Q_1	Policy Value		Tradeoff to Q_1	Policy Value		Tradeoff to Q_1	Policy Value	
none	739.96		none	**240.51**		none	118.33	
y_1	740.22		y_1	242.48		y_1	115.88	
y_2	740.32		y_2	243.01		y_2	117.63	
y_3	**739.95**		y_3	242.47		y_3	118.13	
y_1&y_2	740.58		y_1&y_2	244.98		y_1&y_2	115.28	
y_1&y_3	740.22		y_1&y_3	244.45		y_1&y_3	115.53	
y_2&y_3	740.31		y_2&y_3	244.97		y_2&y_3	117.4	
y_1&y_2&y_3	740.58		y_1&y_2&y_3	246.94		y_1&y_2&y_3	**114.93**	

Table 2.11: Details of Policy Iteration

For the example in the left of Table 2.11, Run 1, policy iteration chooses the policy that only makes the tradeoff for the small delivery size at y_3, for the example in the middle, Run 4, policy iteration chooses never to make the tradeoff. In the third example, policy iteration indicates the best policy makes the tradeoff whenever possible — again supported by calculating the value of all policies. These policies are all demonstrated to be optimal in the lower rows.

2.5 Further Extensions

One natural extension is to the case where Q_2 is not restricted to $2Q_1$. In this case, if only Q_2 were ordered, the set of Q_2 integer minimizers of $G(\cdot)$ would be the targets and form the band $(R, R+Q_2]$. The long run average cost for this policy is easily computed as the stationary distribution is $\frac{1}{Q_2}$. With no differences in unit costs between delivery sizes, the lowest cost policy will order into the set of Q_1 integer minimizers of $G(\cdot)$. If Q_2 is an integer multiple of Q_1 then $\frac{1}{Q_1}\sum_{i=1}^{Q_1} G(x_i)$ still is a lower bound. If Q_2 is not an integer multiple of Q_1 then it is not guaranteed that the stationary distribution of the policy ordering to the set of Q_1 integer minimizers of $G(\cdot)$ will be constant over all states.

For any two delivery size sets $Q_1 < Q_2$ and unit costs c_i such that it is reasonable to order both, there are $2^{Q_2-Q_1}$ candidates for optimal policy. For each of the $Q_2 - Q_1$ integer minimizers of $G(\cdot)$ $\{x_{Q_1+1}, \ldots, x_{Q_2}\}$ a decision is made to order Q_1 to reach an inventory position with lower value for $G(\cdot)$ at the cost of an incremental delivery size.

Starting with a policy of ordering only the large delivery size, and a set of target states including

- the Q_2 integer minimizers of $G(\cdot)$
- the $Q_2 - Q_1$ states reached by ordering a multiple of Q_1. These states will be a subset of the above group, but are differentiated with the notations $< Q_1 >$ denoting that a small delivery size must be ordered to reach them.

The same policy iteration technique can be used to identify the optimal policy.

Table 2.12 shows the results of some runs with two delivery sizes, $Q_2 \neq 2Q_1$.

These runs illustrate the extension to arbitrary delivery sizes and in these examples the Unichain Policy Iteration converged to the optimal solution within two iterations.

2.6 Conclusions

In this work we studied the problem of optimal inventory policy when orders are made up of a combination of two delivery sizes. We found that the optimal solution was highly dependent on the demand distribution, and so optimal solutions do not in general have a simple form. Our analysis

shows how to identify the candidate policies and provides easily calculated upper and lower bounds on the optimal solution. Finally, we demonstrate that Unichain Policy Iteration Algorithm can leverage the structure of the problem to reduce the policy improvement step to a simple subtraction for each of the states where the option to order a smaller delivery size exists. In the runs in the numerical study, Unichain Policy Iteration Algorithm converged with at most 4 iterations.

2.7 Acknowledgments

Research supported in part by research grants from the Natural Sciences and Engineering Research Council of Canada (NSERC). We thank Jeannette Song, Mahesh Nagarajan, Tim Huh and Eric Cope for their valuable help.

Q_1	2	2	2	2	7
Q_2	6	5	7	8	21
c_1	20	20	15	20	20
c_2	19	19	12	15	19
h	30	30	22	20	30
b	30	30	33	20	30
Demand Uniform	(65,135)	(90,110)	(77,123)	(85,115)	(90,110)
g^*	533.50	158.48	312.45	158.39	167.52
UB Q_1	534.61	159.86	316.22	165.16	169.86
UB Q_2	533.73	160.00	312.45	158.39	209.52
UB	533.73	159.86	312.45	158.39	169.86
LB	532.61	157.86	310.22	155.16	162.86
UB-LB	1.13	2.00	2.22	3.23	7.00
$\frac{UB-LB}{g^*}$	0.21%	1.26%	0.71%	2.04%	4.18%
$\frac{UB-LB}{LB}$	0.21%	1.27%	0.72%	2.08%	4.3%
g^{P*}	533.50	158.48	312.45	158.39	167.52
g^{P_1}	533.50	158.48	312.45	158.39	167.52
$g^{P_1} - g^*$	0	0	0	0	0
# It	2	2	1	1	2
Solve Time (s)					
UPIA	0	0.016	0.016	0.016	0.015
all $2^{Q_2-Q_1}$	0	0	0.016	0.031	125.56

g^*	average cost of the optimal policy obtained through enumeration
UB Q_2	$\frac{1}{Q_2}\sum_{i=1}^{Q_2} G(x_i)$
g^{P*}	average cost of the policy found through the Unichain Policy Algorithm
g^{P_1}	average cost of the policy found from one iteration of the Unichain Policy Iteration Algorithm. Note: The 0^{th} iteration will have value UB Q_2
# It	number of iterations of the Unichain Polity Iteration Algorithm to convergence. Note: Ordering all Q_2 is considered the 0^{th} iteration. One additional iteration is required to confirm convergence.
UPIA	Unichain Policy Iteration Algorithm

Table 2.12: Runs with Two delivery sizes, $Q_2 \neq 2Q_1$

Chapter 3

Sequence Optimization in Block Cave Mining

3.1 Introduction

When ore bodies such as diamonds, gold and copper lie far below the surface, traditional open pit mining techniques are neither cost effective nor environmentally friendly. Instead, deep ore bodies can be excavated using a technique called block cave mining, Gertsch (1998). Vertical shafts are sunk below the ore body and a network of horizontal tunnels are dug as a platform for the mining operation. Blasting is done above the tunnels causing the rock to fall in a controlled manner into the tunnels. The cave formed by the initial blasting weakens the rock above, and the rock continues to fall. Figure 3.1 shows a cross section of a block cave mine.

It is well known that the traditional open pit mine design problem can be represented in a network flow framework, Ahuja et al (1993), and solved using the Lerchs Grossman algorithm, Lerchs & Grossman (1965) or min-cut/ max flow algorithms, i.e., Yegulalp and Arias (1992), Giannini et al (1991). The ore body is divided into blocks via a 3-dimensional grid, each with an estimated value based on the ore composition. The objective is to determine the blocks to extract to maximize the total value, subject to physical constraints. These constraints include having to remove the blocks above a desired block and maintaining a pit shape that contains stable slopes.

At a first approximation, a block cave mine can be thought of as an inverted open pit mine. The ore body is divided into blocks, with each vertical column of blocks accessible from a draw point along the horizontal tunnels. Ore at the bottom of a column must be extracted before ore at the top can be reached, and in order to keep the caves stable, neighbouring draw points must extract at a similar rate.

Block cave mining differs significantly from open-pit mining in two ways. First, the amount extracted from each draw point in each time period must

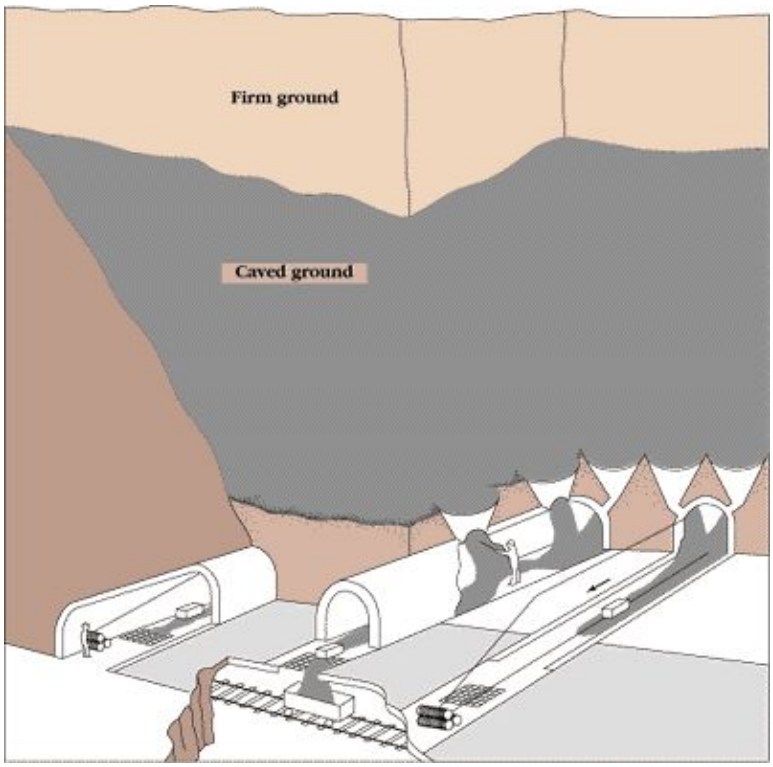


Figure 3.1: A cross section of the underground workings at the Dutoitspan mine, Kimberley, South Africa. Adapted from De Beers. (American Museum of Natural History (2008))

exceed a minimum amount in order to keep the rock flowing, and must not exceed a maximum amount for safety considerations. Secondly, having the draw point locations tied to both the tunnel structure and the blasting procedures used imposes additional constraints on the order in which the draw points are brought into production.

Typically a block cave mine will consist of 200 to 2000 draw points and will be mined for 10 to 20 years. Gemcom (2006), our industrial partner in this work, provides specialised mining productivity solutions that help their clients plan, monitor and improve their mining operations. They have worked with every major mining company in the world and have extensive experience planning block cave mines. To support this work they have developed PC-BC, a program for designing and evaluating block cave mining

operations, Diering (2000). Given the layout of draw points, the order in which they should be opened, and detailed information on the ore body such as grade and grade distribution which would be important to define the NPV values mentioned later, the program produces a production plan that tries to maximize the net present value of the material extracted from the mine. The model takes into consideration both vertical mixing, within the rubble in a draw point, and horizontal mixing, across draw points.

This work is designed to assist in providing a required input to the PC-BC program; the order in which the draw points are opened. This step, known as sequence optimization, is currently a manual process that relies on the expertise of the mine planner. The sequence lists which draw points will be opened in each year of operation and is used by PC-BC to plan each year of mine operation. The current process first manually develops a sequence, then evaluates the sequence using PC-BC, then tries to improve the sequence, etc. As it is practical to evaluate only a small subset of sequences, planners do not know how much value they are missing out on, and how much effort should be invested in the iterative process. The objective of this work is to find an optimal opening sequence in an automated manner.

3.1.1 Current Practice

Currently sequence optimization is done by an experienced person on a trial and error basis. It is assumed once a draw point is opened ore will be withdrawn at a constant rate and a lifetime can be calculated for each draw point. A value can also be assigned to each draw point which takes into consideration the estimated lifetime and the ore profile. The feasible opening sequences are created by incorporating draw point value, knowledge of physical constraints and experience, then evaluated. A series of opening sequences are evaluated and the results used to try to create an opening sequence that improves on the highest achieved net present value. Unless every possible feasible opening sequence is evaluated, this trial and error approach can not be guaranteed to deliver the plan with the highest net present value. It is also not possible to estimate or put bounds on the highest net present value for a given ore body using the trial and error approach.

3.1.2 Tunnel Details

The procedures used in the mine to develop and open the draw points constrains the order in which they can be opened. We now describe the process

to aid in the development of the algebraic constraints.

A block cave mine consists of the tunnels and access routes in the mining footprint, the "draw bells" where the blasting is done and the "draw points" where the ore is removed by the trucks. As the broken rock is removed from draw points, the rock above the draw bell falls into the open space creating more broken rock to be removed. All the blasting, rock removal and other mining activities take place at the mining footprint below the ore body that is being mined. See Figure 3.2

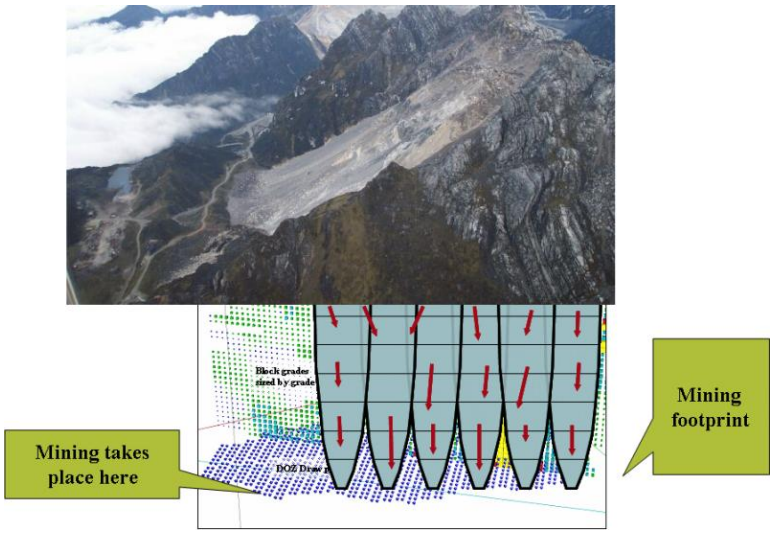


Figure 3.2: Cross Section Picture of Block Cave Mine

The draw points are arranged on a network of parallel, horizontal tunnels at the bottom of the mining area. This is known as the Extraction Level.

Between the tunnels a series of draw bells are constructed, each accessed by two draw points. Figure 3.3 shows draw point and draw bell orientation to tunnels. The rock will fall from above into the draw bells, then be removed through the draw points and transported along the tunnels to the conveyor system which takes the rock out of the mine.

Approximately 18 meters (m) above the Extraction Level is the Undercut Level where the initial blasting is done. The steps of Undercut Level development are summarized in Figure 3.4. Horizontal Undercut Tunnels are dug above and parallel to the tunnels in the Extraction Level. Depending on the mine layout, the Undercut Tunnels are either directly aligned with the Extraction Tunnels, or at some consistent offset. When a draw point is

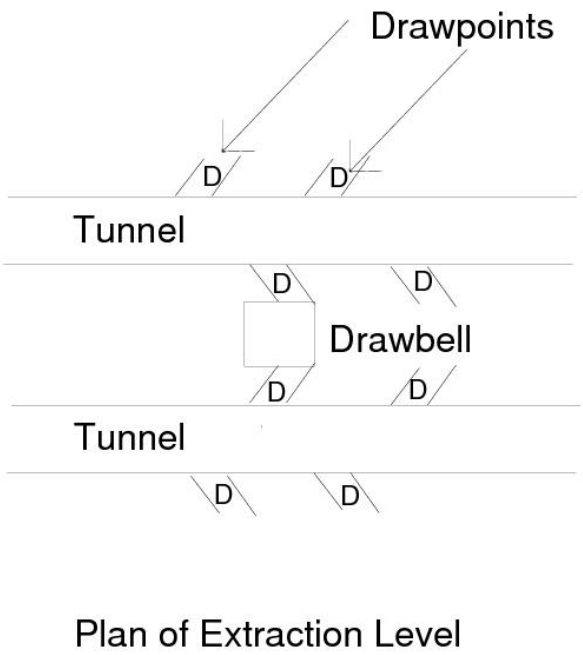


Figure 3.3: Plan of Extraction Level of Block Cave Mining Operation

scheduled to start producing, blasting is done to free the rock above so it can fall into the draw bell. Blast holes are drilled in the walls and ceiling of the Undercut Tunnel above the draw bell, making a ring or fan shape around the tunnel. The holes are packed with explosives and the blasting causes the rock to fall down to the draw bell below. See Figure 3.5.

For the purposes of this work, the constraints of the blasting process can be summarized as follows: blasting starts at one point in the undercut tunnel and works along the tunnel, in both directions, from that point. Since the draw point opening is synonymous with blasting, the draw point opening sequence must also progress along the tunnel from the initial start point. At any time, the opened draw points must form a contiguous group with no gaps between opened draw points. This is illustrated in Figure 3.6 which depicts three single tunnels and the draw points along them. For each example, the number in each square refers to the period in which the draw

Figure 3.4: Steps in Undercut Development. Adapted from Resolution Copper Mining (2009)

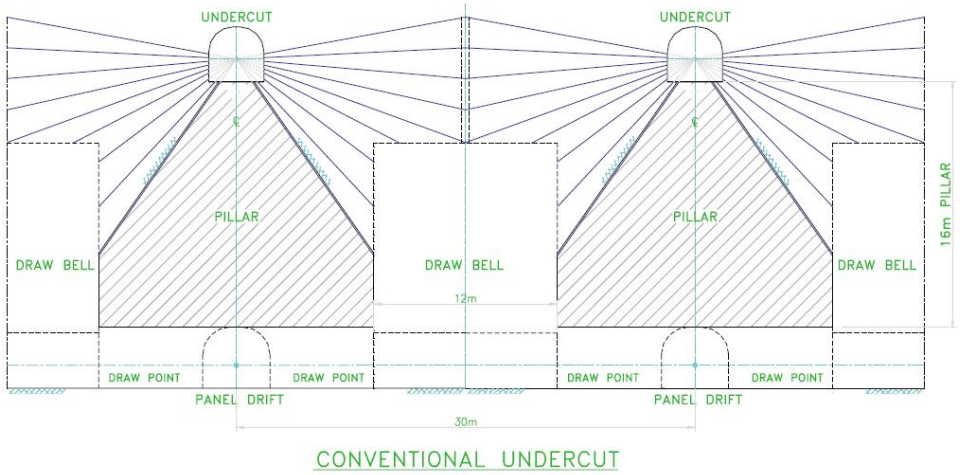


Figure 3.5: Example of Conventional Undercut Layout Showing Blasting Pattern. Barber (2000)

point is opened. In the first example, a draw point in the middle region is the initial start point and the draw points to the right are opened until the end of the tunnel is reached, then those to the left of the start point are opened. In the second example, the opened draw points progress in both directions. Both these examples form contiguous groups and meet the blasting constraints. The third example, at the bottom of the figure, breaks contiguity in the third period by opening a lone draw point at the left hand

end of the tunnel.

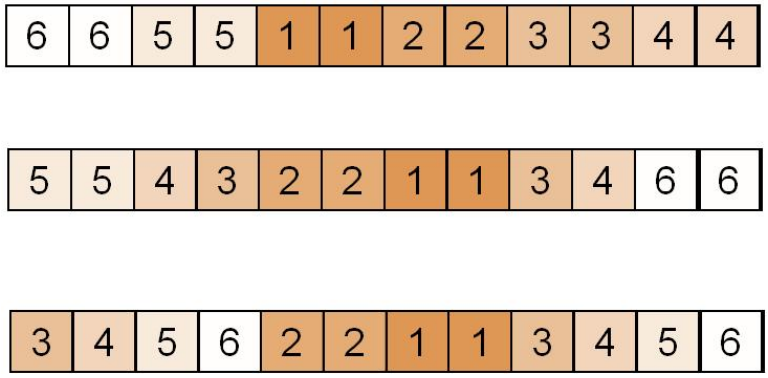


Figure 3.6: Three Examples of Single Tunnel Opening Sequences. Each square represents a draw point on a tunnel and the number in it the period in which the draw point is opened. The top two maintain a contiguous group of opened draw points, the bottom one does not.

In addition to the constraints on the order in which draw points within a tunnel can be opened, there are also some across-tunnel constraints. In general, the mining operation is thought of as a cave and the structural conditions of the cave roof govern the falling rock and the propagation of the cave upwards. The objective is a contiguous cave that grows out from a starting point. The shape should be convex so as not to leave isolated pockets of rock. A diamond shape cave is commonly created as this has resulted in good rock behavior. As a result, the opening sequence of draw points must consider what is happening in adjacent tunnels. In the PC-BC program this is covered under neighbour constraints which prevent one draw point from producing much more than its within-tunnel and adjacent-tunnel neighbours.

To create/preserve the cave structure, some coordination is needed between the tunnels. When work begins in an adjacent tunnel, it will be next to rings that have already been developed, see Figure 3.7. The difference in the number of blasting rings between adjacent tunnels is controlled so that the lead or lag does not exceed an upper limit. Generally it is acceptable to have a lead of one draw bell (approximately 7 rings), but a lead of two draw bells (15 rings) may not be acceptable.

While the cave must grow in some sort of contiguous fashion, it is not always a constraint that there be only one cave. In rare cases such as very

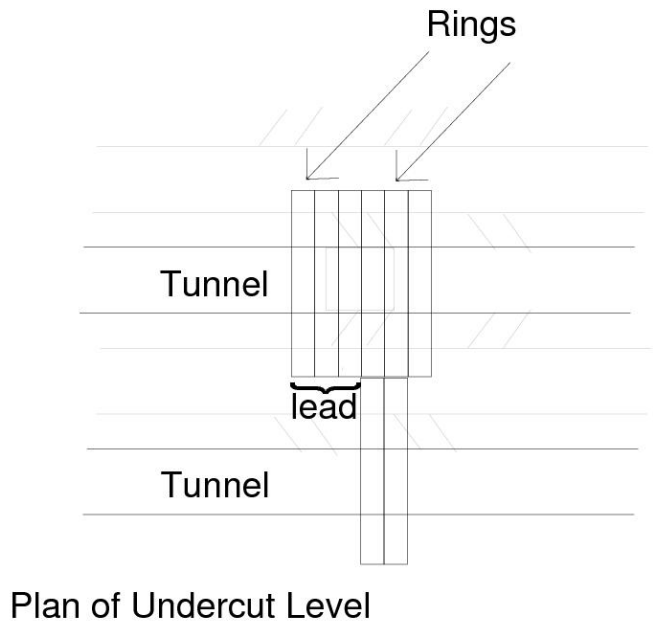


Figure 3.7: Plan of Undercut Level of Block Cave Mining Operation

large mines, mining can begin in two separate tunnels, and the caves merge together over time. In practice, a cave is initiated by opening multiple adjacent draw points in order to create the conditions that cause the rock to drop and it is not common to have multiple caves forming.

3.1.3 Cave Shape

In current practice, the draw point opening pattern grows in a diamond shape, aligned with the tunnels as illustrated in Figure 3.8. A chevron pattern is also common.

This diamond/chevron pattern conserves the within-tunnel constraint of one starting point, and also links the adjacent tunnels together.

Mining engineers prefer the diamond shape opening patterns over other shapes for operational reasons. One reason is the known as the "hydraulic radius". The hydraulic radius is the ratio of the opening area to perimeter,

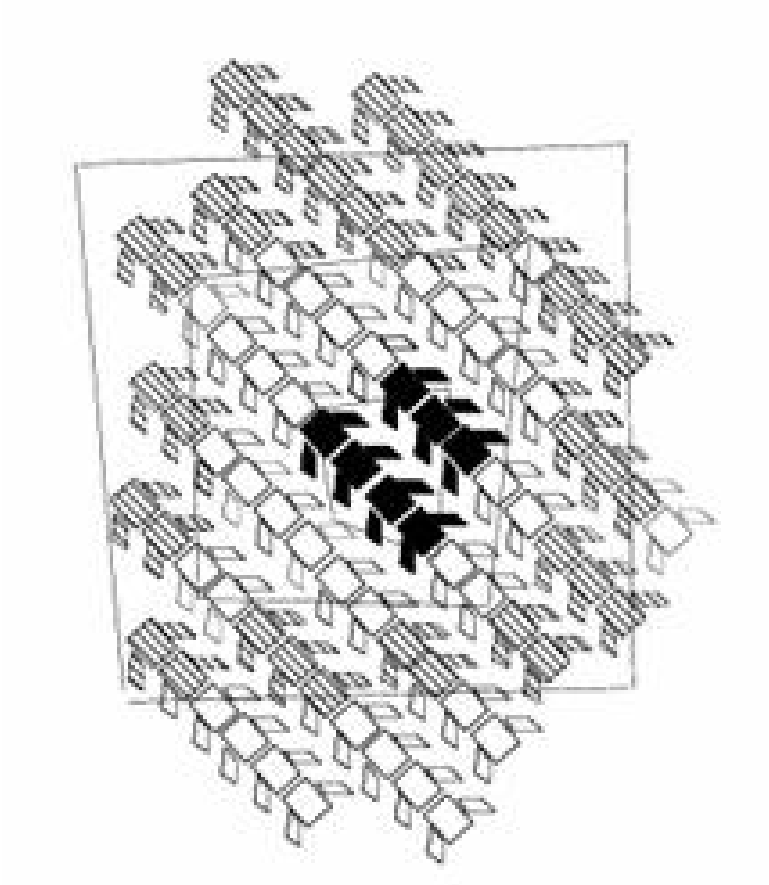


Figure 3.8: Typical Diamond Pattern of Opened Draw Points. The solid dark draw points are opened in the first period, the white ones in the second period and the hatched ones in the third period. Diamond shapes have been drawn around the draw points opened in each period.

and gives an indication of how well the rock will break apart and fall into the draw points. Diamond shaped openings result a hydraulic radius in the desired range for good caving. The diamond shapes also ensure that mining progresses into new rock in a straight-edged wedge shape. While the interior angle of this wedge can vary, in practice values below 40 degrees have not been used.

3.1.4 Other Considerations

There are other considerations that can arise. At this time they are outside of the scope of this thesis, but will be listed here.

Cave Limitations

Some mines may have additional constraints imposed that are unique to that mine. For example, there may be surface activity above part of the ore body that must be completed before the block cave mining can begin. As a result, the options for the starting point would be restricted to a smaller subset of draw points not below the surface activity.

There may also be additional data available from sampling that indicates additional information or constraints. This could also take the form of setting the starting draw points for the mine.

Grade Limitations

Some ore bodies are contaminated with other minerals. While this is usually not a concern with mine planning, it may be a concern in the final product leaving the downstream processing plant. A planning schedule may have to take into account that on average the final product may have limits on the contaminant content.

3.1.5 Assumptions

A starting assumption in this work is that we can estimate a lifetime for each draw point by assuming a constant draw rate. The draw rate is the amount of broken rock that is removed in each period from each draw point. In reality this is not the case as the PC-BC program helps to optimize the draw rates of each draw point. In addition, production rate increases over time since the rock removed generally becomes finer over time. That is, when a draw point is first opened, the rock removed is large, coarse pieces, but over time the rock pieces become smaller.

3.1.6 Previous work

Models have been created of many stages of the mining process. Newman (2000) reviews mine planning applications of operations research and notes the literature dates back to the 1960s. The work had been conducted on surface and underground mine planning problems that include equipment selection and long- and short-term production scheduling. Work in block cave

mining includes Diering (2006), who models the rock flows from individual draw point, taking into account lateral mixing. Rubio (2002) maximizes NPV in a block cave mine using draw rate as one constraint. The cave area is set by specifying the area of a contiguous cave. Weintraub (2008) creates a global model of all the copper mines owned by CODELCO, the Chilean state copper mine. All these models assume that the opening sequence is known and use it as a constraint that is determined prior to running their models.

3.2 Model Framework

The problem that we consider is the following. Given the total net present value (NPV) and expected lifetime of each draw point, determine which draw points should open in each period to maximize the total NPV of the mine subject to the physical and operational constraints. Due to discounting, the total NPV of the mine is a function of the opening sequence. Intuitively, opening draw points with higher total NPV first increases the total NPV of the mine over starting with draw points of lower value. In the absence of any constraints, opening all positively valued draw points in the first period, and never opening the negatively valued ones, maximises the total NPV of the mine.

The total NPV of draw point i is the net present value if it was started at time 1 and ore was extracted at a constant rate until it was exhausted, and is represented by w_i. "Total" emphasises that this value represents the cumulative value over all the periods from start to completion. The time to exhaustion/completion is its duration, p_i. A discount rate, λ, is used to calculate the net present value. Typically, a discount rate of between 7 to 12 % per year has been used, but values as high as 15% have been used, Diering (2006).

To clarify, the period when the first material is removed from a draw point is the period in which a draw point is *started*. In this model, the draw point will be *active* for the next p_i periods including the period in which it was started. A draw point will be called *open* or *opened* in the period it is started and each following period until the end of the time horizon. In this model we are interested in how many draw points are active in each period, and the contiguity of opened draw points.

3.2.1 Data

Data :

$\mathbf{p}_i$ duration of draw point i

$\mathbf{w}_i$ total NPV value of draw point i if started immediately

λ discount rate, $0 \leq \lambda \leq 1$.

$\mathbf{m}$ capacity constraint on number of draw points active at any time

3.2.2 Decision Variables

Decision Variables $S_{i,t}$ Binary decision variable modelling the start date for draw point i, over the lifetime of the mine, $1 \leq t \leq T$. First

period is 1. $S_{i,t} = \begin{cases} 1 & \text{if draw point } i \text{ is started in period } t, \\ 0 & \text{otherwise} \end{cases}$

Typically the plans are done with a period of length one year or six months. We will assume one period is a year long, but essentially the same formulation works for any period length.

3.2.3 Objective

The objective function is the sum of draw point values, discounted to the start date.

$$\max Value = \sum_i w_i \sum_t S_{i,t} e^{-\lambda(t-1)}$$

for some discount rate λ.

3.2.4 Constraints

Constraints :

Start Once To control the openings, each draw point can be started at most once

$$\sum_t S_{i,t} \leq 1 \quad \forall \text{ draw points } i \tag{3.1}$$

Note: If $\sum_{u=1}^{t} S_{i,u} = 1$ then the draw point was started at some period $\{1, \ldots, t\}$ and is considered open. Otherwise, this sum equals zero.

Global Capacity We assume a global capacity constraint, whereby at most m draw points can be active at any time. In general, this capacity is linked to the capacity of the downstream ore processing plant.

$$\sum_i \sum_{u=t-p_i+1}^{t} S_{i,u} \leq m \quad \forall \text{ periods } t \tag{3.2}$$

Due to the high cost of developing each draw point, approximately $300,000, production will typically ramp up over the first few years.

This can be handled by imposing a maximum number, m_t to open each year t, i.e., $m_1 = 50, m_2 = 120, m_t = 200$ for $t \geq 3$.

Ideally, the ramping rate would be dictated by the model and be a function of the opening cost and the initial available money. A budget constraint for each period would ensure that the draw point opening expenditure did not exceed the available money plus any revenue generated. In reality, the opening cost is incurred at time zero, and the revenues start to flow back at time 1. In the current structure of the model, the revenue has been lumped to a NPV to the end of the first period after opening – so all the revenues would occur at time 1 instead of distributed over the assumed life of the draw point. To approximate, we could take the total NPV and distribute it evenly over the life of the draw point. If more detailed information of the revenue variation over time was available it could also be included.

Tunnel Development

The mechanics of Block Cave Mining impose two main tunnel development constraints, one within-tunnel and the other across-tunnels.

Additional Constraints :

Within-Tunnel Contiguity Within a tunnel, the draw point opening starts at one point in a tunnel, then progresses successively in both directions along the tunnel. There are no gaps left between open draw points.

Across-Tunnel Contiguity Between tunnels, the opened draw points must form some sort of contiguous cave. Thus when the first draw point in a tunnel is opened, it must be next to opened draw points in an adjacent tunnel.

These constraints are difficult to write in algebraic form. Several options will be presented in this work.

3.2.5 Unconstrained Sequence Optimization as a Draw Point Scheduling Model

The unconstrained version has the *Start Once* (3.1) and *Global Capacity* (3.2) constraints, but none of the cave defining constraints. A parallel can be drawn between the unconstrained version of this problem and machine scheduling problems. The unconstrained version of sequence optimization can be viewed as scheduling the n draw points as "jobs" on m "parallel, identical machines".

This problem is well studied in a non-discounted setting as $p\|\sum w_j C_j$. See Pinedo (1995) for notation explanation and introduction to scheduling problems. The objective is to schedule jobs, all available from the first time period, to minimize the weighted sum of completion times. For the case of a single machine, the optimal solution can be found using Smith's Rule or the Weighted Shortest Processing Time (WSPT) rule: sequence the jobs in nondecreasing order of their ratio of processing time to weight, $\frac{p_i}{w_i}$. When there is more than one machine, the problem is NP Hard. Results from this problem show that once jobs have been allocated to a machine, they are scheduled using WSPT on that machine.

We look at the discounted problem under two conditions, one of uniform draw point duration, and the other when there is no assumption of uniformity.

Uniform Draw Point Duration $p_i = p$

In the case of uniform draw point duration of p periods, the problem reduces to deciding which m draw points will open in periods 1, $p+1$, $2p+1$ ... In the absence of discounting, it does not matter in what order they are opened. With discounting, it is intuitive that to maximize NPV, the most valuable draw points should be opened first. We now show that a greedy algorithm on w_i, the total value of the draw point, is optimal.

Proposition 3.1. *When scheduling jobs of uniform duration p on m machines, greedily choosing jobs in non-increasing order of w_i, the total value of the job, maximizes NPV, with discount factor λ.*

Proof. It is clear that for non-negative w_i it is optimal to start each job as soon as possible, so it would never be optimal not to start m jobs in the first period, or to ever have less than m jobs active in any given period (except at the end if there are no remaining jobs).

We order the jobs by value w_i, in non-increasing order, $w_1 \geq w_2 \geq \ldots \geq w_J$. In period 1 start the m most valuable jobs, i.e., $w_1, \ldots, w_m$ for a value of $\sum_{k=1}^{m} w_k$, then, in period p+1, the next m most valuable jobs are started for a value of $\sum_{k=m+1}^{2m} w_k e^{-\lambda p}$, etc.

Consider any alternate sequence of the same job values $(s_1, s_2, \ldots, s_n)$, and the resulting schedule defined similarly, by starting the first m jobs, i.e., those with values $s_1, \ldots, s_m$ for a value of $\sum_{k=1}^{m} s_k$, then, in period p+1, the next m jobs are started for a value of $\sum_{k=m+1}^{2m} s_k e^{-\lambda p}$, etc.

We will switch the first job in the alternate sequence, that does not match with greedy sequence. Denote this as job r. We have that $s_i =$

w_i for $i = 1 \ldots r-1$ and $s_r < w_r$. Let $s_\alpha = w_r$ and we shall switch the positions of s_r and s_α. If s_r and s_α are originally in the same period, then there is no change to the value of the sequence by making the switch. If s_r was originally in period $q+1$ and s_α was originally in a later period $q+\beta+1, \beta > 0$, then the value of the sequence will change by $-s_r e^{-\lambda q} + s_r e^{-\lambda(q+\beta)} - w_r e^{-\lambda(q+\beta)} + w_r e^{-\lambda q}$. Reorganizing, the change in objective value is $(w_r - s_r)e^{-\lambda q} + (s_r - w_r)e^{-\lambda(q+\beta)} = (w_r - s_r)(e^{-\lambda q} - e^{-\lambda(q+\beta)})$. For $\lambda \geq 0$ we conclude that this switch is an improvement to the value of the sequence and the thus the algorithm gives us an optimal solution. □

Non-uniform Draw Point Duration

This problem is more difficult.

In this form the problem is similar to the NP-hard, Garey (1979), scheduling problem $p\|\sum w_j C_j$, which minimizes the sum of delay penalties calculated as the time to completion (C_j) times a per period penalty (w_j). To see the similarity, assign each draw point i value $wdelay_i$, the total value of the draw point at the end of its lifetime p_i, given that the draw point was started at period 1. Thus our problem becomes $\max_i \sum wdelay_i e^{-\lambda Ci}$. If instead we had a linear discount $\frac{T-C_i}{T}$ the problem would become $\max_i \sum wdelay_i \frac{T-C_i}{T} = \sum wdelay_i - \sum wdelay_i \frac{C_i}{T}$, and the similarity to $\min_j \sum w_j C_j$ becomes clear. However, the discounting in our problem is not linear so a different solution is needed.

Let's start with a single machine and a modified greedy algorithm.

Proposition 3.2. *When scheduling jobs of non-uniform duration p_i, each with total value w_i, on a single machine, greedily choosing jobs in non-increasing order of $\frac{w_i}{1-e^{\lambda p_i}}$, maximizes NPV, with discount factor λ.*

Proof. Order the jobs by ratio of $\frac{w_i}{1-e^{\lambda p_i}}$ in non-increasing order and run the jobs in that order. Again it is clear that the machine should always be busy for non-negative w_i. The total objective value will be $w_1 + w_2 e^{-\lambda p_1} + w_3 e^{-\lambda(p_1+p_2)} + \ldots + w_J e^{-\lambda(p_1+\ldots+p_{J-1})}$. Consider any alternate sequence of the same ratios $(\frac{s_1}{1-e^{\lambda d_1}}, \frac{s_2}{1-e^{\lambda d_2}}, \ldots, \frac{s_n}{1-e^{\lambda d_n}})$. Running the jobs in this order will result in an objective value of $sw_1 + s_2 e^{-\lambda d_1} + s_3 e^{-\lambda(d_1+d_2)} + \ldots + s_J e^{-\lambda(d_1+\ldots+d_{J-1})}$. Let r be the position of the first ratio in the alternate sequence that differs from the greedy sequence. We have that $\frac{s_i}{1-e^{\lambda d_i}} = \frac{w_i}{1-e^{\lambda p_i}}$ for $i = 1 \ldots r-1$ and $\frac{s_r}{1-e^{\lambda d_r}} < \frac{w_i}{1-e^{\lambda p_i}}$. Let $\frac{s_\alpha}{1-e^{\lambda d_\alpha}} = \frac{w_r}{1-e^{\lambda p_r}}$. Unless both jobs have the same duration, or are immediately adjacent to each other, i.e., $\alpha = r+1$, more than these two terms in the objective function will change. Let us

first consider the case of adjacent jobs, $\alpha = r+1$. The objective function will change by $-s_r e^{-\lambda(p_1+\ldots+p_{r-1})} - w_r e^{-\lambda(p_1+\ldots+p_{r-1}+p_\alpha)} + s_r e^{-\lambda(p_1+\ldots+p_{r-1}+p_r)} + w_r e^{-\lambda(p_1+\ldots+p_{r-1})} = e^{-\lambda(p_1+\ldots+p_{r-1})}(w_r + s_r e^{-\lambda p_r} - s_r - w_r e^{-\lambda p_\alpha})$
$= e^{-\lambda(p_1+\ldots+p_{r-1})}(w_r(1 - e^{-\lambda p_\alpha}) - s_r(1 - e^{-\lambda p_r})) > 0$. For $\lambda > 0$ this is an improvement to the value of the sequence. Next look at the case where $\alpha = r + 2$. This switch can be broken down into two adjacent job switches. First change the order of job $\alpha = r + 2$ and job $r + 1$. We have shown above that this switch improves the objective value. Now switch jobs α and r. Since $\frac{w_r}{1-e^{\lambda p_r}} \geq \frac{w_{r+1}}{1-e^{\lambda p_{r+1}}} \geq \frac{w_\alpha}{1-e^{\lambda p_\alpha}}$ we can again use the argument above to show that the switch improves the value of the alternate sequence. Since there is improvement from both of the two adjacent switches we conclude there is improvement from the switching of jobs r and α. A similar argument can be made for any switch of jobs and we conclude that the ordering $\frac{w_i}{1-e^{\lambda p_i}}$ maximises the objective value. □

Next, move on to scheduling on multiple machines. We have shown above that once the jobs are allocated to each machine, there is a greedy algorithm that is optimal. It is the allocation of jobs to machines that is the problem. It can be shown by the following counter-example that using WSPT ordering to assign to multiple machines is not optimal when minimizing weighted completion times.

Example: Schedule the 8 jobs shown below on three parallel machines; the jobs have been sorted in nonincreasing $\frac{w_i}{p_i}$

Job Number	w_i	p_i	$\frac{w_i}{p_i}$
1	10	2	5
2	4	1	4
3	9	3	3
4	3	1	3
5	8	4	2
6	9	5	1.8
7	5	3	1.667
8	7	5	1.4

A scheduling of the jobs as ordered above is as follows. In period 1 start job 1 on machine 1, job 2 on machine 2 and job 3 on machine 3. In period 2 job 2 will be completed with a value of 2, and job 4 is started on machine 2. In period 3 jobs 1, value 20, and 4, value 6, are completed and job 5 is started on machine 1 and job 6 started on machine 2. In period 4 job 3 is completed, value 27, and job 7 started on machine 3. In period 7 job 5 is completed, value 48, and job 8 started on machine 1. Finally, job 7 is

completed in period 6 for a value of 30, job 6 is completed in period 7 for a value of 63 and job 8 is completed in period 11 for a value of 77. The total weighted completion time is 275. The job schedule and weighted completion times are illustrated below.

	Period	1	2	3	4	5	6	7	8	9	10	11	
Machine 1	job	1	1	5	5	5	5	8	8	8	8	8	total
	w_iC_i		20				48					77	145
Machine 2		2	4	6	6	6	6	6					
	w_iC_i	4	6					63					73
Machine 3		3	3	3	7	7	7						
	w_iC_i			27			30						57
													275

Now let's switch the scheduled order of jobs 6 and 7. There is no change in the schedule until period 3. Instead of starting job 6 on machine 2, job 7 is started in its place. In period 4 job 3 is completed and job 6 is started on machine 3. In period 5 job 7 is completed with a new, lower, value of 25, and job 8 is started on machine 2. Next job 6 completes in period 8 for an increased value of 72 and job 8 is completed in period 10 for a reduced value of 70. Overall the weighted completion time is reduced to 272, an improvement over the original schedule as illustrated below.

	Period	1	2	3	4	5	6	7	8	9	10	11	
Machine 1	job	1	1	5	5	5	5						total
	w_iC_i		20				48						68
Machine 2		2	4	7	7	7	8	8	8	8	8		
	w_iC_i	4	6			25					70		105
Machine 3		3	3	3	6	6	6	6	6				
	w_iC_i			27					72				99
													272

This example demonstrates that using WSPT is not optimal for scheduling jobs, with non-uniform duration, on multiple machines when minimizing weighted completion time.

The same example can be used to show that a greedy algorithm on the ratio $\frac{w_i}{1-e^{-\lambda p_i}}$ does not maximize total NPV when $\lambda = 0.1$. Switching the order of jobs 6 and 7 increases the NPV from 47.180 to 47.272 as shown below.

Job Number	w_i	p_i	$\frac{w_i}{p_i}$	$\frac{w_i}{1-e^{-0.1*p_i}}$	Value with greedy	Value with jobs 6 and 7 switched
1	10	2	5	55.167	10	10
2	4	1	4	42.033	4	4
3	9	3	3	34.725	9	9
4	3	1	3	31.525	2.175	2.175
5	8	4	2	24.266	6.550	6.550
6	9	5	1.8	22.873	7.369	6.667
7	5	3	1.667	19.291	3.704	4.094
8	7	5	1.4	17.790	3.842	4.246
				total	47.180	47.272

The forgoing shows that even simple variations of this problem are challenging. This motivates investigating the use of general integer programming methods.

3.3 Single Tunnel

We begin by capturing the physical neighbour constraints on opening the draw points begun within a single tunnel, as the *Within-Tunnel Contiguity* constraint is a more manageable starting point than the combined *Within-Tunnel Contiguity and Across-Tunnel Contiguity* constraints of the two-dimensional problem.

We assume that a tunnel consist of n draw points and they are numbered in increasing order along the tunnel, so the neighbours of draw point i are $i-1$ and $i+1$. The draw points at the end of the tunnel, 1 and n, each only have a single neighbour, 2 and $n-1$ respectively.

The problem with formulating the single tunnel problem is to write algebraic constraints that result in a single, contiguous cave in each period. The cave includes previously opened draw points as well as any that may open in that period.

A simple approach is to say that a given draw point can be started in a period if either of its immediate neighbours was started in the current or any previous period, i.e., if considered open in the current period. For the draw points at the ends of the tunnel there is only one immediate neighbour.

This constraint works if the global capacity constraint is $m = 1$, but if multiple draw points can be started in a period then they can be next to each other and act as open neighbours for each other, since the sum is up to and including the current period. This can be avoided by only counting draw points opened in previous periods, i.e., sum to previous period. Unfortunately, this would only allow at most two draw points to start each period, one on each end as no others would have an neighbour open in an earlier period.

Two formulations were developed for the single tunnel problem. The alternating constraints formulation, see Section 3.3.1, sums up each odd subset of draw points in a tunnel. If there is only one cave along the tunnel, each sum will be at most one. If there are multiple tunnels, at least one sum will exceed one. There are a large number of these constraints, but we show a subset that is sufficient if binary variables are used.

In the extended formulation, see Section 3.3.2, an additional binary decision variable is added that compares each draw point to its neighbour and indicates if that draw point is the first in a new cave. By allowing only one transition to a new cave, a single, contiguous cave is achieved.

3.3.1 Alternating Constraints Formulation

Single Tunnel: Alternating Constraints

This section describes a model that uses a set of constraints we have called alternating constraints to achieve the *Within-Tunnel Contiguity* constraint.

Mathieu Van Vyve (2005) suggested the following set of constraints to describe the convex hull of the *Within-Tunnel Contiguity* constraint.

Decision Variables $x_{i,t} \geq 0$ $x_{i,t} = \sum_{u=1}^{t} S_{i,u}$ and indicates if the draw point is open in period t. $x_{i,t} = \begin{cases} 1 & \text{if draw point } i \text{ is open} \\ 0 & \text{otherwise} \end{cases}$

and the following alternating inequalities:

for any increasing sequence $i(1) < i(2) < \ldots < i(k) \in \{1, \ldots, n\}$ with k positive, odd integer

$$\sum_{j=1}^{k} (-1)^{j+1} x_{i(j),t} \leq 1.$$

The $k = 1$ constraints, $x_{i,t} \leq 1$ are satisfied by the definition that $0 \leq S_{i,t} \leq 1$ and the *Start Once* (3.1) constraint.

Van Vyve further suggested that if $S_{i,t}$ is forced to binary, that $k = 3$ is sufficient, see Theorem 3.1 below, but the higher values of k are necessary for the linear relaxation. This is further discussed below.

For a single tunnel of n draw points there are $\binom{n}{3} = \frac{(n)(n-1)(n-2)}{6}$, $k = 3$ constraints plus $\binom{n}{j}$, $k = j$ constraints for $j = 5, 7, \ldots, n$ if integrality of $S_{i,t}$ is not specified, in each period.

Theorem 3.1. *The $k = 3$ alternating constraints ensure the Within-Tunnel Contiguity constraint if $S_{i,t}$ are binary. Let $C = \{x \in \{0,1\}^n \mid x \text{ is contiguous }\}$ and let $A = \{x \in \{0,1\}^n \mid x_{i,t} - x_{j,t} + x_{l,t} \leq 1 \; \forall \; i < j < l \;\; \forall \; t\}$. We claim that $A = C$.*

Proof. It suffices to show $C \subseteq A$ and $A \subseteq C$.

Start with the first, that every consecutive vector satisfies all the ($k = 3$) alternating constraints.

Given $x \in C$, for any pair $x_{i,t}, x_{j,t}$ $i < j$ there are 4 cases

$(x_{i,t} = 0, x_{j,t} = 0)$ Then for any $x_{l,t}, l > j$ the alternating sum will be ≤ 1

$(x_{i,t} = 0, x_{j,t} = 1)$ Then for any $x_{l,t}, l > j$ the alternating sum will be ≤ 0

$(x_{i,t} = 1, x_{j,t} = 1)$ Then for any $x_{l,t}, l > j$ the alternating sum will be ≤ 1

$(x_{i,t} = 1, x_{j,t} = 0)$ Then since $x \in C$, for any $l > j$ $x_{l,t} = 0$ and the alternating sum will be zero.

Thus $x \in A$ and $C \subseteq A$.

For the reverse, take $x \in A$. By definition, x satisfies all of the $k = 3$ alternating constraints. For $x \in \{0,1\}^n$ the only way to fail the alternating constraint is to have the triple $\{x_{i,t}, x_{j,t}, x_{l,t}\} = \{1,0,1\}, \forall\ i < j < l$. Thus $x \in A$ means that for every pair $\{x_{i,t}, x_{l,t}\}, i < l$, if $x_{i,t} = 1$ and $x_{l,t} = 1$ then each $x_{j,t} = 1\ \forall\ i < j < l$. This means that there are no "gaps" in the 1's, and thus $x \in C$ and $A \subseteq C$.

□

In fact, some of the $k = 3$ alternating constraints are redundant and not all $\binom{n}{3}$ are required.

Lemma 3.2. *One sufficient set of $k = 3$ alternating constraints is $x_{i,t} - x_{i+1,t} + x_{l,t} \leq 1\ \forall\ i + 1 < l$. There are $\frac{(n-1)(n-2)}{2}$ constraints in each period in this set.*

Proof. Let $B = \{x \in \{0,1\}^n \mid x_{i,t} - x_{i+1,t} + x_{l,t} \leq 1\ \forall\ i + 1 < l\}$.

Following the proof for Theorem 3.1, we first recognize that since B is a subset of A, every consecutive vector satisfies the subset of the $(k = 3)$ alternating constraints.

For the reverse, take $x \in B$. By definition, x satisfies the subset of the $k = 3$ alternating constraints. For $x \in \{0,1\}^n$ the only way to fail the alternating constraint is to have the triple $\{x_{i,t}, x_{j,t}, x_{l,t}\} = \{1,0,1\}, \forall\ i < j < l$.

Since $x \in B$, for every pair $\{x_{i,t}, x_{l,t}\}, i < l$, if $x_{i,t} = 1$ and $x_{l,t} = 1$ then $x_{i+1,t} = 1\ \forall\ i + 1 < l$. Thus if $x_{i,t} = 1$ and $x_{l,t} = 1$, then $x_{i+1,t} = 1$. Now we have $x_{i+1,t} = 1$ and $x_{l,t} = 1$, so $x_{i+2,t} = 1$. Thus each pair $\{x_{i+u,t}, x_{i+u+1,t}\}$ will both be $1\ \forall\ u < l - 1 - i$. This means that there are no "gaps" in the 1's, and thus $x \in C$ and $B \subseteq C$. □

Separation of Alternating Constraints

There are potentially a huge number of alternating constraints, roughly 2^{n-1}, the number of odd subsets of $\{1, \ldots, n\}$.. If these constraints do describe the convex hull of the contiguity constraint, it may be more useful to employ a "Branch and Bound" strategy (Wolsey 1998). This would require

a separation algorithm to determine if a produced x vector were in the hull, and if not, which constraints it violated.

The following algorithm is a first attempt at a separation algorithm for the $k = 3$ alternating constraints. The idea is to start with the $x(i)$ entries with the highest value. If adding them together does not get us over the 1 hump, then there is no chance of subtracting something and staying over 1.

Algorithm for $k = 3$ alternating constraints

Start with vector $x(i), 1 \leq i \leq n$ with i designating the position in the tunnel, $i = 1$ at one end and $i = n$ at the other.

Create $d(j)$, a sorting of x, such that $d(j)$ gives the position in the tunnel of the j^{th} largest value of all $x(i)$, i.e., $d(1) = a$ if $x(a) \geq x(k) \ \forall \ k = 1, \ldots, n$ and $d(n) = a$ if $x(a) \leq x(k) \ \forall \ k = 1, \ldots, n$

Start with the highest two x values, if their sum is not greater than one, stop. If they do, look for an x located between them in the tunnel that, when subtracted from the sum, does not bring the total below 1. If one cannot be found, go on to next highest value and try again.

procedure $k = 3(x, n)$

if $x(d(1)) + x(d(2)) \leq 1$ **then**
  STOP {will never find a triple to violate the constraints.}
end if
if $x(d(1)) + x(d(2)) - x(d(n)) \leq 1$ **then**
  STOP {will never find a triple to violate the constraints.}
end if
for $j = 1 : n - 1$ **do**
  if $x(d(j)) + x(d(j + 1)) \leq 1$ **then**
    STOP {no more options of finding a triple}
  end if
  for $l = j + 1$ to N **do**
    if $x(d(j)) + x(d(l)) > 1$ **then**
      {continue and look for a middle value to complete the triple}
      for $h = \min\{d(j), d(l)\} + 1$ to $\max\{d(j), d(l)\} - 1$ **do**
        if $x(d(j)) - x(h) + x(d(l)) > 1$ **then**
          {have violation, record and proceed to next h}
        end if
      end for{next h}
    end if
  end for{next l}
end for{next j}

Running Time: The actual running time will be a function of the num-

ber of constraints that are violated. To estimate the worst-case run time, consider that all triples are checked, so the run time is $O(n^3)$ as there are n^3 triples.

Related Work

In the unit commitment problem Rajan (2005), machines are scheduled to meet known future demand at minimum cost. For some machines, for example electric generators, there are requirements on the minimum time that they must be run, and the minimum time that they must be down before restarting. If the operating status of a generator is modelled using a binary variable for each period, then the problem is to create sequences of ones and zeros that meet the demand and are long enough to meet the minimum up and down times. Lee (2003) developed a set of constraints for the relaxation of this problem which they called the "alternating up inequalities" (for the time periods the machine is running) and "alternating down inequalities" (for the periods the machine is down) and presented a separation algorithm that runs in linear time. Rajan (2005) studied an extension of this problem in which there are start-up and shut-down costs. In order to track these costs, an additional binary variable that indicates if a machine is started up in a given period is needed. This new variable is then used in a set of "turn on/off inequalities" which limit the number of start-ups/shut-downs in each interval of length minimum up time /minimum down time. This new set of inequalities dominates the "alternating up/down" inequalities, and it is also "much smaller in size". A linear time separation algorithm is presented for use in a branch-and-cut algorithm. This additional variable and set of inequalities is a more general form of the second formulation we developed for the single tunnel problem, which is presented below.

3.3.2 Extended Formulation

The idea behind the extended formulation is based on the spatial setting of the draw points in the tunnel and the requirement that a group of opened draw points must have a starting point, or first opened draw point, as one moves along the tunnel in order of increasing draw point number. The beginning of a set of open draw points is characterized by moving from an unopened draw point to an opened one. Setting a limit of only one such transition results in one contiguous set of open draw points. We have called it an extended formulation as an additional decision variable that marks the draw point where the transition occurs, has been added.

Single Period

In the single tunnel, single period setting, the problem is to determine which contiguous set of draw points to open. Since there is only one period, either a draw point is open/active in the period or remains unopened; we need not be concerned about draw points that are open/inactive.

Data : p_i, w_i, λ, m

n number of draw points in tunnel

Decision Variables : $S_{i,t}$

$F_{i,t}$ Binary decision variable modelling the "first" open draw point in the tunnel in period t, $1 \leq i \leq n$.

$$F_{i,t} = \begin{cases} 1 & \text{if draw point } i \text{ is the first open draw point along the tunnel in period } t, \\ 0 & \text{otherwise} \end{cases}$$

Then since there is only one period and hence no discounting, the objective is

max $\sum_{i=1}^{n} w_i S_{i,1}$

Constraints :

Start Once (3.1)

Global Capacity (3.2)

One First Open at most one first open draw point.

$$\sum_{i=1}^{n} F_{i,1} \leq 1 \tag{3.3}$$

Definition of First Open i can only be first open draw point if i is open and $i-1$ is unopened, $2 \leq i \leq n$

$$F_{i,1} \geq S_{i,1} - S_{i-1,1} \tag{3.4}$$

Definition of First Open, First Draw point in Tunnel draw point 1 can only be first open draw point if it is open.

$$F_{1,1} \geq S_{1,1} \tag{3.5}$$

First Open Must have Started i can not be first open draw point if unopened, $1 \leq i \leq n$.

$$F_{i,1} \leq S_{i,1} \tag{3.6}$$

The new binary variable $F_{i,1}$ calculates any difference between two draw points in their cumulative operating history, and indicates the beginning of a contiguous group of open draw points. Beginning is defined in the sense of moving down the tunnel of consecutively numbered draw points starting from the lowest numbered one.

Thus, $F_{i,1} = 0$ if both, or neither, draw point i and $i-1$ are open, or if i is open, but $i-1$ is not. This definition is enforced by the *Definition of First Open* (3.4) constraint. *Definition of First Open, First Draw Point in Tunnel* (3.5) is included to indicate when the first open draw point is at the beginning of the tunnel. In order to form a contiguous group, the sum of $F_{i,1}$ $i = 1 \ldots n$ must be 1 (unless nothing is open and the sum would be 0).

This single tunnel, single period problem can be framed as a most profitable path problem with the network shown in Figure 3.9 with one unit available to flow from source to sink. The nodes $\{1, n\}$ represent the draw points in the tunnel and the nodes $\{1', n'\}$ represent the open draw points. Leaving the source, the unit chooses the path $F_{s,1},\ s \in \{1, n\}$, the variable

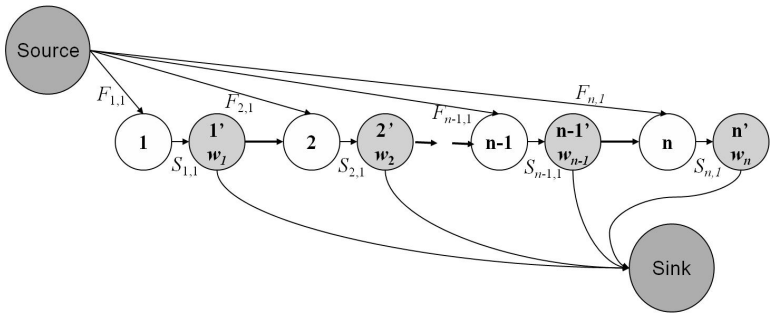


Figure 3.9: Network Representation of Single Tunnel in One Period

indicating which draw point is the first open in the tunnel in the first period. Next the unit continues along the horizontal path until the node denoting the last open draw point, $t,\ t \in \{s, n\}$ is open, before travelling to the sink. As a result, arcs $S_{s,1}, S_{s+1,1}, \ldots, y_t$ are chosen and set to 1.

This problem has an integral optimal solution when the binary constraints on $F_{i,1}$ and $S_{i,1}$ are relaxed. We can show this by first showing that a simple network flow with integral capacities has an integral optimal

solution. Then we show a transformation from the simple network to that in Figure 3.9.

Figure 3.10 shows a simple network with a source and sink and n nodes. The flow on each of the arcs are given by the variables below.

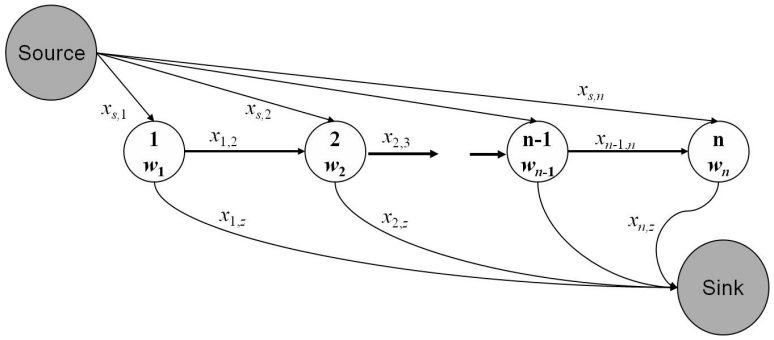


Figure 3.10: Simple Network

let

$x_{s,i}$ be the flow from the source to draw point i

$x_{i,z}$ be the flow from draw point i to the sink

$x_{i,j}$ be the flow from draw point i to draw point $j = i + 1$

Then the objective is

max $\sum_{i=1}^{n} w_i(x_{s,i} + x_{i-1,i})$

subject to

$\sum_{i=1}^{n} x_{s,i} \leq 1$	send at most one unit of flow from source
$x_{s,1} = x_{1,2} + x_{1,z}$	flow balance on node 1
$x_{s,i} + x_{i-1,i} = x_{i,i+1} + x_{i,z}\ 2 \leq i \leq n$	flow balance on node i

The Integral Flow Theorem, Dantzig (1956), states that if the capacities of a network are integers, then there exists an integral maximum flow.

The integrality of the optimal solution can also be shown through unimodularity. A linear programming problem of the form $\max\{cx : Ax \leq b, x \in \mathbb{R}^n_+\}$ will have an optimal solution that is integral if the matrix A is totally unimodular Wolsey (1998). There are three conditions that are sufficient to show that matrix A is totally unimodular, and we can show they are met with the problem above. First, all elements $a_{ij} \in \{-1, 0, +1\}$, which is true for this problem. Secondly, each column contains a most two nonzero coefficients. If A is constructed in the order of the constraints listed above, we can see that each $x_{s,i}$ will appear in the first row, then again in row for the balance on node i, meeting the condition. Similarly, $x_{i,z}$ only

appears in the row for the balance on node i, meeting the condition. Finally, $x_{i,i+1}$ appears once in the balance on node $i-1$ and once in the balance on node i again meeting the condition. The third condition is that there is a *partition* (M_1, M_2) *of the set* M *of rows such that each column* j *containing two nonzero coefficients satisfies* $\sum_{i \in M_1} a_{ij} - \sum_{i \in M_2 a_{ij}} = 0$ (Wolsey (1998)). If we partition A such that M_1 is the constraint on the flow from source, and M_2 is the remaining constraints, we can satisfy this third condition and show that A is totally unimodular and the problem will have an integral optimal solution.

Next we show the translation from the simple network problem Figure 3.10 to the single tunnel, single period problem Figure 3.9. To show the translation, split each node into two and
let

$$F_{i,1} = x_{s,i}$$
$$S_{1,1} = x_{s,1}$$
$$S_{i,1} = x_{i-1,i} + x_{s,i} \quad \text{the flow between two parts of each node}$$

And we get the formulation for the single tunnel above, which will also have an integral optimal solution.

max	$\sum_{i=1}^{n} w_i(x_{s,i} + x_{i-1,i})$	$=$	$\sum_{i=1}^{n} w_i S_{i,1}$
subject to			
	$\sum_{i=1}^{n} x_{s,i} \leq 1$	becomes	$\sum_{i=1}^{n} F_{i,1} \leq 1$
	$x_{s,1} = x_{1,2} + x_{1,z}$		$S_{1,1} = S_{2,1} - F_{1,1} + x_{1,z}$
			$S_{1,1} \geq S_{2,1} - F_{1,1}$
			$F_{1,1} \geq S_{2,1} - S_{1,1}$
	$x_{s,i} + x_{i-1,i} = x_{i,i+1} + x_{i,z}$		$F_{i,1} + S_{i,1} - F_{i,1} = S_{i+1,1}$
			$-F_{i+1,1} + x_{i,z}\ 2 \leq i \leq n$
			$F_{i+1,1} \geq S_{i+1,1} - S_{i,1}\ 2 \leq i \leq n$

While this result is promising for the extended formulation, the addition of the *Global Capacity* Constraint (3.2) can result in fractional solutions.

Multiple Periods

In a single tunnel, multiple period setting a contiguous set of draw points is opened in the first period. As each active draw point reaches the end of its duration p_i, it becomes open but inactive and an additional draw point may be opened. The newly opened draw points in each period together with the previously opened draw points (both active and inactive) must be part of contiguous set. In each period, the capacity constraint parameter m provides an upper limit on the number of active draw points.

The formulation becomes

Data : n, p_i, w_i, λ, m

T number of periods in the lifetime of the mine

Decision Variables : $S_{i,t}, F_{i,t}$

Now with multiple periods, the discounting appears in the objective function:

max $\sum_{i=1}^{n} w_i \sum_{t=1}^{T} S_{i,t} e^{-\lambda t}$

And the constraints are extended to include multiple periods

Constraints :

Start Once (3.1)

Global Capacity (3.2)

One First Open at most one first open draw point, in each period

$$\sum_{i=1}^{n} F_{i,t} \leq 1 \; \forall \; 1 \leq t \leq T \tag{3.7}$$

Definition of First Open i can only be first open draw point if i is open and $i-1$ is unopened, $2 \leq i \leq n$

$$F_{i,t} \geq \sum_{u=1}^{t} S_{i,u} - \sum_{u=1}^{t} S_{i-1,u} \; \forall \; 1 \leq t \leq T \tag{3.8}$$

Definition of First Open, First Draw point in Tunnel draw point 1 can only be first open draw point if it is open.

$$F_{1,t} \geq \sum_{u=1}^{t} S_{1,u} \; \forall \; 1 \leq t \leq T \tag{3.9}$$

First Open Must have Started i can not be first open draw point if unopened, $1 \leq i \leq n$.

$$F_{i,t} \leq \sum_{u=1}^{t} S_{i,u} \; \forall \; 1 \leq t \leq T \tag{3.10}$$

3.3.3 Computational Results

A series of runs were done to test the practicality of computing with the two formulations. CPLEX 9.1 was used on a computer with an Intel Pentium(R) D CPU 3.20 GHz processor with 1.99 GB of RAM for all computational runs.

	$k = 3$ Alternating Constraints	Extended Formulation
Number of Variables	$D \times T$ (all binary)	$2D \times T$ (half are binary)
Number of Constraints	approx $D^2 \times T$	approx $2D \times T$
Example 1: single tunnel 10 draw points time to solve	300 binary 1,120 constraints **2.23 seconds**	300 binary, 300 continuous 650 constraints **1.59 seconds**
Example 2 single tunnel 23 draw points time to solve	1,150 binary 11,623 constraints **15,701 seconds**	1,150 of each 2,423 constraints **1,513 seconds**

Table 3.1: Table of Single Tunnel Comparison of Alternating Constraints Formulation and Extended Formulation

These runs showed that for small tunnels there was little difference in the computation times in CPLEX between the two formulations. However, when the tunnel got longer, the extended formulation solved much quicker than the alternating constraints. Hence we will use the extended formulation as we move towards multiple tunnel formulations in the next section.

3.4 Multiple Tunnels

3.4.1 Introduction/Overview

With this understanding of the single tunnel problem, we move to the real mine setting of multiple tunnels. In this setting the idea of one contiguous cave still holds within a tunnel, but is expanded to link together activity within adjacent tunnels.

The biggest challenge in this work is the definition of an acceptable cave, and writing the constraints to achieve it. It is not fully clear from discussions with Gemcom exactly just what 'acceptable' means here, and thus there is some trade-off between mathematical tractability and goodness of the model.

The models fall into three main approaches. The first (see Section 3.4.3) is an expansion of the single tunnel formulations discussed in the previous section. Keeping the *Within-Tunnel Contiguity* conditions, the indices of the decision variables are expanded to reflect multiple tunnels. In particular, a second dimension approximately perpendicular to the tunnels and called "across-tunnel" is added. The same ideas for contiguity (either alternating constraints or extended formulation) are applied in the across-tunnel direction to satisfy the *Across-Tunnel Contiguity* constraint.

The second model, (see Section 3.4.4) comes from a suggestion by Malkin & Wolsey (2006). They propose a series of models, but realize that in the 2-dimensional setting (multiple tunnels) it is hard to guarantee the formation of a single contiguous cave. Instead, multiple contiguous caves could be produced. Their suggestion is a model based on a centre point, where only draw points connected to the centre points by open (active or inactive) draw points can be opened.

The third set of models, (see Sections 3.4.5— 3.4.7) is motivated by the desire of practitioners (see e.g., Deiring (2006)) to have a "diamond-shaped" cave. The diamond is defined by its vertices and all draw points within the diamond should be open and will be explained more fully later. Finding the relationship between vertices and the draw points is the challenge of this modelling approach.

3.4.2 Computation and Data Sets

In order to develop the models proposed in this work, Gemcom made available four data sets to us. See Figure 3.11 for the mining footprints of the four data sets. In the figures, the circles represent the draw points and the zig-zag lines represent the tunnels. These four data sets include one footprint (data set P2) that is (almost) convex, but the others display the

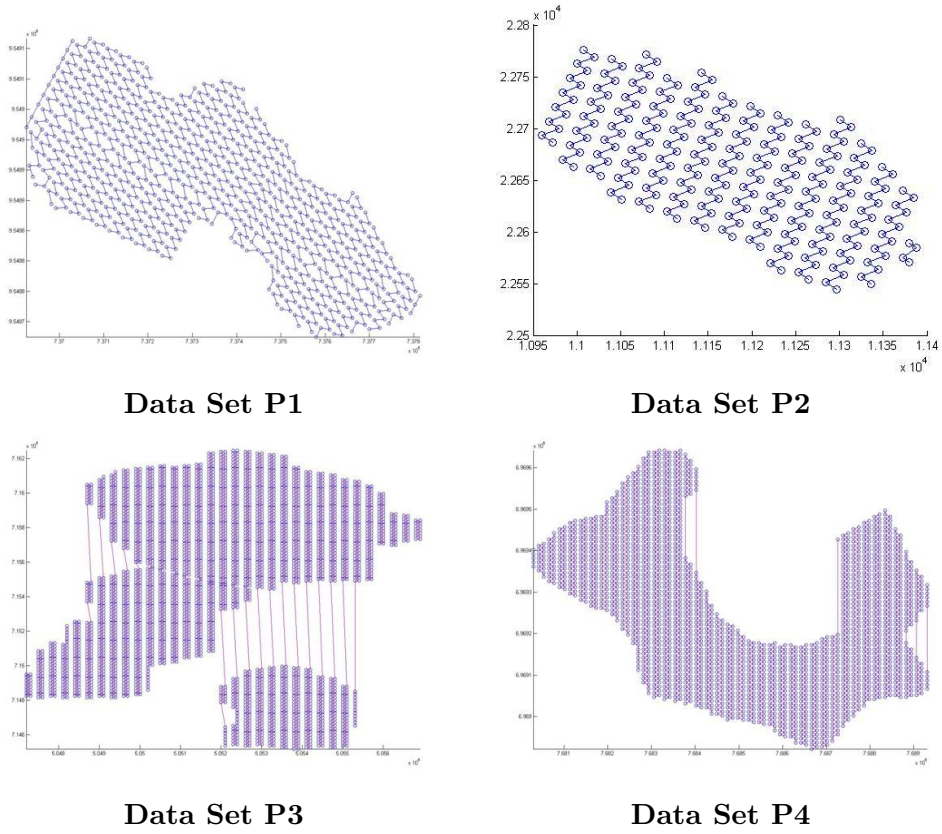


Figure 3.11: Footprints of Data Sets Provided for Model Development

variety of footprint shapes that can be encountered in actual mines. Data set P3 has the additional property of two distinct ore bodies.

The data sets also demonstrate the size variations encountered in mine planning. Table 3.2 show that these sets range from 236 draw points up to 3181 draw points.

Unfortunately this data is confidential and so is not publicly available. Some of these data sets will be referred to in the model description sections later in this document. Multiple data sets were provided to illustrate some of the possible footprint variations, and to aid in model development. Not all of the data sets were used for all models. The performance of all the models will be compared on two additional data sets in Section 3.5. These data sets were not involved in the model development.

	Number of Tunnels	Total Number of Draw Points
Data Set P1	31	780
Data Set P2	14	236
Data Set P3	33	3181
Data Set P4	39	2387

Table 3.2: Size of Data Sets Provided for Model Development

3.4.3 Adapting Single Tunnel Formulations

This section shows the extension of the single tunnel formulations to cover the mine footprint. Unfortunately, the computational results at the end of this section will show that these are not sufficient to guarantee a single contiguous cave.

Basic Model Formulation

In moving from the one tunnel setting to multiple tunnels, the challenge is to incorporate the *Across-Tunnel Contiguity* Constraint. This constraint appeared to have direct parallels to the *Within-Tunnel Contiguity* Constraint in that the cave had to start in one tunnel or a number of parallel tunnels, and gradually grow outwards leaving no gaps. From this idea the single tunnel models were adapted by writing contiguity constraints similar to the *Within-Tunnel Constraints*, for the across-tunnel direction.

We proposed two formulations for single tunnel, alternating constraints and extended formulation. In theory we could apply either of these in the within and across-tunnel directions, for a total of four combinations. The combination of extended formulation for both directions was used since the extended formulation performed so well in the single tunnel setting.

The Basic model formulation is shown below. Note that the draw point index is now two dimensional such that $\cdot, dpt$ is within tunnel and $tn, \cdot$ is across-tunnel. In the draw point numbering example shown in Figure 3.12, the six horizontal tunnels are numbered from bottom to top, and the draw points from left to right. The across-tunnel dimension runs from bottom to top and includes the (tn, dpt), $(tn, dpt + 1)$ pair from each tunnel, for each odd dpt.

Data : T, m, λ

nTNLS the number of tunnels in the mine

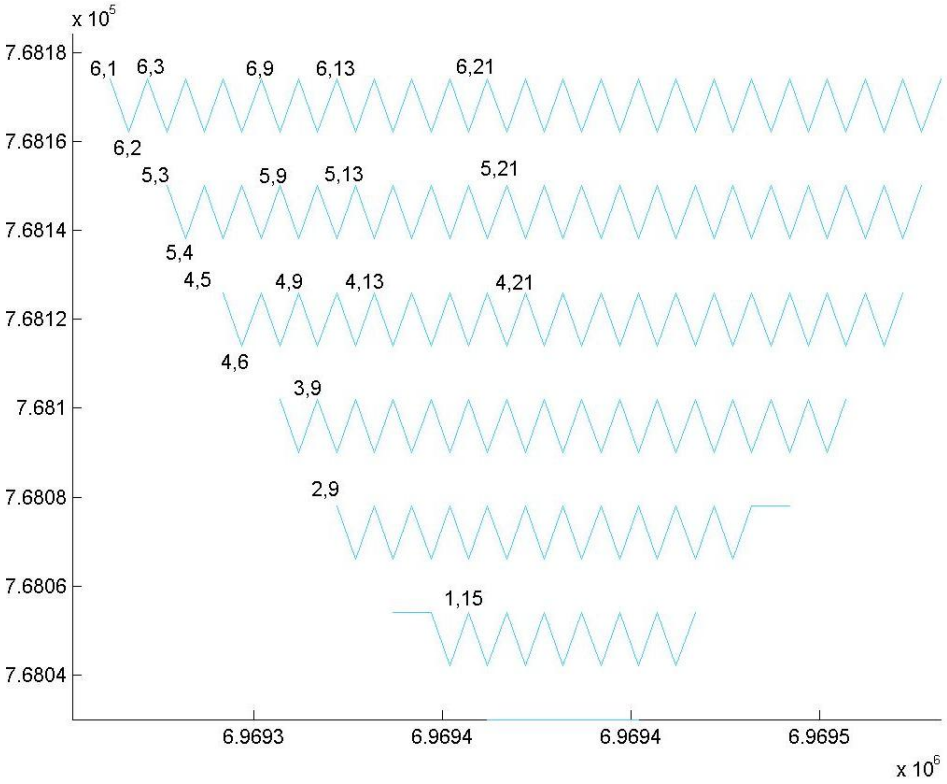


Figure 3.12: Example of Tunnel Numbering

Fdpt$_{tn}$ the number of the first draw point in tunnel tn, $1 \leq tn \leq nTNLS$

Ldpt$_{tn}$ the number of the last draw point in tunnel tn, $1 \leq tn \leq nTNLS$

p$_{tn,dpt}$ the expected duration or periods draw point dpt in tunnel tn will remain open, $1 \leq tn \leq nTNLS$, $dpt = Fdpt_{tn} \ldots Ldpt_{tn}$

w$_{tn,dpt}$ the expected total NPV, if started immediately, of draw point dpt in tunnel tn, $1 \leq tn \leq nTNLS$, $dpt = Fdpt_{tn} \ldots Ldpt_{tn}$

Decision Variables :

S$_{tn,dpt,t}$ binary variable that is 1 in the period the draw point is opened

$\mathbf{F}_{tn,dpt,t}$ binary variable that is 1 if the draw point is the first along the tunnel to be open in the period

$\mathbf{G}_{tn,dpt,t}$ binary variable that is 1 if the draw point is the first across the tunnel to be open in the period

Total Number of (binary) Decision Variables: $nDPTS \times T \times 3$

Objective :

$$\max Value = \sum_{tn=1}^{nTNLS} \sum_{dpt=Fdpt_{tn}}^{Ldpt_{tn}} w_{tn,dpt} \sum_{t} S_{tn,dpt,t} e^{-\lambda(t-1)}$$

Constraints :

Start Once

$$\sum_{u=1}^{T} S_{tn,dpt,u} \leq 1 \; \forall \; tn, \; \forall \; dpt \tag{3.11}$$

Global Capacity

$$\sum_{tn=1}^{nTNLS} \sum_{dpt=Fdpt_{tn}}^{Ldpt_{tn}} \sum_{u=t-p_{tn,dpt}}^{t} S_{tn,dpt,u} \leq m \; \forall \; t \tag{3.12}$$

Define First Along Tunnel A draw point is the first along the tunnel to be open if it is open but the previous draw point is not

$$F_{tn,dpt+1,t} \geq \sum_{u=1}^{t} S_{tn,dpt+1,u} - \sum_{u=1}^{t} S_{tn,dpt,u} \; \forall \; t, \; \forall \; tn, \; \forall \; dpt \tag{3.13}$$

First Along Tunnel Must be Open A draw point must be open to be the first along the tunnel

$$\sum_{u=1}^{t} S_{tn,dpt,u} - F_{tn,dpt,t} \geq 0 \; \forall \; t, \; \forall \; tn, \; \forall \; dpt \tag{3.14}$$

Only One First Along Tunnel Each tunnel can only have one first to be open in each period

$$\sum_{dpt=Fdpt_{tn}}^{Ldpt_{tn}} F_{tn,dpt,t} \leq 1 \; \forall \; t, \; \forall \; tn \tag{3.15}$$

Define First Across Tunnel A draw point is the first across the tunnel to be open if it is open but the previous draw point is not. The next draw point is either in the same tunnel or the adjacent one

$$G_{tn,dpt,t} \geq \sum_{u=1}^{t} S_{tn,dpt,u} - \sum_{u=1}^{t} S_{tn,dpt+1,u} \; \forall \; t, \; \forall \; tn, \; \forall \; dpt \; odd \tag{3.16}$$

$$G_{tn,dpt,t} \geq \sum_{u=1}^{t} S_{tn,dpt,u} - \sum_{u=1}^{t} S_{tn-1,dpt-1,u} \; \forall \; t, \; \forall \; tn, \; \forall \; dpt \; even \tag{3.17}$$

First Across Tunnel Must be Open A draw point must be open to be the first across the tunnel

$$\sum_{u=1}^{t} S_{tn,dpt,u} - G_{tn,dpt,t} \geq 0 \; \forall \; t, \; \forall \; tn, \; \forall \; dpt \tag{3.18}$$

Only One First Across Tunnel

$$\sum_{tn=1}^{nTNLS} (G_{tn,dpt,t} + G_{tn,dpt+1,t}) \leq 1 \; \forall \; t, \; \forall \; dpt \; odd \tag{3.19}$$

Total Number of Constraints : $nDPTS{+}T{+}nDPTS{\times}T{\times}4{+}nTNLS{\times}T{\times}2$

Results

We had success with data set P1 (see Section 3.4.2); the formulation produced a single, contiguous cave that grew over time. The progression is shown in Figure 3.13

Unfortunately, the results for data set P2 were not so nice, see Figure 3.14. In the first period the opening pattern has two distinct caves, violating our goal of a single, contiguous cave.

As this result does not violate the constraints as written in the formulation, it points out a failing of the formulation. Conceptually, in order to guarantee a single, contiguous cave these constraints would have to be applied across every possible axis through the mining footprint. Our formulation only addresses two axes, along and across the tunnels and hence does

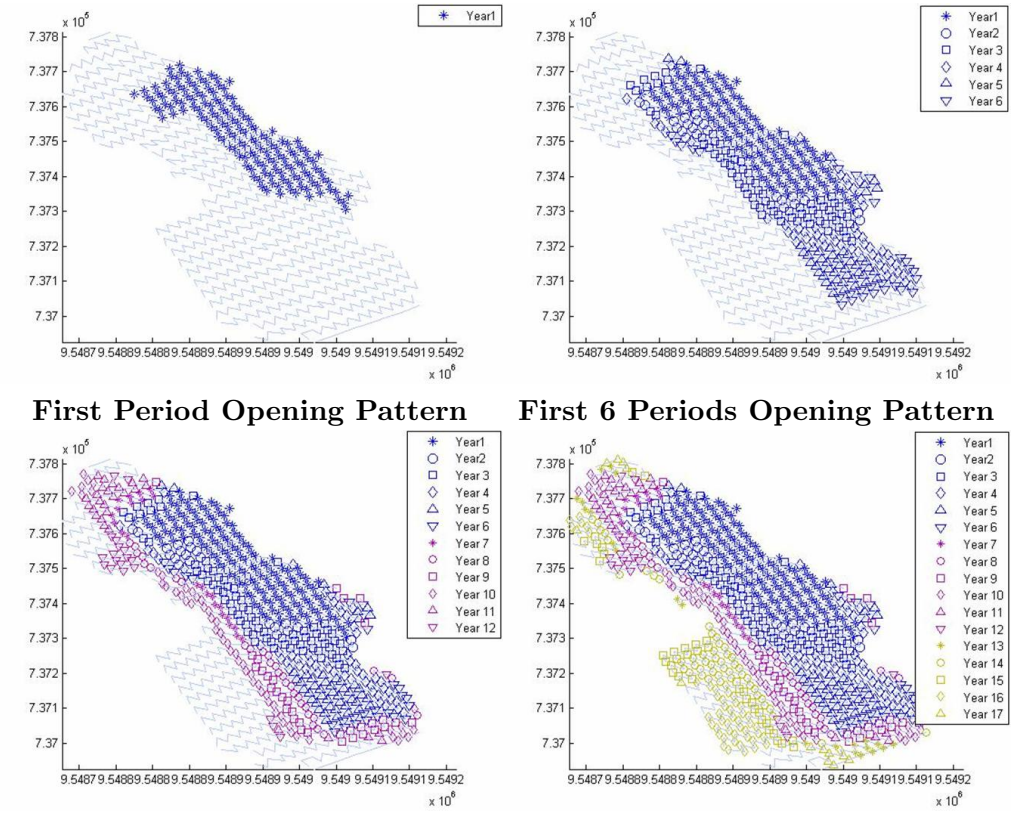
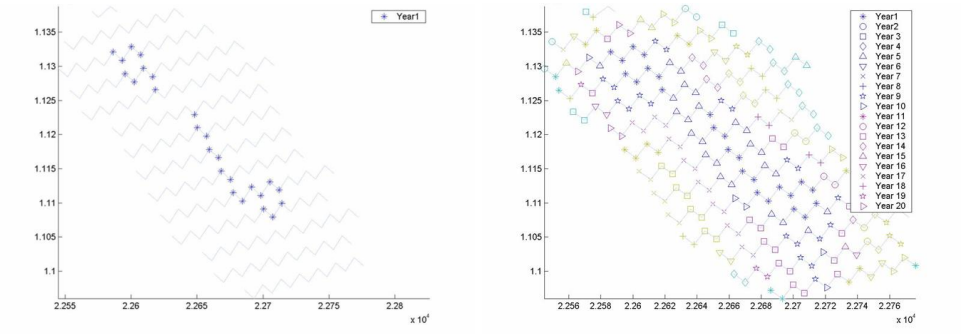


Figure 3.13: Opening Patterns from Data Set P1 using Extended Formulation Within-Tunnels and Across-Tunnels

not guarantee a single cave. It apparently would take a much bigger, more complicated formulation to try to truly enforce contiguity.

Even though this model does not guarantee a single cave, it could prove useful to the mine planner. In Section 3.5 the performance of this model is compared to others we developed for this problem. If a single cave is produced, it is a feasible solution. If a single cave is not produced, the solution highlights high value draw point groups in the mining footprint. The result can also form an upper bound against which solutions from other models can be compared.

The appearance of multiple caves in data set P2, but not in data set P1 led us to rely more heavily on data set P2 than data set P1 for the development of the other models.



First Period Opening Pattern **First 20 Periods Opening Pattern**

Figure 3.14: Opening Patterns from Data Set P2 using Extended Formulation Within-Tunnels and Across-Tunnels

Contribution

The Basic formulation presented in this section is a new idea I developed for this thesis work. The formulation and implementation is new work on the problem of Sequence Optimization and is part of the contribution of this thesis to the research literature.

3.4.4 Malkin and Wolsey's 2D Integral Formulation

Motivated by the possibility of obtaining a disconnected solution from the two-dimensional solution proposed in Malkin (2006), Malkin and Wolsey proposed an integral two-dimensional formulation (Section 3.1). This formulation chooses a starting or centre point with coordinates (r, c), and ensures that the row and column of this centre point form a dominant axis of the opened draw points in each period. They write "The row with the largest interval must be r and the column with the largest interval must be c" and "The region somewhat resembles a diamond" which is good in practice.

Formulation

The formulation suggested by Malkin and Wolsey, translated to our standard variables, is as follows

Data :

r, c the coordinates of the centre point, where the diamond axes intersect

Decision Variables : $S_{tn,dpt,1}$

Constraints : The column-wise neighbour between the draw point and column c must be open to open draw point tn, dpt:

$$S_{tn,dpt,1} \leq S_{tn,dpt+1,1} \; \forall \; tn, \; \forall \; dpt < c \tag{3.20}$$

$$S_{tn,dpt,1} \leq S_{tn,dpt-1,1} \; \forall \; tn, \; \forall \; dpt > c \tag{3.21}$$

Similarly, the row-wise neighbour between the draw point and row/tunnel r must be open to open draw point tn, dpt:

$$S_{tn,dpt,1} \leq S_{tn+1,dpt,1} \; \forall \; tn < r, \; \forall \; dpt \tag{3.22}$$

$$S_{tn,dpt,1} \leq S_{tn-1,dpt,1} \; \forall \; tn > r, \; \forall \; dpt \tag{3.23}$$

Adapting to Mining Layout

The formulation proposed by Malkin has two apparent drawbacks. The first is that the centre point must be specified. This can be overcome by running the formulation using each draw point as the centre point ($nDPT$ times) and evaluating the objective function for each centre point. This may be feasible for a smaller mine but may be too time consuming for the larger mines. Alternatively, a heuristic could be developed to select good centre points. The second drawback is that since the axes are set by the centre point, the axes for the diamond are fixed for the duration of the mine. In reality, a better schedule might result from allowing the axes to vary over time as is possible under current methods.

This formulation requires two approximately perpendicular axes which can be realized in the tunnels and the across direction discussed in the previous section. We use the convention that in the mining data, the "rows" are the tunnels and the "columns" are the across-tunnel pairs of draw points. Using the naming convention (tunnel, across), the draw points zigzag down the tunnels as shown earlier in Figure 3.12.

If we choose the centre point $4, 21$ we get the axes for the diamond as shown by the dark line in Figure 3.15. It might seem that the results from centre point $4, 21$ and centre point $4, 22$ would be the same, but this is not necessarily true. For example, if draw point $4, 22$ has a low value, a cave generated from centre point $4, 21$ may not include $4, 22$ and draw points in adjacent tunnels. However, draw points on these tunnels may be included if low valued $4, 22$ were chosen as the centre point, resulting in a different cave.

Malkin Model Formulation

The Malkin formulation is as follows

Data : $nTNLS, Fdpt_{tn}, Ldpt_{tn}, T, m, p_{tn,dpt}, w_{tn,dpt}, \lambda$

TnC,DptC the coordinates of the centre point, where the diamond axes intersect

Decision Variables : $S_{tn,dpt,t}$

Total Number of (binary) Decision Variables: $nDPTS \times T$

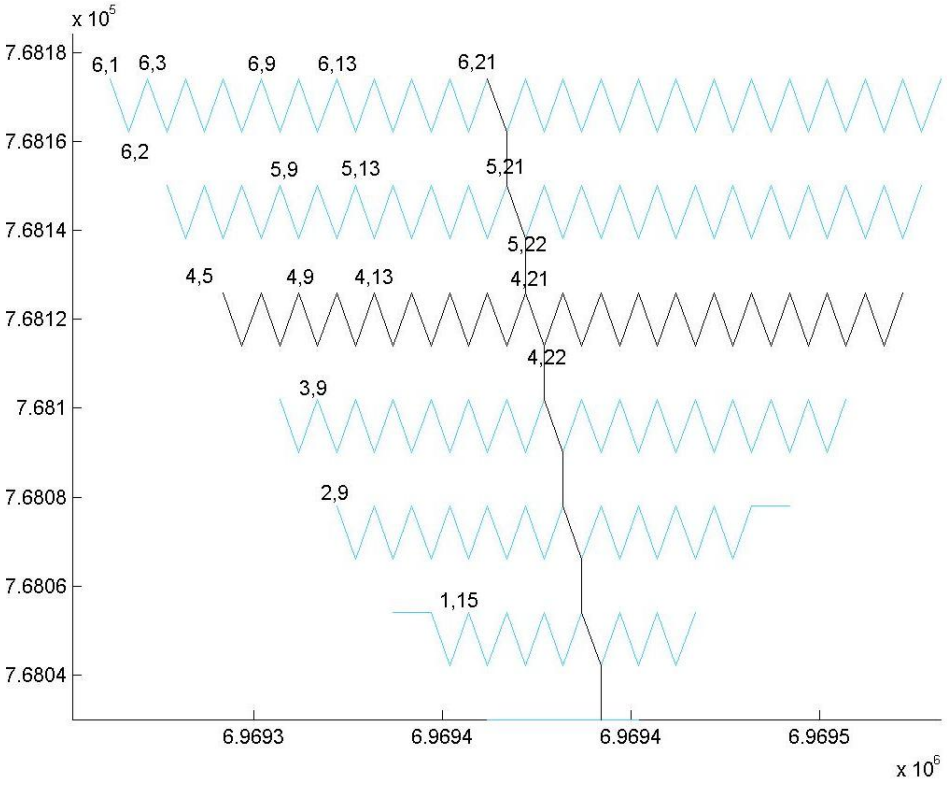


Figure 3.15: Example of Centre Point 4,21

Objective :

$$\max Value = \sum_{tn=1}^{nTNLS} \sum_{dpt=Fdpt_{tn}}^{Ldpt_{tn}} w_{tn,dpt} \sum_{t} S_{tn,dpt,t} e^{-\lambda(t-1)}$$

Constraints :

Column Connected The column-wise neighbour between the draw point and column c must be open in order to open draw point i, j:

$$\sum_{u=1}^{t} S_{tn,dpt,u} \leq \sum_{u=1}^{t} S_{tn,dpt+1,u} \; \forall \; tn, \; \forall \; dpt < DptC, \; \forall \; t \qquad (3.24)$$

$$\sum_{u=1}^{t} S_{tn,dpt,u} \leq \sum_{u=1}^{t} S_{tn,dpt-1,u} \; \forall \; tn, \; \forall \; dpt > DptC, \; \forall \; t \quad (3.25)$$

Row Connected The row-wise neighbour between the draw point and row r must be open to open draw point i, j:

$$\sum_{u=1}^{t} S_{tn,dpt,u} \leq \sum_{u=1}^{t} S_{tn+1,dpt+1,u} \; \forall \; tn < TnC, \; \forall \; dpt \; odd, \; \forall \; t \quad (3.26)$$

$$\sum_{u=1}^{t} S_{tn,dpt,u} \leq \sum_{u=1}^{t} S_{tn,dpt-1,u} \; \forall \; tn < TnC, \; \forall \; dpt \; even, \; \forall \; t \quad (3.27)$$

$$\sum_{u=1}^{t} S_{tn,dpt,u} \leq \sum_{u=1}^{t} S_{tn,dpt+1,u} \; \forall \; tn > TnC, \; \forall \; dpt \; odd, \; \forall \; t \quad (3.28)$$

$$\sum_{u=1}^{t} S_{tn,dpt,u} \leq \sum_{u=1}^{t} S_{tn-1,dpt-1,u} \; \forall \; tn > TnC, \; \forall \; dpt \; even, \; \forall \; t \quad (3.29)$$

Start Once (3.11)

Global Capacity (3.12)

Total Number of Constraints: $nDPTS + T + nDPTS \times T \times 2$

Figure 3.16 illustrates the *Column Connected Constraints* in the tunnel setting. The arrows point to the draw point that must be open before the indicated draw point can open. The dark line shows the across-tunnel axis, and it is evident that all the arrows point towards this axis. The draw points directly on the axis, in this example those numbered 21, do not have constraints.

Figure 3.17 shows the *Row Connected* constraint with the second axis at tunnel 4, shown in black. Again, the arrows point to the draw point that must be open before the indicated draw point can open. It is evident that all the arrows point towards the within-tunnel axis, tunnel 4. The draw points directly on the axis do not have constraints.

The combined picture is presented in Figure 3.18

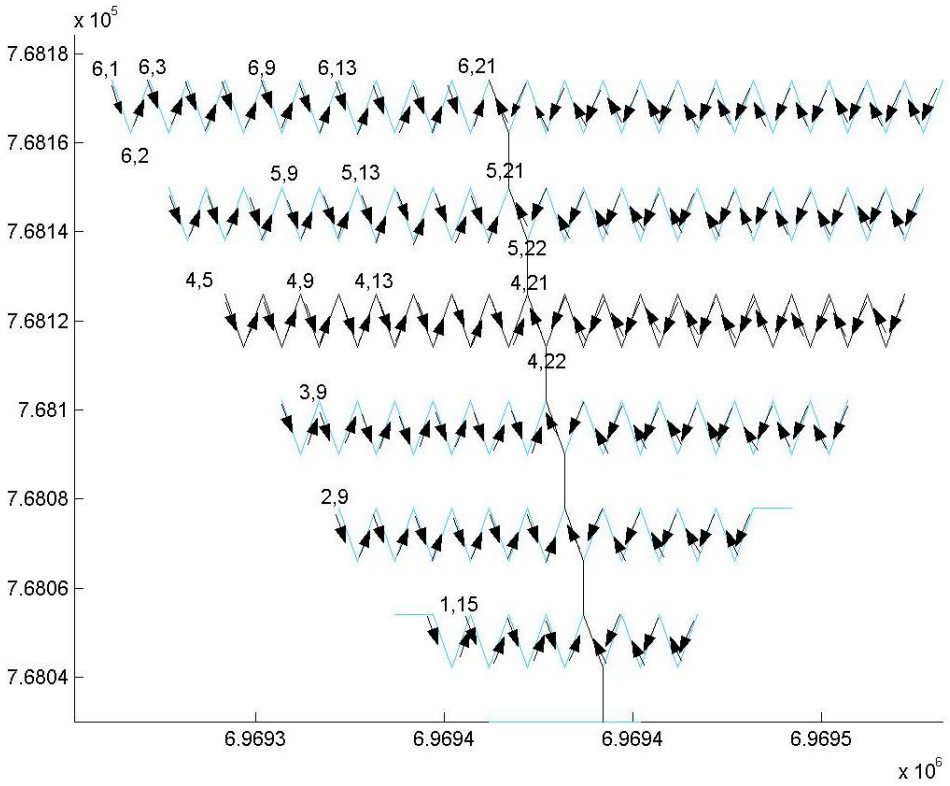


Figure 3.16: Example of Centre Point 4,21 and Direction of the *Column Connected* Constraint((3.24)-(3.25)). Each arrow represents a constraint and points to the draw point that must be opened before the draw point at the tail of the arrow can open.

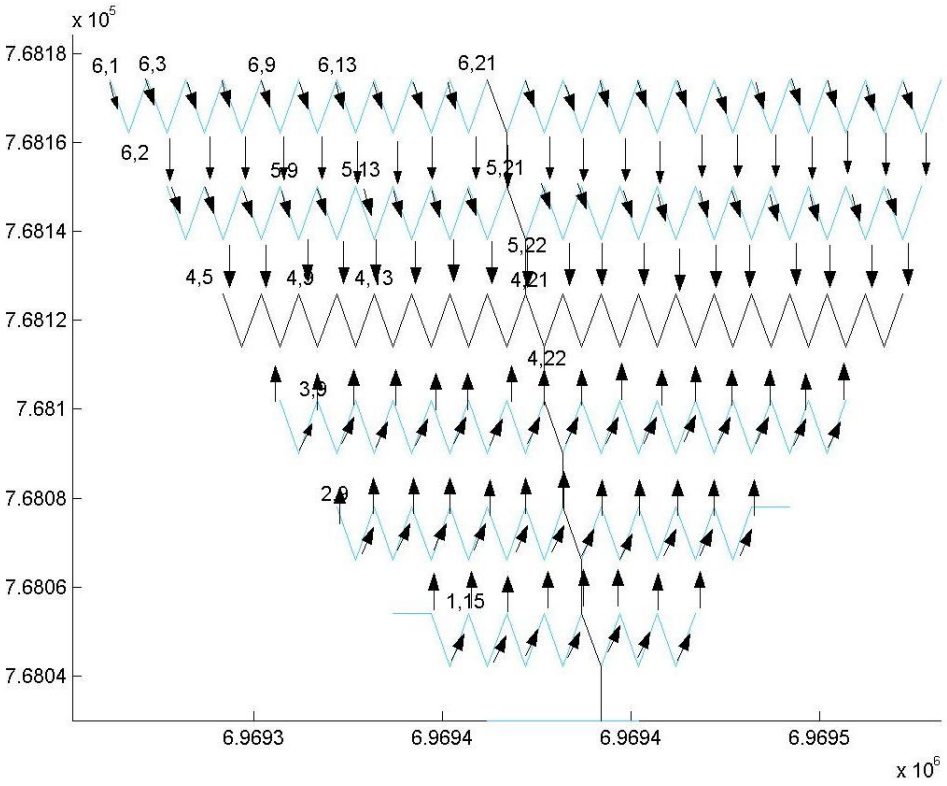


Figure 3.17: Example of Centre Point 4,21 and Direction of the *Row Connected* Constraint((3.26)-(3.29)). The arrows show the draw point that must be open before the draw point at the tail of the arrow can open.

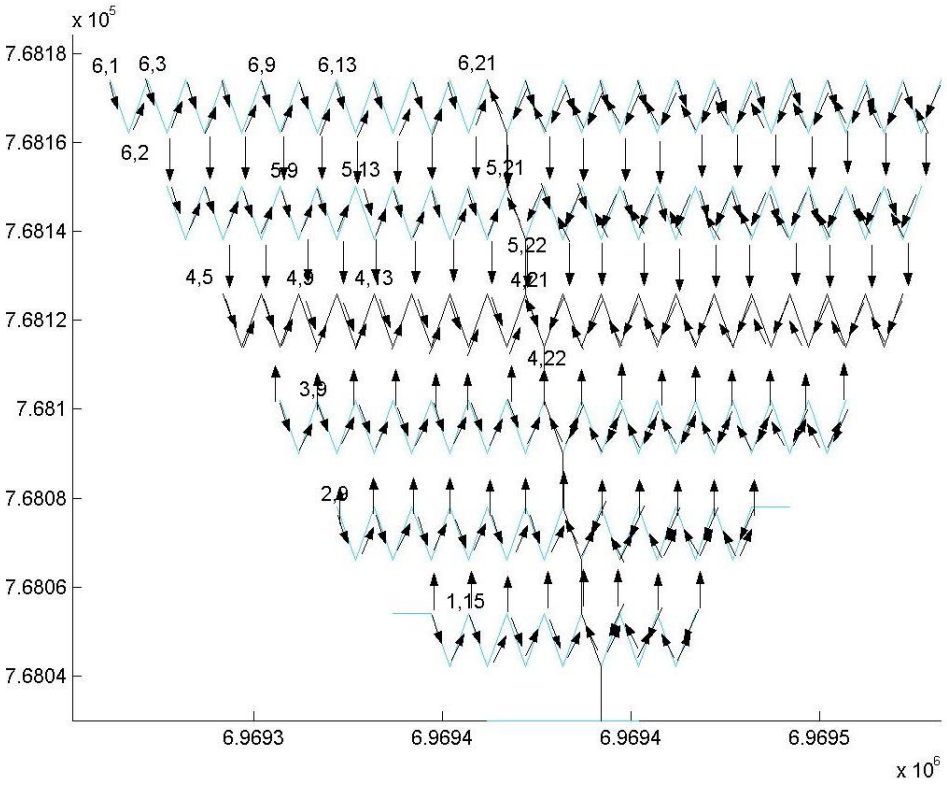


Figure 3.18: Example of Centre Point 4,21 and Arrows for All Constraints

Theoretical Properties

The basic formulation of the one-period model as proposed by Malkin and Wolsey, (3.20) — (3.23), gives an integral solution. A linear programming problem of the form $\max\{cx : Ax \leq b, x \in \mathbb{R}^n_+\}$ will have an optimal solution that is integral if the matrix A is totally unimodular Wolsey (1998). It is straightforward to show that A is unimodular and we conclude that the polyhedron is integral.

The multiperiod formulation presented is more complicated for several reasons. First, the simple neighbour constraints are extended to the multiperiod setting. Secondly, the *Start Once* and *Global Capacity* constraints are added.

Nominally, the multiperiod constraints say that a given draw point can not open in period t unless the neighbour closer to the axis started in period t or prior. These could be written in the form

$$S_{tn,dpt,t} \leq \sum_{u=1}^{t} S_{tn,dpt+1,u} \; \forall \; tn, \; \forall \; dpt < DptC. \tag{3.30}$$

Instead we have chosen to use an aggregated form of the constraints (3.24) — (3.29) which we will show is a tighter formulation. In the aggregated form, the constraints say a given draw point can only have started in period t or earlier if the neighbour closer to the centre point started in period t or prior, for all periods $1 \leq t \leq T$.

To demonstrate the aggregated constraints are stronger than the disaggregated constraints we look at the following example for a single draw point.

Period one In period one both constraints are the same

$$S_{tn,dpt,1} \leq S_{tn,dpt+1,1}$$

Period two In period two the disaggregated constraint is

$$S_{tn,dpt,2} \leq S_{tn,dpt+1,1} + S_{tn,dpt+1,2}$$

and the aggregated constraint is

$$S_{tn,dpt,1} + S_{tn,dpt,2} \leq S_{tn,dpt+1,1} + S_{tn,dpt+1,2}$$

This constraint is stronger than the disaggregated one as there are two non-negative variables on the left-hand side instead of the single one

in the disaggregated constraint. There is also no possibility of having $S_{tn,dpt,1} = 1$ and $S_{tn,dpt+1,2} = 1$ since the period one constraint would require that if $S_{tn,dpt,1} = 1$ then $S_{tn,dpt+1,1} = 1$ and then the *Start Once* constraint would not allow $S_{tn,dpt+1,2} = 1$

The following example shows a fractional solution that satisfies the disaggregated constraints (3.30) but violates the aggregated constraints (3.24):

$$S_{tn,dpt,1} = \frac{1}{2}, S_{tn,dpt,2} = \frac{1}{2}$$

$$S_{tn,dpt+1,1} = \frac{3}{4}, S_{tn,dpt+1,2} = 0.$$

The disaggregated constraints are satisfied:

$$S_{tn,dpt,1} = \frac{1}{2} \leq S_{tn,dpt+1,1} = \frac{3}{4}$$

$$S_{tn,dpt,2} = \frac{1}{2} \leq S_{tn,dpt+1,1} + S_{tn,dpt+1,2} = \frac{3}{4},$$

but the aggregated constraints are violated

$$S_{tn,dpt,1} = \frac{1}{2} \leq S_{tn,dpt+1,1} = \frac{3}{4}$$

$$S_{tn,dpt,1} + S_{tn,dpt,2} = 1 \not\leq S_{tn,dpt+1,1} + S_{tn,dpt+1,2} = \frac{3}{4}.$$

The multiperiod setting also requires the addition of the *Start Once* constraint. This constraint is required to ensure that the valuable draw points are not opened multiple times in order to increase the objective value. An alternative may be to only restrict the centre point, draw point $TnC, DptC$ to open once.

$$\sum_{t=1}^{T} S_{TnC,DptC,u} \leq 1$$

Combining this constraint with the aggregated neighbour constraints, should limit the sum $\sum_{t=1}^{T} S_{tn,dpt,u}$ to one for all draw points thus ensuring each opens at most once. This constraint was not tested computationally.

Finally, the *Global Capacity* constraint is needed in its multiperiod form to reflect the non-uniformity of draw point durations. If all draw points had uniform duration then this constraint and the overall formulation could be

simplified. With uniform duration p one is only interested in periods $t = kp$ for $k = 1, 2, \ldots \left\lfloor \frac{T}{p} \right\rfloor$ and the *Global Capacity* constraint becomes

$$\sum_{tn=1}^{nTNLS} \sum_{dpt=Fdpt_{tn}}^{Ldpt_{tn}} S_{tn,dpt,t} \leq m \; \forall \; t$$

Initial inspection suggests that the multiperiod formulation may be integral with uniform duration of all draw points, but that with varying durations the *Global Capacity* constraint becomes an obstacle to integrality.

Results

This formulation provides solutions with one, contiguous open cave if the mine footprint is convex. The shape may not always be a diamond, but it is based on two axes.

Sample results from data set P2 are shown below in Figure 3.19. The tunnels are the zig-zag lines running approximately parallel to the horizontal axis and the axes for the formulation are shown in black and intersect at the chosen centre point.

However, when the footprint is not convex, multiple caves can form. This can be seen in Figure 3.20 which shows results on data set P4. The draw points on the horizontal axis of the diamond are only constrained to have a neighbour closer to the vertical axis of the diamond. In the case of a gap in the footprint, the neighbour to the edge draw point is on the other side of the gap, and it opened in period 1. This allows draw points along the horizontal axis of the diamond to open and form an additional cave.

This can be fixed by bending the axes around the footprint. We have chosen to bend the axes around the opening in the mine footprint, then continue in a straight line tangent to the opening. An example of this is in Figure 3.21.

This method finds a solution, and finds it relatively quickly for some centre points in data set P2. Results from runs from data set P2 and data set P4 are shown in Table 3.3 for comparison. The comment Original Axes refers to the setting of the axes from the center point even if they cross a empty section of the mining footprint. The comment Bent Axes means that the axes are bent around any empty sections. There are no such comments for data set P2 as there are no empty sections in this data set. While some runs from data set P2 complete in less than a minute, most runs from data set P4 still take over an hour. This table also indicates that there are some variations in objective values from different centre points. Most of the cases

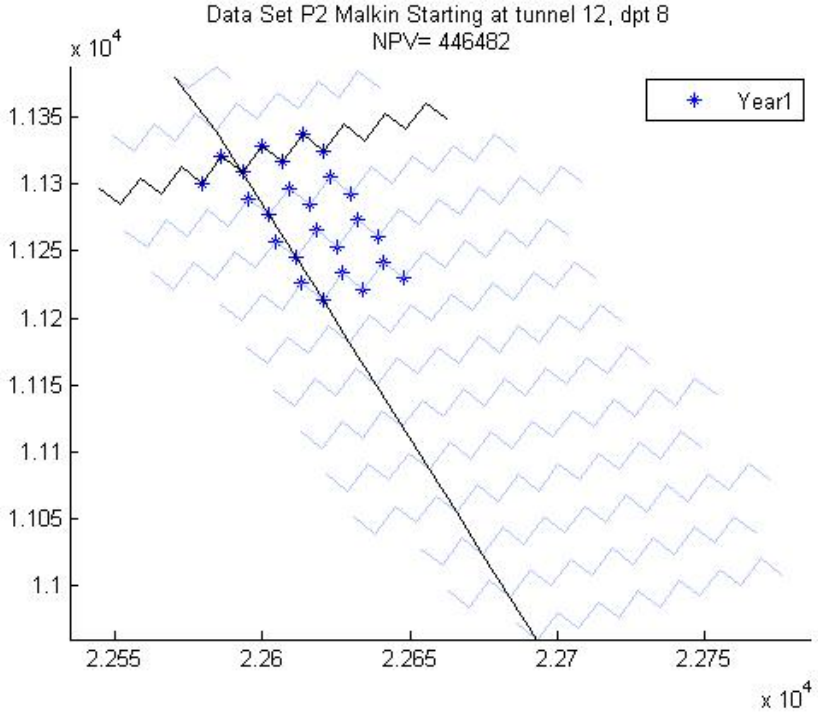


Figure 3.19: Data Set P2, Period 1 Opening

where the axes are bent took longer to solve than those with the original axes, but not in all cases. Since the solutions without bent axes resulted in multiple caves, the extra time is the price for a feasible solution. Similarly, in most but not all cases the bending of the axes increased the objective value.

We are still faced with the problem of finding a good centre point.

Since data set P2 has only 236 draw points, the model was repeatedly run using each draw point as the centre point of the formulation. The results of these runs show the variation in objective value as well as differences in running time over the footprint. Table 3.4 and Figure 3.22 below show some of the results from these runs.

Data Set	Run Time (sec)	Centre Tunnel	Centre Dpt	Total NPV	comments
P2	89.8	5	12	452,746	max Total_NPV
P2	55.2	6	12	451,207	
P2	4,217.9	1	20	412,537	min Total_NPV
P2	11.6	5	10	451,894	fastest run time
P2	25,763.7	10	21	424,758	longest run time
P4	3,178.6	13	90	606,162	Original Axes
P4	5,280	13	90	609,400	Bent axes
P4	4,176.4	14	65	608,322	Original Axes
P4	3,105.6	14	65	608,648	Bent axes
P4	3,076.6	14	95	604,280	Original Axes
P4	4,028.3	14	95	607,859	Bent Axes
P4	4,488	28	135	606,454	Original Axes
P4	6,184.9	31	57	607,713	Original Axes
P4	2,978.6	31	103	609,901	Original Axes
P4	3,605	31	103	609,416	Bent Axes

Table 3.3: Table of Running Times of Malkin Model from Data Sets P2 and P4

Data Set	Run Time (sec)	Centre Tunnel	Centre Dpt	Total_NPV
P2	544.234	1	7	435,414
P2	387.891	1	8	436,801
P2	196.031	1	9	443,198
P2	17.188	1	10	443,198
P2	120.422	1	11	443,232
P2	517.094	1	12	444,724
P2	879.984	1	13	443,340
P2	173.406	1	14	442,610
P2	178.844	1	15	437,418
P2	168.516	1	16	436,455
P2	371.829	1	17	430,991
P2	423.516	1	18	429,554
P2	4714.88	1	19	413,992
P2	4217.95	1	20	412,537
P2	1052.12	2	5	431,796
P2	447.437	2	6	432,407
P2	117.297	2	7	439,908
P2	91.828	2	8	441,360
P2	95.094	2	9	445,872
P2	96.234	2	10	446,274
P2	69.625	2	11	446,743
P2	75.36	2	12	447,803
P2	266.407	2	13	447,159
P2	128.313	2	14	446,376
P2	664.437	2	15	442,037
P2	178.843	2	16	441,616
P2	201.047	2	17	434,607
P2	241.547	2	18	434,960
P2	1398.74	2	19	422,581
P2	2149.45	2	20	421,452
P2	557.906	3	5	436,816
⋮	⋮	⋮	⋮	⋮

Table 3.4: Partial Table of Running Times for Data Set P2

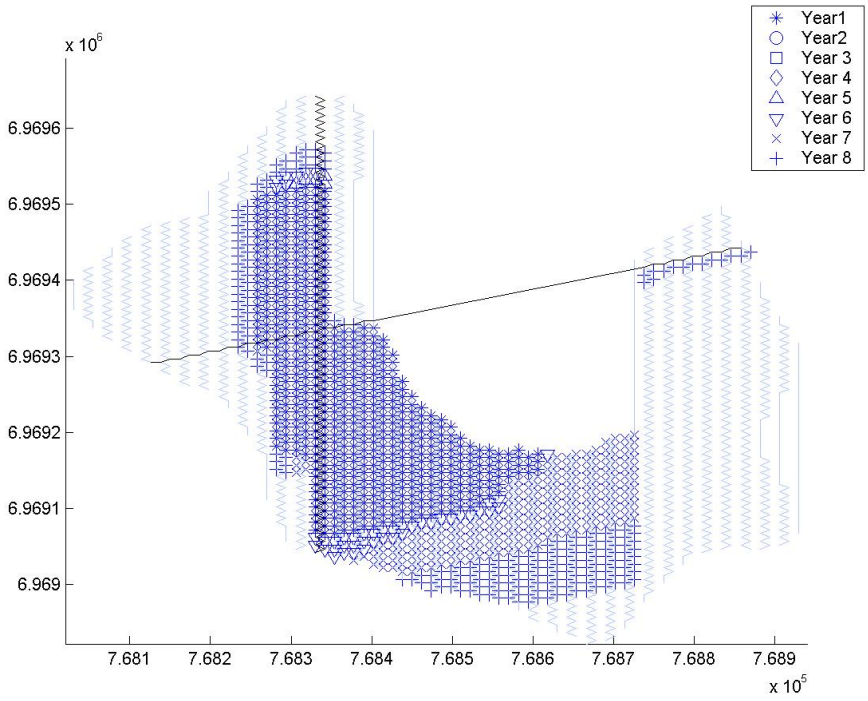


Figure 3.20: Data Set P4, Centre Point (14,65)

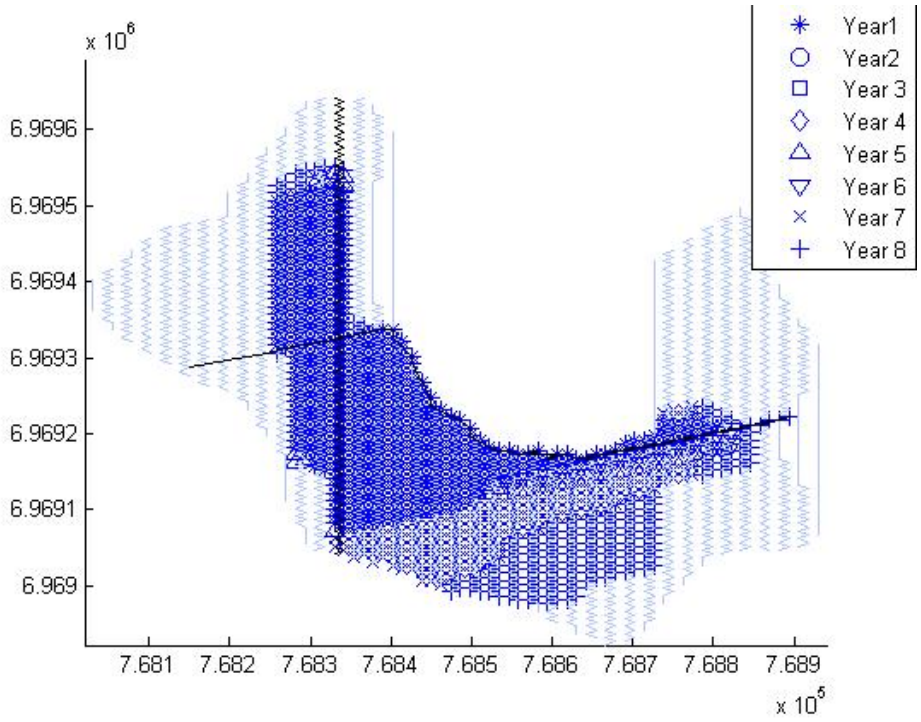


Figure 3.21: Horizontal Axis Bent Around Hole in Footprint, Centre Point (14,65)

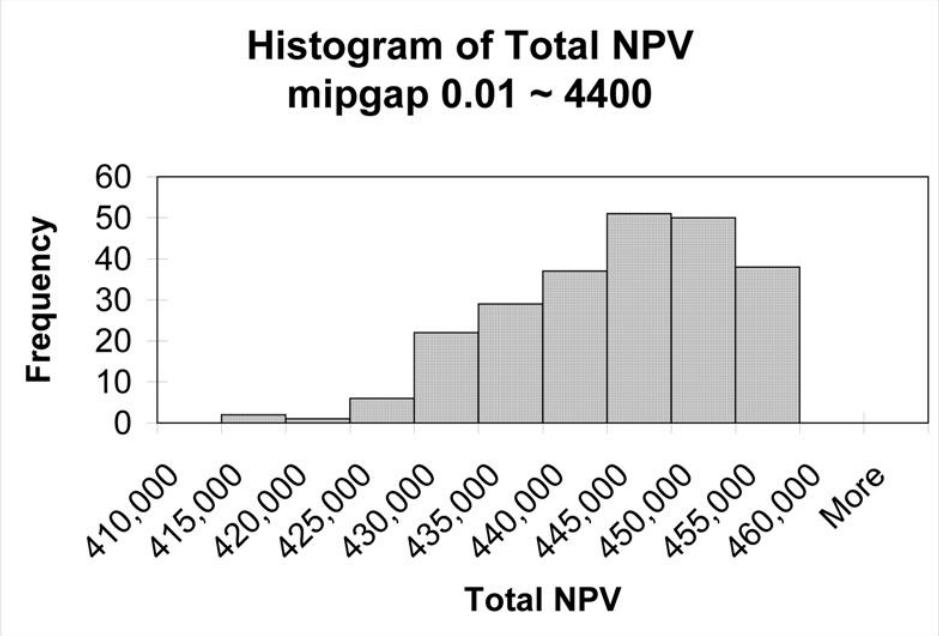


Figure 3.22: Histogram of Total_NPV

There is a range of about 9% from the lowest NPV of 412,537 to the highest NPV of 452,746 for data set P2, so it is worth some effort to find the centre point yielding the highest value. The distribution is shown in Figure 3.22. Note, these runs were done with the CPLEX setting of MIPGAP 0.01. This means that computation is stopped when a solution within 1% of optimality is reached. For this range of objective values, 1% is approximately 4,400 which explains some of the variation in the results.

A surface plot of objective values over the footprint of the mine is shown in Figure 3.23 for data set P2. For this data set, highest objective values are for centre points in the centre of the mine, dropping off quite steeply at the mine edges. While there is some variation at the centre, there is no single, small cluster of high objective value points.

Figure 3.24 is a preliminary attempt to heuristically locate the optimal centre point. Each of the three graphs is a contour plots over the footprint of the mine for data set P2. The draw points are the intersection of the grid lines. The tunnels run horizontally across the plot and the across numbering runs vertically. The plot on the right is for the TOTAL NPV given the draw point is the centre point for the Malkin model. The dark shape in the

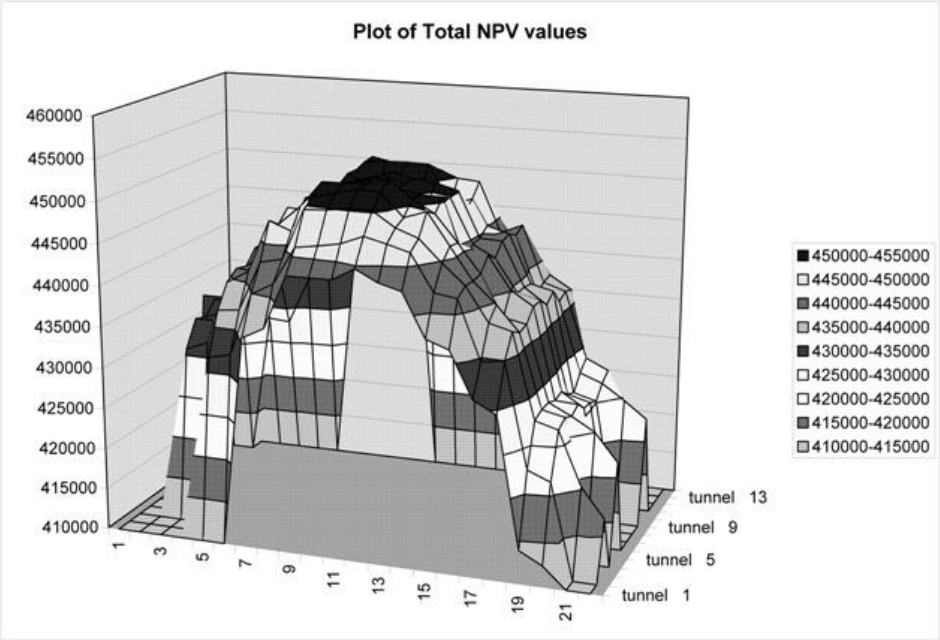


Figure 3.23: Surface Plot of Total_NPV for Data Set P2

centre indicates the highest value draw points. The two plots on the left are representations of the value of the individual draw points. The top one is the ratio $\frac{w_i}{p_i}$, the total value divided by the duration. The lower one is the ratio modified to represent the discount factor and is $\frac{w_i}{\exp(-\lambda p_i)}$. This figure does not support the conjecture that individual draw point value predict TOTAL NPV if used as a centre point. There are three clusters of draw points with the highest $\frac{w_i}{p_i}$ ratio. The top group does not at all overlap with the high TOTAL NPV draw points, and there are perhaps one or two from the other two groups that overlap. There are also three clusters of high $\frac{w_i}{\exp(-\lambda p_i)}$ ratio draw points, but there is only a single draw point (tunnel 4, draw point 10) that is also in the cluster of high TOTAL NPV draw points.

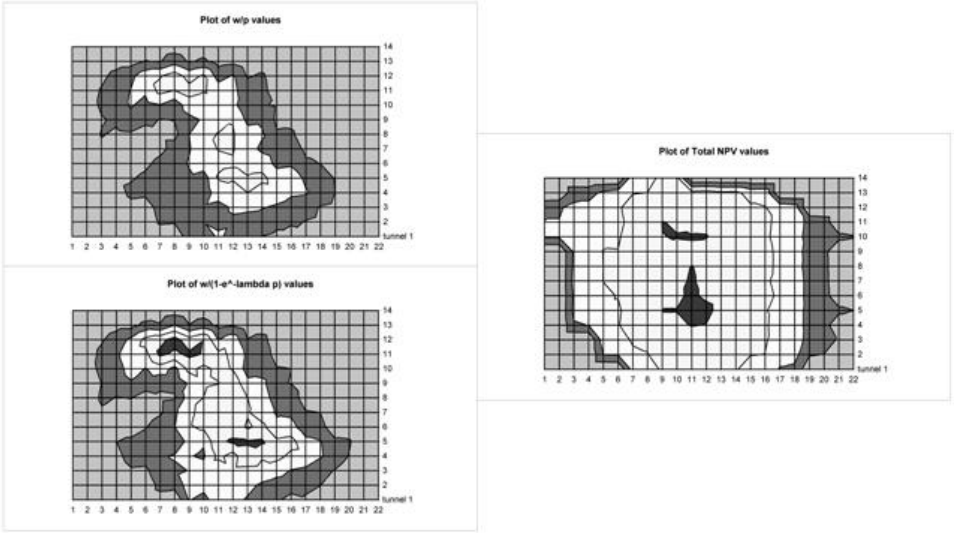


Figure 3.24: Projections of the Surface Plots for Draw Point Values of $\frac{w_i}{p_i}$, $\frac{w_i}{\exp(-\lambda p_i)}$ and Total NPV for Data Set P2

Variation: Wandering Axes formulation

One potential drawback to the formulation proposed by Malkin and Wolsey is that the axes for the diamonds are fixed over the entire duration of the mine. It is not clear whether this is an necessary restriction on the mine growth or not, and it may overly constrain the problem. The following variation of the above formulation allows the axes to deviate from the tunnel and draw point chosen as the centre point. The idea is that if the axes are allowed to deviate, or wander, then the formulation may lead to better objective values; we have named it the Wandering Axes formulation.

The variation is executed by picking a centre point, then deciding where the axes will run. When this has been done, the model runs as above with the constraints requiring neighbours between the draw point and the axes at the location of the draw point to be open before the draw point can start.

The centre point is heuristically chosen as the draw point with the highest average value of a cluster of neighbours, and the progress of the axes are also influenced by the average value of neighbours. The neighbours are defined as those within a diamond shape containing approximately m draw points around the centre point. For simplicity we looked at diamond shapes of size $\frac{n(n-1)}{2} + \frac{n-1}{2}$ when n is an odd integer. The diamond extends $\frac{n-1}{2}$ draw points along the tunnel, in either direction from the centre point. In the adjacent tunnel there are $n-2$ draw points in the diamond. Each adjacent tunnel is included with the number of draw points decreasing by two until $\frac{n-1}{2}$ tunnels away there is a single draw point in the diamond. Since not every centre draw point will have a full diamond of neighbours (for example the draw points at the edge of the mining footprint) the average value over the number of existing neighbours is used. The following procedure was used.

1. For each draw point the average neighbour value was calculated. The total value of the draw point w_i was used. The centre point was chosen as the draw point with the highest neighbour value. For data set P2 a diamond of size 25 was used and the chosen centre point was tunnel 6, draw point 12.

2. The axes started at the chosen centre point $TnC, DptC$ and stayed with the centre tunnel and draw point in the adjacent tunnels and draw points for one step. From here the axis following the centre draw point alternates between choosing the direction of highest neighbour value in one tunnel, holding that same path in the next tunnel, then again deciding the highest value route in the next tunnel. That is, in

the next adjacent tunnels $TnC \pm 2$, the neighbour values of three draw points $(TnC + 2, DptC - 1), (TnC + 2, DptC), (TnC + 2, DptC + 1)$ are compared, and the axis moves to the draw point with the highest neighbour value. The axis stays the same in the adjacent tunnel, but in the next, a similar choice is made. Similarly, for the along tunnel axis a decision is made at $DptC + 2$ between $(TnC - 1, DptC + 2), (TnC, DptC + 2), (TnC + 1, DptC + 2)$ for the draw point with the highest neighbour value, and also at $DptC - 2$ between $(TnC - 1, DptC - 2), (TnC, DptC - 2), (TnC + 1, DptC - 2)$. An example is shown in Figure 3.25

									Drawpoint												
Tunnel	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0	0	0	0	0	0	1,829	2,354	2,775	2,918	3,282	**3,303**	3,250	2,961	2,602	2,163	1,817	1,507	1,267	1,003	0	0
2	0	0	0	0	1,494	1,763	2,001	2,254	2,678	3,151	3,488	**3,673**	3,559	3,347	2,900	2,573	2,114	1,842	1,479	1,171	0	0
3	0	0	0	0	1,665	1,829	2,043	2,370	2,955	3,450	3,842	**4,000**	3,958	3,686	3,372	2,927	2,507	2,032	1,615	1,310	1,017	678
4	0	0	**1,387**	**1,398**	**1,509**	**1,760**	2,092	2,690	3,278	3,810	4,163	**4,372**	4,291	4,126	3,721	3,320	2,786	**2,313**	**1,762**	**1,333**	0	0
5	0	0	1,272	1,296	1,381	1,649	**2,225**	**2,791**	3,548	4,022	4,503	**4,641**	4,605	**4,313**	**3,909**	**3,383**	**2,811**	2,207	1,697	1,327	1,087	816
6	0	0	1,046	1,240	1,366	1,654	2,095	2,693	**3,420**	**4,182**	**4,656**	4,889	**4,611**	4,266	3,638	3,065	2,431	1,930	1,477	1,109	0	0
7	0	0	1,009	1,107	1,398	1,574	2,069	2,599	3,415	4,111	4,717	**4,735**	4,464	3,841	3,230	2,526	1,921	1,432	1,126	797	0	0
8	0	0	1,078	1,241	1,385	1,742	2,200	2,857	3,534	4,190	**4,476**	4,435	3,969	3,360	2,634	2,035	1,427	1,011	695	464	0	0
9	0	0	1,268	1,494	1,767	2,108	2,670	3,259	3,852	4,230	**4,294**	4,048	3,480	2,839	2,104	1,549	1,061	709	428	255	0	0
10	780	1,131	1,370	1,745	2,158	2,709	3,145	3,659	4,016	**4,186**	4,040	3,666	3,048	2,341	1,655	1,138	784	526	313	198	142	116
11	798	1,121	1,456	1,910	2,562	3,040	3,422	3,712	3,952	**3,919**	3,808	3,266	2,598	1,923	1,281	901	622	471	278	178	0	0
12	627	997	1,449	2,001	2,577	3,025	3,281	3,447	3,535	**3,558**	3,295	2,862	2,104	1,492	1,049	757	550	375	0	0	0	0
13	0	0	1,236	1,799	2,293	2,606	2,899	2,956	2,976	**2,880**	2,630	2,181	1,590	1,173	787	579	0	0	0	0	0	0
14	0	0	0	0	0	0	2,449	2,521	**2,512**	2,195	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3.25: Table of Neighbour Values for Data Set P2. Rows are tunnels and columns are draw point numbers. The boxes show the three values considered in each decision and the bold values indicated the selected axes.

In the data set P2 this criterion chose tunnel 6, draw point 12 as the centre point. The results from this run of the model are shown in Table 3.5. Runs from the basic Malkin model with centre points $5, 12$ and $6, 12$ are included for comparison.

Data Set	Run Time (sec)	Centre Tunnel	Centre Dpt	Total NPV	Cplex Optimality gap	comments
P2	54.6	6	12	452,530	0.10%	Wandering Axes
P2	89.782	5	12	452,746	0.02%	basic Malkin
P2	55.156	6	12	451,207	0.36%	basic Malkin

Table 3.5: Comparison of Results of Wandering Axes Variation

The results show that the Wandering Axes variation did not give a higher objective value than the basic Malkin model, in that the values of 452,746 for the basic model and 452,530 for the Wandering Axes variation are within the

optimality gap of CPLEX. The Wandering Axes variation was successful as a single run heuristic of approximating in a single run the maximum objective value obtained by running the basic formulation over all draw points.

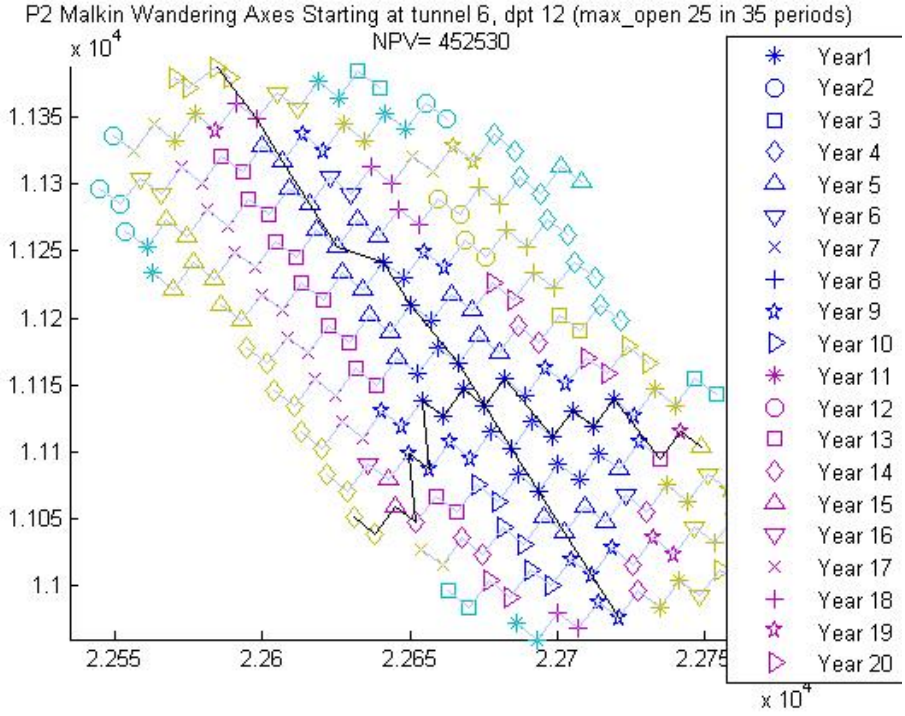


Figure 3.26: Results from the Wandering Axes Model Variation on Data Set P2

Figure 3.26 shows the resulting plan for the mine. The black lines show the axes. While the along tunnel axis does "wander" there is little movement in the across-tunnel axis.

Figure 3.27 shows the resulting plan for the mine from the basic Malkin model centred at tunnel 6 draw point 12. The black lines show the axes and Figure 3.28 compares wandering axis and basic Malkin model results side by side.

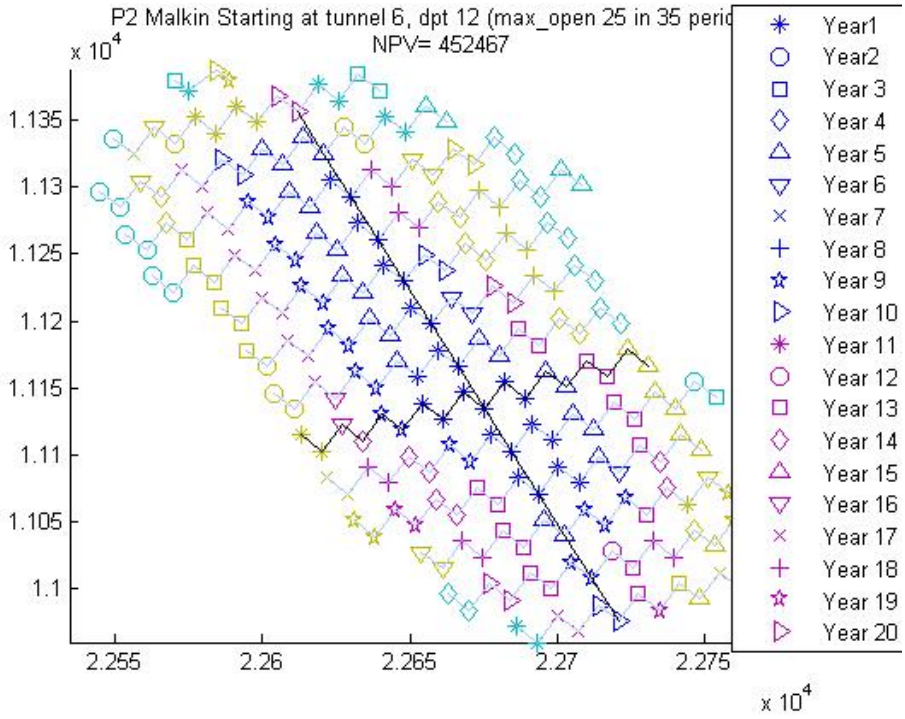
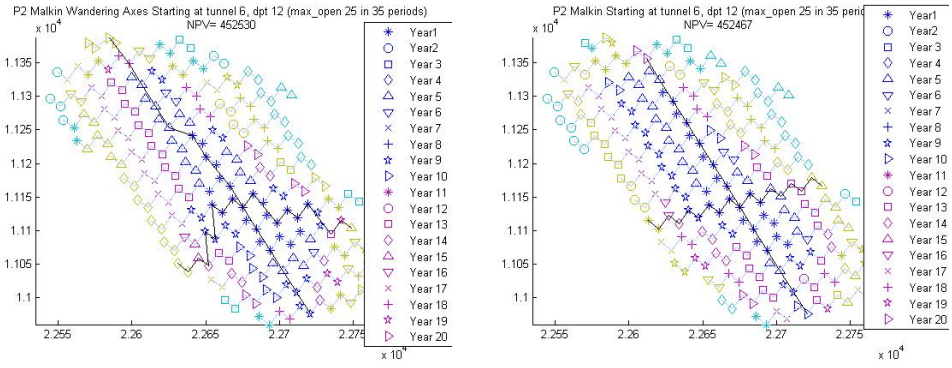


Figure 3.27: Results from the Malkin Model Centered at Tunnel 6, Draw Point 12



(Wandering Axes) **(Basic Malkin)**

Figure 3.28: Same Figures Side by Side

This model runs quickly and does produce a single, contiguous cave. The best value for data set P2 is 452,746. The Wandering Axes heuristic did not produce a higher objective value, but did closely approach the best value over the data set. Results from this data set suggest that the extra work required to determine the changes in the axes was not justified by improved objective value. The formulation does work so could be evaluated on other data sets if there was reason to believe an improvement is achievable.

The Malkin model guarantees a single cave in each period which is an improvement over the Basic model. The main cost of this model is the determination of the fixed centre point. For smaller data sets it may prove feasible to run the model for all draw points. In Section 3.5 the performance of this model is compared to others we developed for this problem.

Contribution

The single period formulation of the Malkin model was suggested by Malkin and Wolsey after discussions with Maurice Queyranne. They have not published this work. Nor, to my knowledge, have they done further work on it or Sequence Optimization. The extension of the formulation to multiple periods and the implementation on data sets is new work I did for this thesis and is part of the contribution of this thesis to the research literature.

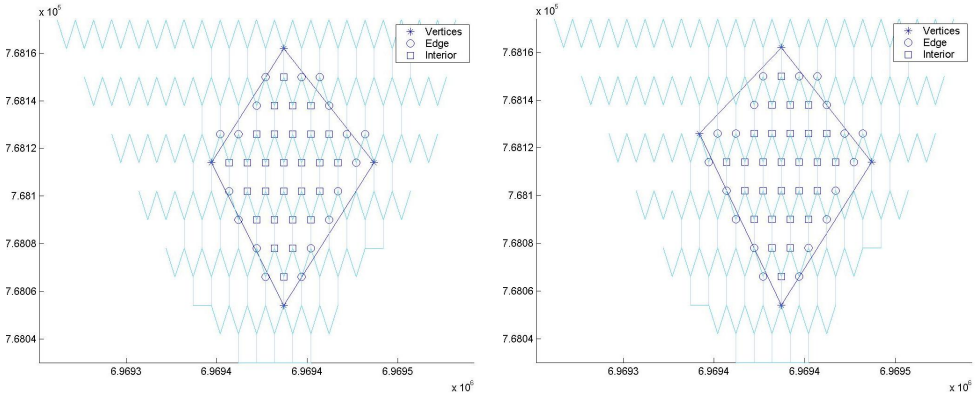
3.4.5 Formulations Based on 4 Vertices

A third approach to formulating this problem is rooted in the operational preference for a diamond-shaped group of draw points in each period (Deiring (2006)). The mining engineers favour a cave with a diamond shape as it has the desired area to perimeter ratio to have good rock behaviour. Ideally a square cave is favoured over a long and thin one.

Diering (2006) defines a diamond as "On a given front [of the cave], the line of draw points is relatively straight". We define a diamond as a shape with 4 vertices such that the diagonals intersect at right angles. The formulations that follow are based on this definition.

4 Vertices

In the 4 Vertices model the general idea is that a diamond is defined by its 4 vertices which induce its 4 edges and interior points. The points in a diamond can be described by the number of open neighbours. In the mine layout a typical draw point has two within-tunnel neighbours and one across-tunnel neighbour. Draw points at the edges of the footprint may only have a subset of these three neighbours. Clearly a point in the interior of the diamond has all three neighbours open. Points on an edge have two neighbours open and a vertex has either one or two open neighbours.



Diamond Example 1 **Diamond Example 2**

Figure 3.29: Example of Diamonds with Vertices Having Only One Open Neighbour

Figure 3.29 shows examples of diamonds in which the vertices have only one neighbour. The zig zag lines represent the tunnels and the draw points occur at the bends in the zig-zag. The vertical lines connect the draw points to their across-tunnel neighbours. The vertices are marked with asterisks, draw points with two open neighbours are marked with open circles, and those with three open neighbours are marked with open squares.

With this picture in mind, an attempt was made to write a formulation to open draw points if they either have at least two open neighbours or are one of the four vertices. The size of the diamond is constrained by the *Global Capacity* constraint, and each draw point can only be opened once.

There were difficulties in writing the formulation for the constraint to achieve that a draw point requires two neighbours open, or be one of the 4 vertices in order to open. The attempts quickly identified two problems with this model concept.

1. The vertex definition of having only one open neighbour does not hold for all possible diamonds.

2. Interior points are not forced to have all three neighbours open and holes can form within the diamond

The first problem is illustrated in Figure 3.30. The draw point in the top vertex position has 2 open neighbours and so can open without being designated as a vertex. As a result, another draw point, not in this diamond, can be assigned the vertex and initiate another diamond, not connected to the one shown.

The second problem arises from the constraint that a draw point must have at least two open neighbours to open. Ideally there would be one constraint for edges, allowing only two open neighbours, and one for interior draw points requiring all three. We were not able to write a set of constraints that differentiated between edge and interior points. Instead, a more restrictive definition of the diamond was sought.

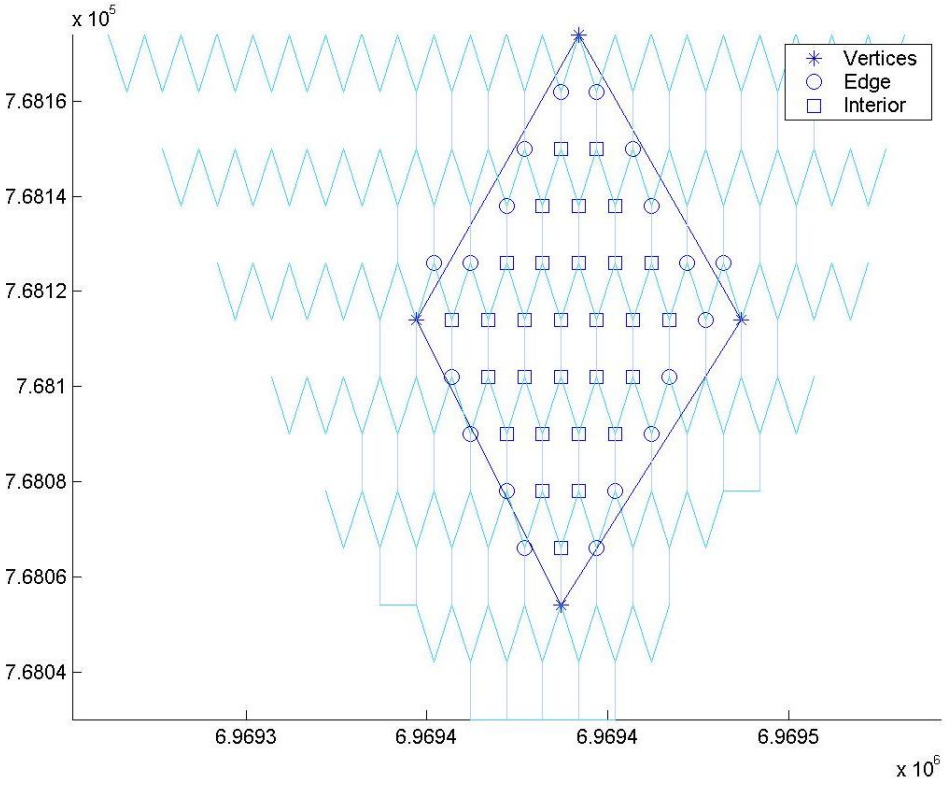


Figure 3.30: Example of Diamond with a (Top) Vertex Having Two Open Neighbours

2 Vertices and Cones

The 2 Vertices and Cones formulation is an adaptation of the 4 Vertices formulation to specifically address the target of one contiguous cave. In this formulation the diamond is identified by a "top" vertex (Vertex1), a "bottom" vertex (Vertex4), and the intersection of a upward-pointing cone defined by Vertex1, and an downward-pointing cone defined by Vertex4.

If a draw point is Vertex1 in period t then the draw points in the cone above it are forced open to form BottomCone for period t. Similarly if a draw point is Vertex4 in period t then the draw points in the cone below it are forced open to form TopCone for period t. The draw points in the intersection of the two cones must either have opened in an earlier period $i < t$ or in that period t.

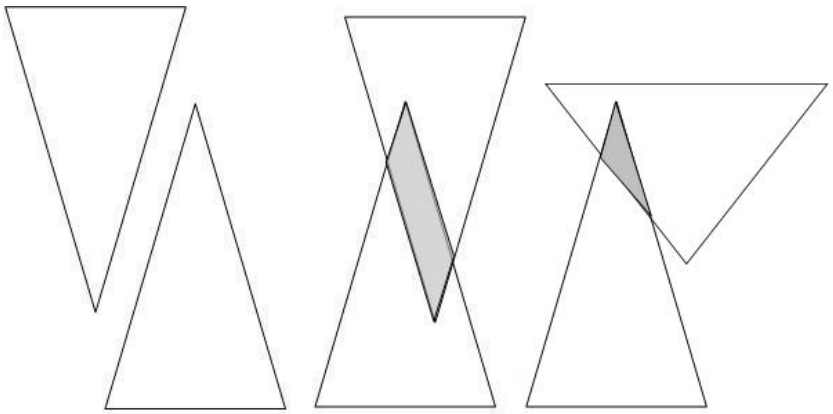


Figure 3.31: Three Examples of Pairs of Cones and Their Intersections. Only when a vertex from each cone are in both cones (middle case) does a diamond result

The intersections of the two cones will only give the desired diamond shape if both vertices are in both cones, see Figure 3.31. A constraint is desired that will ensure that both vertices are in both cones.

The definition of the cones defines the possible shapes that can occur. More specifically, the definition of the cones fixes the interior angles at each vertex.

Two attempts were made at this model. In the first, unsuccessful, attempt, constraints were written to define which draw points could be included in a cone, but allow the model to determine the actual shape of the cones during execution. The cone definition began with the corresponding

vertex, then neighbours adjacent to the vertex could be included, then their neighbours etc. Despite all attempts, the formulations either generated multiple caves in a single period or over multiple periods, and proved to have many of the problems of the 4 Vertices model; the results are not presented here.

The second attempt was based on a much more rigid definition of the cones. In this 2Cone model there is a defined cone that is forced open for each vertex. Based on discussions with Gemcom (Deiring (2006)), we proceeded with cones that grow by two draw points per incremental tunnel. If the intertunnel spacing equals the inter draw point spacing this will result in a base angle of approximately 45 degrees.

2Cone Formulation

If the vertices are limited to the footprint of the mine, then it will never be possible to include all the draw points in the defined cone and open them. For example, if the top vertex is in the top tunnel at draw point dpt, then at most its two within-tunnel neighbours $dpt-1$ and $dpt+1$ will be in the cone, and $dpt-2$ can not open unless the vertex moves to $dpt-1$ in the next period. To overcome this difficulty, the space for the vertices is extended beyond the mine footprint with extra tunnels. In general, enough tunnels should be added below the first tunnel and above the last tunnel so that the maximum tunnel length in the footprint can be in a cone if it occurred any tunnel. An extended footprint is created by adding $eTNLS$ tunnels above and below the mine footprint. For a cave that grows at two draw points per tunnel, the $eTNLS$ is one half the longest tunnel length. An example is shown in Figure 3.32.

Data : $nTNLS, T, m, p_{tn,dpt}, w_{tn,dpt}, \lambda$

eTNLS the number of tunnels added above and below the mine footprint to create the extended footprint. For a cave that grows by two draw points per tunnel, the value is half of largest across-tunnel value for all tunnels, $eTNLS = \lceil 0.5 * \max_{tn}(Ldpt_{tn}) \rceil$.

Decision Variables :

$\mathbf{S}_{tn,dpt,t}$ binary variable that is 1 in the period the draw point is in both cones in a given period. Note the definition of this variable is altered slightly for this model only.

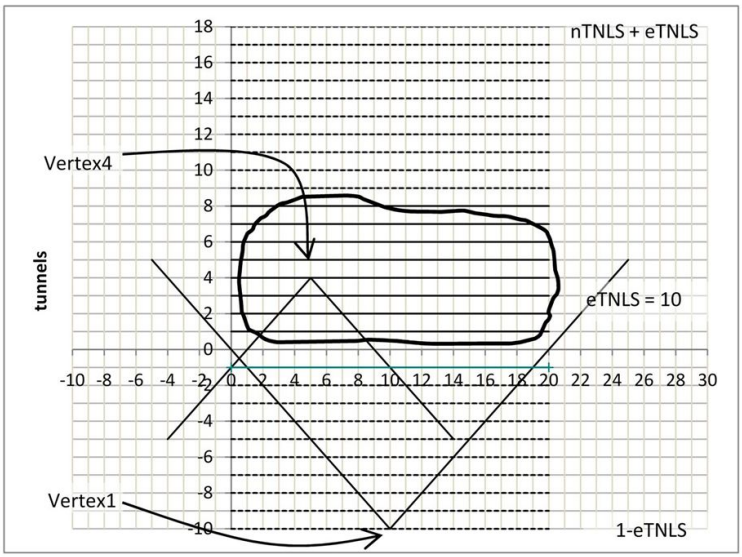


Figure 3.32: An Example of the Extended Footprint for the 2Cone Model. The outline of the mine footprint is shown, and the 8 tunnels (horizontal lines) shown in solid lines. Since the longest tunnel is 20 draw points long, $eTNLS = 10$ and 10 additional tunnels above and below the mine footprint are shown in dashed lines. Two example cones are shown, one originating at a vertex in the extended footprint.

Vertex1$_{tn,dpt,t}$ binary variable that is 1 if the draw point is the bottom (lowest tunnel) vertex in a given period. This variable is defined over the extended footprint $1 \leq tn \leq nTNLS + eTNLS$, $1 \leq dpt \leq 2 * eTNLS$

Vertex4$_{tn,dpt,t}$ binary variable that is 1 if the draw point is the top (highest tunnel) vertex in a given period. This variable is defined over the extended footprint $1 + eTNLS \leq tn \leq nTNLS + 2 * eTNLS$, $1 \leq dpt \leq 2 * eTNLS$

BottomCone$_{tn,dpt,t}$ binary variable that is 1 if the draw point is the the cone projecting up from Vertex1 in a given period

TopCone$_{tn,dpt,t}$ binary variable that is 1 if the draw point is in the the cone projecting down from Vertex4 in a given period

Total Number of (binary) Decision Variables: $nDPTS \times T \times 5$

Objective :

$$\max Value = \sum_{tn=1}^{nTNLS} \sum_{dpt=Fdpt_{tn}}^{Ldpt_{tn}} w_{tn,dpt} \sum_{t} S_{tn,dpt,t} e^{-\lambda(t-1)}$$

Constraints :

Start Once (3.11)

Global Capacity (3.12)

One Vertex4 : in each period

$$\sum_{tn=1+eTNLS}^{nTNLS+2*eTNLS} \sum_{dpt=1}^{2*eTNLS} Vertex4_{tn,dpt,t} \leq 1 \; \forall \; t \tag{3.31}$$

One Vertex1 : in each period

$$\sum_{tn=1}^{nTNLS+eTNLS} \sum_{dpt=1}^{2*eTNLS} Vertex1_{tn,dpt,t} \leq 1 \; \forall \; t \tag{3.32}$$

Define Top Cone : Starting at Vertex4, the cone grows by two draw points per tunnel.

$$TopCone_{tn,dpt,t} = \sum_{tun=tn}^{nTNLS+eTNLS} \sum_{d=dpt-(tun-tn)}^{dpt+(tun-tn)} Vertex4_{tun+eTNLS,d,t} \quad \forall \; tn, \forall \; dpt, \forall \; t \tag{3.33}$$

Define Bottom Cone

$$BottomCone_{tn,dpt,t} = \sum_{tun=1-eTNLS}^{tn} \sum_{d=dpt-(tn-tun)}^{dpt+(tn-tun)} Vertex1_{tun+eTNLS,d,t} \quad \forall \; tn, \forall \; dpt, \forall \; t \tag{3.34}$$

Open Intersection of Cones draw points in both cones must be in the pattern

In Both

$$\sum_{u=1}^{t-1} S_{tn,dpt,u} \geq TopCone_{tn,dpt,t} + BottomCone_{tn,dpt,t} - 1 \; \forall \; tn, \forall \; dpt, \forall \; t \tag{3.35}$$

In Top

$$S_{tn,dpt,t} \leq TopCone_{tn,dpt,t} \; \forall \; tn, \forall \; dpt, \forall \; t \tag{3.36}$$

In Bottom

$$S_{tn,dpt,t} \leq BottomCone_{tn,dpt,t} \; \forall \; tn, \forall \; dpt, \forall \; t \tag{3.37}$$

Previous Cave in Current Cave

$$\sum_{u=1}^{t-1} S_{tn,dpt,u} \leq BottomCone_{tn,dpt,t} \; \forall \; tn, \forall \; dpt, \forall \; t \geq 2 \tag{3.38}$$

$$\sum_{u=1}^{t-1} S_{tn,dpt,u} \leq TopCone_{tn,dpt,t} \; \forall \; tn, \forall \; dpt, \forall \; t \geq 2 \tag{3.39}$$

Total Number of Constraints : $nDPTS + T \times 3 + nDPTS \times T \times 7 - nDPTS \times 2$

Results

The 2Cone model was successfully run on data set P2. The model ran in 22 seconds and returned a single cave in each period with a total NPV of 356,187. This model significantly faster than the Malkin model, which took 89 seconds on the maximum value run alone. The value of the 2Cone result was approx 79% of value of the maximum value Malkin run. The initial cave was centred in the middle of the footprint, on tunnels 6 and 7, close to the centre point for the maximum value Malkin run.

In Section 3.5 the performance of this model is compared to others we developed for this problem on two trial data sets.

Contribution

The 2Cone formulation presented in this section is a new idea I developed for this thesis work. The formulation and implementation is new work on the problem of Sequence Optimization and is part of the contribution of this thesis to the research literature.

3.4.6 Column Generation

Since other approaches did not seem to guarantee a single contiguous cave that could move freely over the mining footprint, a completely different approach was proposed by Maurice Queyranne. This approach uses the well known technique called column generation Wolsey (1998). Column generation breaks the problem down into a Master problem which selects a solution from a set of presented options, and a Subproblem that generates additional feasible options that will improve the objective function. The presented options are denoted as 'columns' and the Subproblem generates more 'columns' for the Master problem, hence the name column generation. The appeal of this technique for solving Sequence Optimization is that while the Master problem must be run as a linear/integer program, the Subproblem is not limited to the linear/integer programming structure. Note that these "columns" are not the same columns used to describe the mining footprint.

Clearly, if we could enumerate all the possible opening patterns at the start of the procedure the Master problem would be sufficient and the problem solved in one pass. Since each opening pattern is already an acceptable, single contiguous cave, the constraints in the Master problem would guarantee the other requirements such as *Start Once* Constraint, *Global Capacity* Constraint, and any others such as the lead/lag time of adjacent draw point openings. However, there are far too many possible patterns for this to work.

ColGen Model Formulation

The Master problem will contain a variable that indicates which columns are chosen. While this will be a binary integer variable, the linear program relaxation of the Master problem is solved in order to generate dual values/shadow prices for each of the constraints. These values are used by the Subproblem to generate a new column that is not only feasible, but also that will improve the objective function of the Master problem. After convergence is reached, the Master problem is run as an integer program to determine the final solution.

Column generation has been successfully used in large scheduling problems such as course registration at colleges and airline scheduling (Lübbecke (2005)). In these settings the constraints on feasible schedules are complex, and there are a large number of feasible schedules, which is similar to our problem.

The overall objective is to determine in which period to start each draw

point in order to maximize total NPV subject to a global capacity constraint and shape constraints on the open mine in each period.

Decision Variables :

$\mathbf{S}_{tn,dpt,t}$ binary variable that is 1 in the period the draw point is opened.

Constraints :

Start Once (3.11)

Global Capacity (3.12)

Open Mine shape The open draw points in each period must form a contiguous shape that is roughly a diamond.

Master Problem

Our motivation in applying the column generation method is to remove the *Open Mine Shape* constraint from the rest of the problem as it has proven difficult to formulate in the linear and integer programming setting.

The Master problem will choose one mine opening pattern for each period $t \leq T$ to form a feasible opening sequence that maximizes NPV.

We construct the Master problem as

Data : $nTNLS, Fdpt_{tn}, Ldpt_{tn}, T, m, p_{tn,dpt}, w_{tn,dpt}, \lambda$

$\mathbf{K}_t$ the number of feasible mine opening patterns for period t, $1 \leq t \leq T$

$\mathbf{L}_{tn,dpt,t,k}$ a binary array of feasible mine opening patterns (tn, dpt) for period t, $1 \leq tn \leq nTNLS$, $dpt = Fdpt_{tn} \ldots Ldpt_{tn}$, $1 \leq k \leq K_t$, $1 \leq t \leq T$. These patterns are created in the Subproblem and are included as data in the Master problem.

Decision Variables :$S_{tn,dpt,t}$

$\mathbf{X}_{t,j}$ binary variable that is 1 if mine opening pattern $L_{*,*,t,j}$ is chosen in period t, $1 \leq t \leq T$

Objective :

$$\max Value = \sum_{tn=1}^{nTNLS} \sum_{dpt=Fdpt_{tn}}^{Ldpt_{tn}} w_{tn,dpt} \sum_{t} S_{tn,dpt,t} e^{-\lambda(t-1)}$$

Constraints :

Start Once (3.11)

Global Capacity (3.12)

Link Selected Mines Together: First period

$$S_{tn,dpt,1} = \sum_{k=1}^{allpatterns} L_{tn,dpt,1,k} X_{1,k} \quad \forall\, tn, \forall\, dpt \tag{3.40}$$

Link Selected Mines Together: All other periods

$$S_{tn,dpt,t} = \sum_{k=1}^{allpatterns} L_{tn,dpt,t,k} X_{t,k} - \sum_{k=1}^{allpatterns} L_{tn,dpt,t-1,k} X_{t-1,k} \quad \forall\, tn, \forall\, dpt, \forall\, t > 1 \tag{3.41}$$

Select At Most One Pattern Per Period

$$\sum_{j=1}^{allpatterns} X_{t,j} \leq 1 \;\forall\, t \tag{3.42}$$

Improving Objective Function

When the Master problem is run as a linear program, a shadow price will be generated for each constraint. In choosing new patterns for each period to add to L (the role of the Subproblem), the Master objective function will be improved if the reduced cost of the added patterns improves the objective. In this case, the objective is

$$\max Value = \sum_{tn=1}^{nTNLS} \sum_{dpt=Fdpt_{tn}}^{Ldpt_{tn}} w_{tn,dpt} \sum_{t} S_{tn,dpt,t} e^{-\lambda(t-1)}$$

for the given discount rate λ. For each period t, we seek a new pattern such that the reduced cost is greater than zero. The reduced cost is a function of the objective coefficient of X for period t, in this case zero, and the shadow prices of the constraints on X for period t.

Let us assign the following dual variables to the constraints of interest.

Link Selected Mines Together: First Period (3.40)

dual variable $\pi_{tn,dpt,1} \ \forall\, tn, \forall\, dpt$

Link Selected Mines Together: All Other Periods (3.41)

dual variable $\pi_{tn,dpt,t} \ \forall\, t > 1, \forall\, tn, \forall\, dpt$

Select At Most One Pattern Per Period (3.42)

dual variable $\mu_t \ \forall\, t$

The dual formulation related to the X variables becomes

$$X_{t,k}: \ -\sum_{tn}\sum_{dpt} L_{tn,dpt,t,k}\pi_{tn,dpt,t} + \sum_{tn}\sum_{dpt} L_{tn,dpt,t,k}\pi_{tn,dpt,t+1} + \mu_1 \ \ \forall\, t < T$$

$$X_{T,k}: \ -\sum_{tn}\sum_{dpt} L_{tn,dpt,T,k}\pi_{tn,dpt,T} + \mu_T$$

Simplifying, the dual formulation related to the X variables becomes

$$X_{t,k}: \ -\sum_{tn}\sum_{dpt} L_{tn,dpt,t,k}(\pi_{tn,dpt,t} - \pi_{tn,dpt,t+1}) + \mu_t \ \ \forall\, t \leq T$$

$$X_{T,k}: \ -\sum_{tn}\sum_{dpt} L_{tn,dpt,T,k}\pi_{tn,dpt,T} + \mu_T$$

An optimal solution to the (relaxed) Master problem is attained if all possible columns have negative reduced costs, that is, there is no way to generate a new column with positive reduced cost. Thus the Subproblem is to generate an opening pattern with positive reduced costs for each period to be added to the columns X of the Master problem.

In each period t, we are looking to add a new pattern $L_{tn,dpt,t,i}$ such that

$$-\sum_{tn}\sum_{dpt} L_{tn,dpt,t,i}(\pi_{tn,dpt,t} - \pi_{tn,dpt,t+1}) - \mu_t \geq 0$$

Equivalently, we are looking for an "acceptable" $L_{tn,dpt,t,i}$ for each period satisfying

$$\sum_{tn}\sum_{dpt} L_{tn,dpt,t,i}(\pi_{tn,dpt,t+1} - \pi_{tn,dpt,t}) - \mu_t \leq 0$$

Subproblem

The Subproblem is to generate new "acceptable" patterns that satisfy

$$-\sum_{\{tn,dpt\}\in pattern}(\pi_{tn,dpt,t}-\pi_{tn,dpt,t+1})-\mu_t\geq 0$$

for each period, if they exist.

Two Subproblem routines were developed, both based on the two intersecting cone concept. The first is an integer program formulation, and the second is an enumeration routine.

It was hoped that since the *Global Capacity Constraint* was included in the Master problem the dual values would transfer over information on the size of the pattern.

The Subproblem is run for each period $t \leq T$.

Data : $nTNLS, Fdpt_{tn}, Ldpt_{tn}, eTNLS$

Dual$_{tn,dpt}$ the dual value of each draw point for the chosen period t, $Dual_{tn,dpt} = \pi_{tn,dpt,t} - \pi_{tn,dpt,t+1}$, $1 \leq tn \leq nTNLS$, $Fdpt_{tn} \leq dpt \leq Ldpt_{tn}$

The integer programming formulation of the Subproblem for a single period is based on constructing cones with a fixed angle at the vertex. The cone starts with the vertex then grows by two draw points per tunnel.

Decision Variables :

Pat$_{tn,dpt}$ binary variable that is 1 if draw point is the pattern

Vertex1$_{tn,dpt}$ binary variable that is 1 if the draw point is the bottom (lowest tunnel) vertex in a given period. This variable is defined over the extended footprint $1 + eTNLS \leq tn \leq nTNLS + 2 * eTNLS$, $1 \leq dpt \leq 2 * eTNLS$.

Vertex4$_{tn,dpt}$ binary variable that is 1 if the draw point is the top (highest tunnel) vertex in a given period. This variable is defined over the extended footprint $1 + eTNLS \leq tn \leq nTNLS + 2 * eTNLS$, $1 \leq dpt \leq 2 * eTNLS$

BottomCone$_{tn,dpt}$ binary variable that is 1 if the draw point is in the cone projecting up from Vertex1 in a given period

TopCone$_{tn,dpt}$ binary variable that is 1 if the draw point is in the cone projecting down from Vertex4 in a given period

Objective :

$$\max Value = \sum_{tn=1}^{nTNLS} \sum_{dpt=Fdpt_{tn}}^{Ldpt_{tn}} Pat_{tn,dpt} Dual_{tn,dpt}$$

Constraints :

One Vertex4

$$\sum_{tn=1+eTNLS}^{nTNLS+2*eTNLS} \sum_{dpt=1}^{2*eTNLS} Vertex4_{tn,dpt} \leq 1 \; \forall \; tn, \forall \; dpt \quad (3.43)$$

One Vertex1

$$\sum_{tn=1}^{nTNLS+eTNLS} \sum_{dpt=1}^{2*eTNLS} Vertex1_{tn,dpt} \leq 1 \; \forall \; tn, \forall \; dpt \quad (3.44)$$

Define TopCone Starting at the vertex, the cone grows by two draw points per tunnel.

$$TopCone_{tn,dpt} = \sum_{tun=tn}^{nTNLS+eTNLS} \sum_{d=dpt-(tun-tn)}^{dpt+(tun-tn)} Vertex4_{tun+eTNLS,d} \quad \forall \; tn, \forall \; dpt \quad (3.45)$$

Define BottomCone

$$BottomCone_{tn,dpt} = \sum_{tun=1-eTNLS}^{tn} \sum_{d=dpt-(tn-tun)}^{dpt+(tn-tun)} Vertex1_{tun+eTNLS,d} \quad \forall \; tn, \forall \; dpt \quad (3.46)$$

Open Intersection of Cones draw points in both cones must be in pattern

In Both

$$Pat_{tn,dpt} \geq TopCone_{tn,dpt} + BottomCone_{tn,dpt} - 1 \; \forall \; tn, \forall \; dpt \quad (3.47)$$

In Top

$$Pat_{tn,dpt} \leq TopCone_{tn,dpt} \; \forall \; tn, \forall \; dpt \tag{3.48}$$

In Bottom

$$Pat_{tn,dpt} \leq BottomCone_{tn,dpt} \; \forall \; tn, \forall \; dpt \tag{3.49}$$

The enumeration routine of the Subproblem for a single period is also based on the two cone idea. The algorithm can be summarized as follows: For each possible pair of Top Vertex, Bottom Vertex draw points

- Determine the mine opening pattern for the vertex pair
- Determine the size (number of draw points), and value of the mine opening
- If the current pattern has a higher value than the previous contender continue, else go to next vertex pair
- If the current pattern is the correct size, the current pattern replaces the contender
- Go to the next vertex pair

The mine opening pattern is determined by the vertex pair and, as above, starts at the top vertex and grows by two draw points per tunnel until the mid point, then reduces by two draw points per tunnel until the bottom vertex is reached. If we denote tunnel and across coordinates of the top and bottom vertices as (T_{tn}, T_{dpt}) and (B_{tn}, B_{dpt}), for each tunnel t, $T_{tn} \leq t \leq B_{tn}$, the mine opening consists of draw points $\{\max(T_{dpt} - (t - T_{tn}), B_{dpt} - (B_{tn} - t)), \ldots, \min(T_{dpt} + (t - T_{tn}), B_{dpt} + (B_{tn} - t))\}$.

Initial Results

The first run was done using the data set P2, size 236 draw points. This data set was chosen both because it is small and because earlier work has shown that this data set tends to violate the one contiguous mine constraint. A value of 20 was chosen for m in the Master problem and the model run for 10 periods. The duration of the draw points range from 2 to 8 periods with very few of duration less than 4.

When the iterations converged, and the master model was finally run as an integer program, there was a huge gap between the IP and LP objective values. Inspection of the generated set of patterns showed some patterns

smaller than the maximum active size m, and some that were clearly too large, i.e., larger than m in period 1. Neither of these groups of patterns were truly feasible patterns. Closer inspection of the intermediate results of the Master problem showed very fractional solutions that pieced together a collection of patterns to maximize the LP relaxation of the problem. Clearly the *Global Capacity* constraint did not transfer enough information from the Master problem to the Subproblem and it was concluded that both a minimum and maximum size constraint were needed for the Subproblem.

Determining Limits on Subproblem Mine Size

For period 1 the mine should be of size m. For all other periods, the size is a function of the activity in the previous periods and it is not possible to determine exact size unless all draw points have the same duration.

Lower size limits for each period can be determined by relaxing the *Open Mine Shape* assumption and instead assuming the draw points will open in order of increasing duration. Number the draw points d_i such that $p_{d_i} \leq p_{d_{i+1}}$. Start with the m shortest duration draw points, $\{d_1, \ldots, d_m\}$, in period 1, then open the next one, d_{m+1} in period $p_{d_1} + 1$ when the first active draw point closes. Continue to open draw points as the active draw points close, keeping track of the size of the mine in each period. This generates a lower bound on the number of open draw points or size of the mine in each period. Similarly, the upper size limits for each period can be determined by opening the draw points in order of decreasing duration. Since these estimation methods do not consider the feasibility the pattern size they are conservative and, if there is a wide spread of duration values, are not expected to be tight bounds.

Generating Sequences of Patterns

The preliminary runs also revealed that while the algorithm generated feasible mine patterns in a given period, it did not guarantee the generation of a feasible sequence of mine patterns over all periods. The linear relaxation of the model could use fractional solutions to incorporate the newly generated patterns and increase the objective value, but the final run as an integer program often reverted back to the initial set of columns to obtain a feasible solution.

We considered two solutions. In the first all mine patterns generated by the Subproblem can be selected by the Master problem for any period. This increases the number of feasible solutions as a mine pattern of size m could be

selected in all periods. In the second, a set of mine patterns is generated by the Subproblem for each period. This solution was strongly recommended by members of the thesis committee, so it was chosen. Since it is not possible to input the pattern of the previous period pattern to the Subproblem, there is no guarantee that the model will return a set of patterns that can be linked from period to period; there may be no feasible solution to the Master problem. To accommodate this, the definition of a feasible solution to the Subproblem is extended to include mine patterns in all other periods that together form a feasible sequence with the generated pattern. Once the Subproblem generates a mine pattern for period t, an algorithm generates a feasible pattern, or precursor for the period $t-1$ that is subset of the pattern t. The algorithm is repeated for each period $t-1 \ldots 2$ generating a sequence of feasible, linked patterns that could lead to the opening of the pattern generated by the Subproblem. If a feasible sequence for periods $1 \ldots t-1$ is generated, then another algorithm generates a feasible, linked sequence of patterns for periods $t+1 \ldots T$ and all the mine patterns are added to the Master problem. If no feasible sequence is generated for period $1 \ldots t-1$, then the pattern generated by the Subproblem is not added to the Master problem.

The algorithms for generating patterns for periods $1 \ldots t-1$, precursor patterns, and for period $t+1 \ldots T$, following patterns, use enumeration to try to maximize the dual value of each pattern. For the precursors, all pairs of vertices and projected cones are evaluated to see if they are contained in the current pattern, and meet the Global Capacity constraint. The pattern with the highest dual value is selected as the precursor, and the algorithm is repeated to find a precursor to this pattern. Similarly, for each of the following patterns, an estimate of the maximum size of the pattern is generated from the durations of the current pattern and all pairs of vertices and corresponding projected cones that cover the current pattern are evaluated to find the one with the maximum dual value.

Generating the Initial Columns

The column generation process begins by running the LP relaxation of the Master problem, which requires a feasible solution. A simple initial solution is the same set of m draw points, in a feasible shape, in each of the periods. As an alternative, the algorithms developed for mine sequence generation, discussed above, were adapted to generate a feasible sequence. First the Subproblem is run to generate the mine pattern for period one, then the algorithm is used to generate a sequence of following patterns.

Since dual values are not available for the Subproblem or following pattern algorithm, draw point values are substituted in all periods. We called this the greedy/myopic algorithm as it opens the adjacent draw points of highest value in each period. It is discussed below, in Section 3.4.7

Contribution

Column Generation is a well used technique in solving large problems, including parallel machine scheduling. In applying Column Generation to Sequence Optimization I developed the mine sequence algorithms necessary to produce a feasible sequence of caves. The algorithms and implementation are new work on the problem of Sequence Optimization and is part of the contribution of this thesis to the research literature.

3.4.7 Greedy/Myopic Algorithm

The Greedy/Myopic algorithm is a direct result of the work done on the ColGen model. In order to begin the ColGen algorithm and generate the first set of dual prices, a feasible set of columns must be included in the Master problem. For the first few attempts the initial set of columns was generated by hand. With the modification of the ColGen algorithm to generate mine openings that preceded and followed the mine opening generated by the Subproblem, the opportunity to generate a feasible initial set of columns presented itself.

The Greedy algorithm uses the ColGen Subproblem routine to generate a mine opening pattern for the first period, then uses the routine written to generate mine openings for the rest of the periods. Draw point weights are substituted for the reduced costs in the routines. As a result, the highest valuation mine opening is selected for each period resulting in a myopic or greedy algorithm. This is essentially the same method used to generate the initial columns for the ColGen algorithm.

Results

Since both the ColGen model and the Greedy algorithm use the same cave shape definition as the 2Cone model, the generated caves are similar to those from the 2Cone model. In Section 3.5 the performance of both the ColGen model and the Greedy algorithm on its own are compared to others we developed for this problem.

Contribution

The idea of a greedy algorithm using a myopic view is well established. In this work, I have created an algorithm that develops a feasible sequence of caves using a greedy approach. The algorithm and implementation is new work on the problem of Sequence Optimization and is part of the contribution of this thesis to the research literature.

3.5 Application of Models to Test Data Sets

This section describes the application of the developed models to two new data sets. The models were developed using the four previously described data sets, so there was a risk that the models were tailored to these data sets. Thus we sought out new data sets from Gemcom. They provided us with two data sets coming from a project with Rio Tinto, one of the worlds leading mining and exploration companies. Unfortunately this data is confidential and so is not publicly available. Evaluating the models on two new data sets was an attempt to objectively evaluate and compare the models. The objective was to evaluate the efficiency and practical applicability of the models to aid in the Sequence Optimization process. In addition, the execution speed of the models is compared. All models discussed in the previous sections were run and the solutions and time to solve were compared. Some additional runs were also conducted with constraints on the starting opening pattern constraints to answer additional planning considerations.

3.5.1 Data Sets

We now describe the two data sets. RT1 is a mine with a small footprint consisting of 332 draw points to be opened over 10 years. Planners believe that the small size of the footprint reduces the possible combination of sequences and makes it easier to determine a good Sequence Optimization by traditional methods. Running the new models on this data set allows us to demonstrate their potential on a data set not considered difficult to plan for using existing techniques. This mine is already under production which provided a basis of comparison for Rio Tinto. The mining sequence was published graphically (Calder (2000)) but without values so a value comparison is not possible.

By contrast, RT2 is a mine with a huge footprint of 6510 draw points to be opened over 20 periods. The size of the data set clearly presents challenges to any solution technique, and so it is a good test of the new models.

Data Set RT1

The data set RT1 consists of 332 draw points along 19 vertical tunnels, the longest of which is 30 draw points long; see Figure 3.33 below.

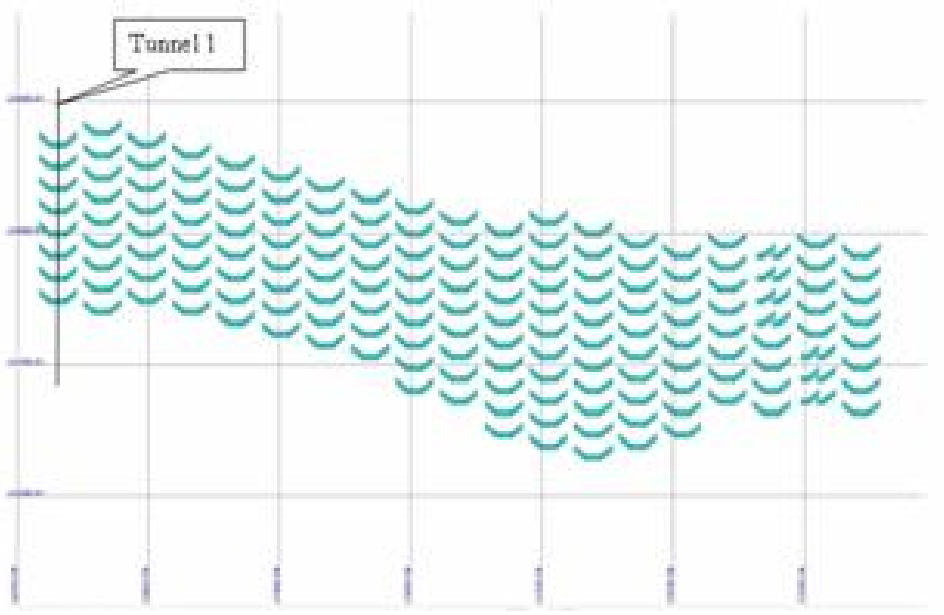


Figure 3.33: Data Set RT1

Provided Data:

Table 3.6 is a sample of the data available for each draw point as supplied by Gemcom:

Draw Point Name: A naming convention used by Gemcom to identify each draw point. The name consist of an "L" followed by a tunnel then either a "W" or an "E" followed by a draw point number

X, Y: are the location coordinates of the draw point

Tot_Dol: this is the total value of the column of ore above the draw point. It is assumed that this is already discounted to represent the net present value (NPV); the details behind the generation of this data were not provided to us as they were considered confidential by Gemcom.

Tonnage: the total tonnage (units) of the column of ore above the draw point.

Required Data:

Duration and Value: Duration and NPV value of each draw point is required for the models. The duration of each draw point is calculated by dividing the amount of ore to be mined by the assumed constant draw rate.

Draw Point Name	X	Y	Tot_Dol	Tonnage
L01W01	13348.78	-24016.7	11061.73	452,008
L01W02	13348.78	-24033.7	8689.515	432,104
L01W03	13348.78	-24050.7	6138.938	443,745
L01W04	13348.78	-24067.7	7823.849	444,517
L01W05	13348.78	-24084.7	17987.16	455,378
L01W06	13348.78	-24101.7	16093.53	439,121
⋮	⋮	⋮	⋮	⋮

Table 3.6: Available Data for Data Set RT1

In this project, the data in column Tonnage was divided by the assumed draw rate which was provided at 120 tons/day. textbf120 tons/day (120 tons/day *365 days/year = **43,800 tons/year**) was used. To determine the duration of each draw point the following formula was used

$$\text{Duration} = \text{round}(\text{Tonnage}/43{,}800 \text{ tons per year})$$

The round function converts the result to the nearest integer value (up or down). As a result, the durations ranged from 7 to 17 years and the distribution of durations is shown in Figure 3.34.

The total value of each draw point, discounted to the period it is opened, is also required as input to the models. The results in this thesis used the Tot_Dol column from the provided data for this input.

In this data set there are 37 draw points with negative value that must be opened to complete the mine. As can be seen in Figure 3.35, these draw points with negative value are grouped in tunnels 6, 7 and 8. The highest value draw points are in a cluster just two tunnels over.

Maximum active draw points: The downstream processing capacity is captured in the models with a maximum active draw points constraint for each period. The downstream processing capacity was provided in the form of an annual production rate which increases from 3 to 12.41 Mt/y over the first four years of operation. In order to determine the number of active draw points the annual production rate was converted into equivalent number of draw points. Table 3.7 shows the estimated number of draw points in each period, for the constant production rates supplied, 120 tons/day.

Discount rate: A discount rate of 10% was supplied by Gemcom

Other data: The coordinates of the draw points were converted into tunnel and draw point numbers consistent with the grid system used in the

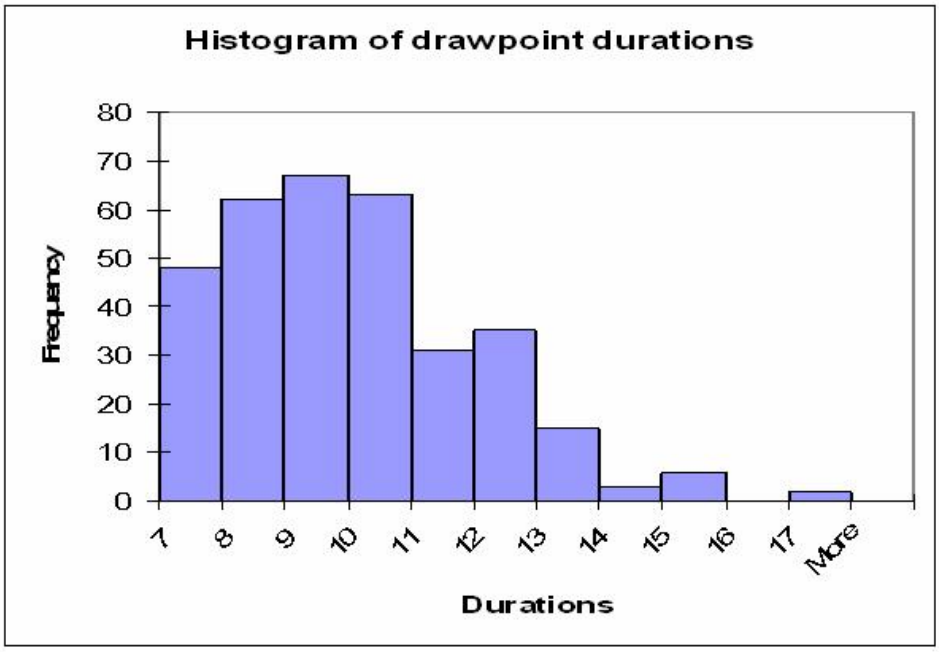


Figure 3.34: Histogram of Draw Point Durations at 120 tons/day for Data Set RT1

		Constant Draw Rate 120 tons/day
	Annual production rate Mt	Maximum active draw points
Year 1	3	70
Year 2	6	138
Year 3	9	206
Year 4	12.41	284
Subsequent years	12.41	284

Table 3.7: Maximum Active Draw Points for Data Set RT1

models.

In this data set the tunnels were parallel to the "Y" coordinate and the across-tunnel draw points were perpendicular to the tunnels. The data were

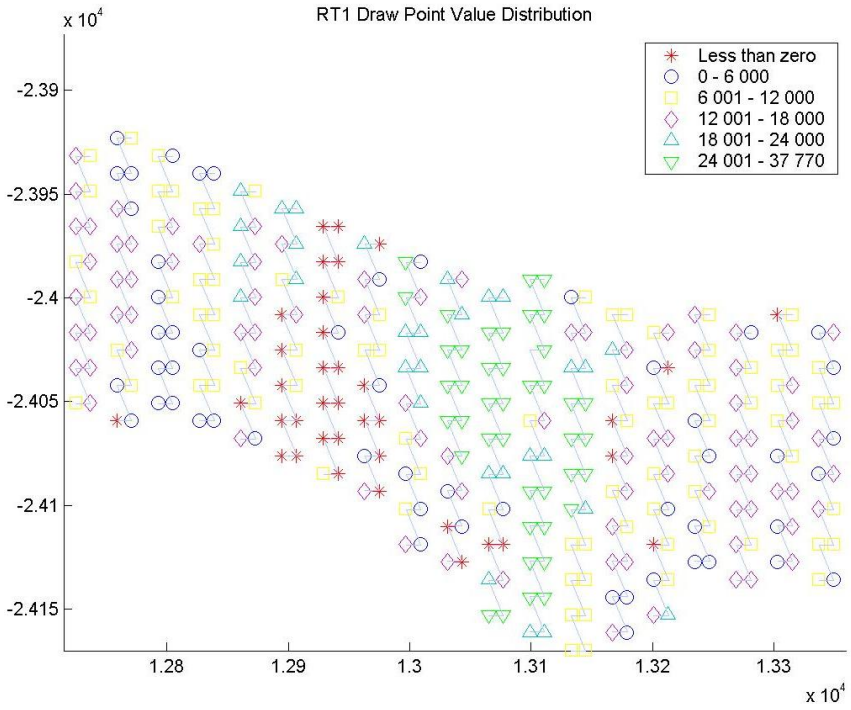


Figure 3.35: Distribution of Draw Point Values over the Mining Footprint for Data Set RT1

rotated through 90° to align the tunnels across the grid to be consistent with our prior usage. Under this transformation, the tunnel numbering is in the reverse order from the tunnel assigned by Gemcom, and tunnel 1 is a combination of L20E and L19W. The "W" draw points take the even numbering and the "E" become the odd. Thus L19W01 becomes tunnel 1, draw point 1 and L20E01 becomes tunnel 1, draw point 2. The Gemcom draw point numbering convention begins with 01 at the start of each tunnel, but due to the shape of the footprint, the ends of the tunnels are not aligned. The new draw point numberings correct for this.

Data Set RT2

The data set RT2 consists of 6510 draw points along 56 vertical tunnels, the longest of which is 196 draw points long; see Figure 3.36 below.

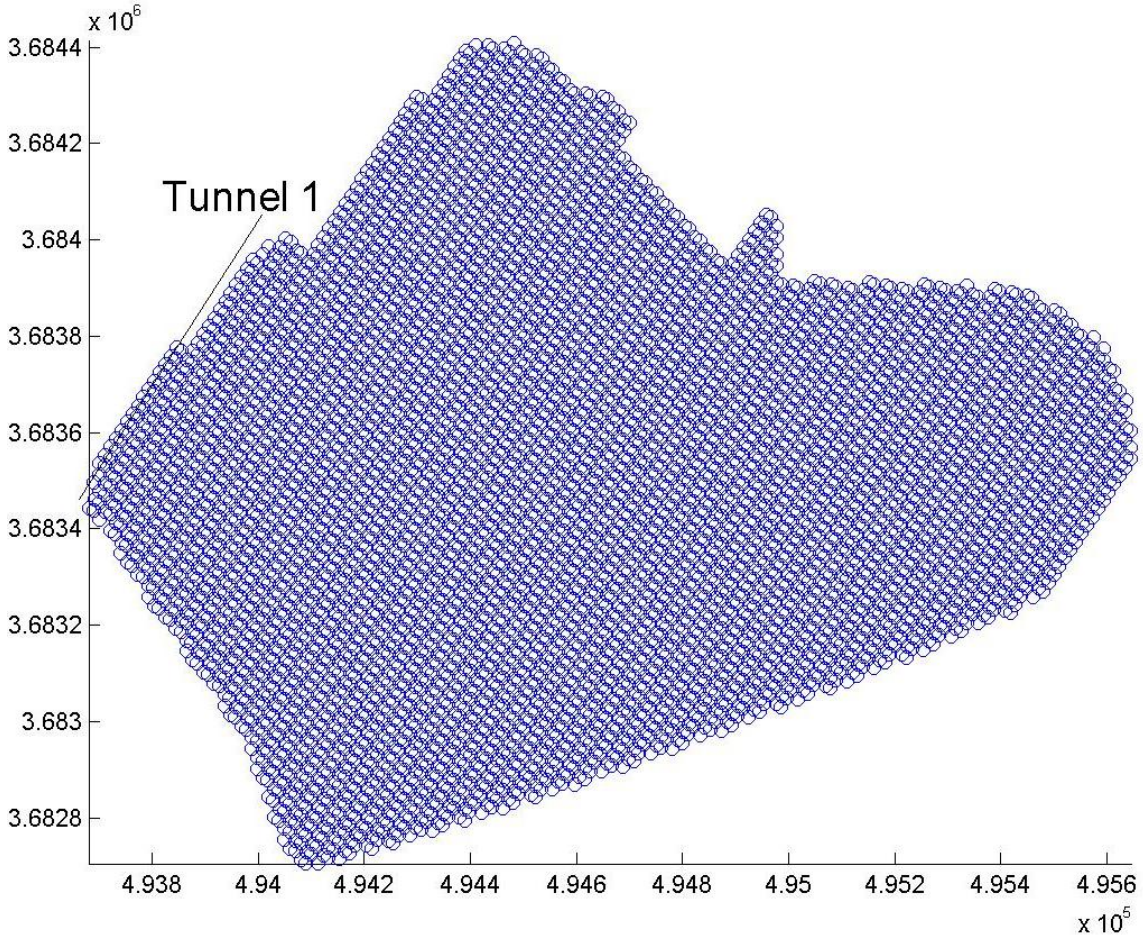


Figure 3.36: Data Set RT2

Provided Data:

Table 3.8 is a sample of the data available for each draw point as supplied by Gemcom:

DPTNAME	XCOORD	YCOORD	Tot_Dol	TON14M
P01_5621W	495460.3	3683259	-214.0677338	116,799
P01_5524W	495408.8	3683232	-200.9185181	75,699
P01_5524E	495417.9	3683226	-435.5866394	101,510
P01_5523W	495419.1	3683249	158.174881	74,210
P01_5523E	495428.3	3683243	10.73392391	86,200
⋮	⋮	⋮	⋮	⋮

Table 3.8: Available Data for Data Set RT2

DPT Name: A naming convention used by Gemcom to identify each draw point. The name begins with a letter and two digits followed by an underscore then 4 digits and either "W" or "E". The meaning of the segment before the underscore is not clear but could refer to a section of the mine, but it appears that the underscore is followed by a tunnel number and a draw point number. While the tunnel numbers appear to be consistent across sections, the draw point numbering is not.

XCOORD, YCOORD: are the location coordinates of the draw point

Tot_Dol: this is the total value of the column of ore above the draw point.

TON14M: the total tonnage (units) of the column of ore above the draw point.

Required Data:

Duration and Value: Duration and NPV value of each draw point is required for the models. The duration of each draw point is calculated by dividing the amount of ore to be mined by the assumed constant draw rate. In this project, the data in column TON14M was divided by the assumed draw rate of 270 tons/day. This value for the draw rate was chosen in conjunction with the maximum active draw points to ensure that the entire footprint could be mined in 20 periods.
270 tons/day (270 tons/day *365 days/year = **98,550 tons/year**) was used. To determine the duration of each draw point the following formula was used:

$$\text{Duration} = \text{round}(\text{TON14M}/98{,}550 \text{ tons per year})$$

As a result, the durations ranged from 1 to 5 years, with a single draw point at 5 years, and the distribution of durations is shown in Figure 3.37.

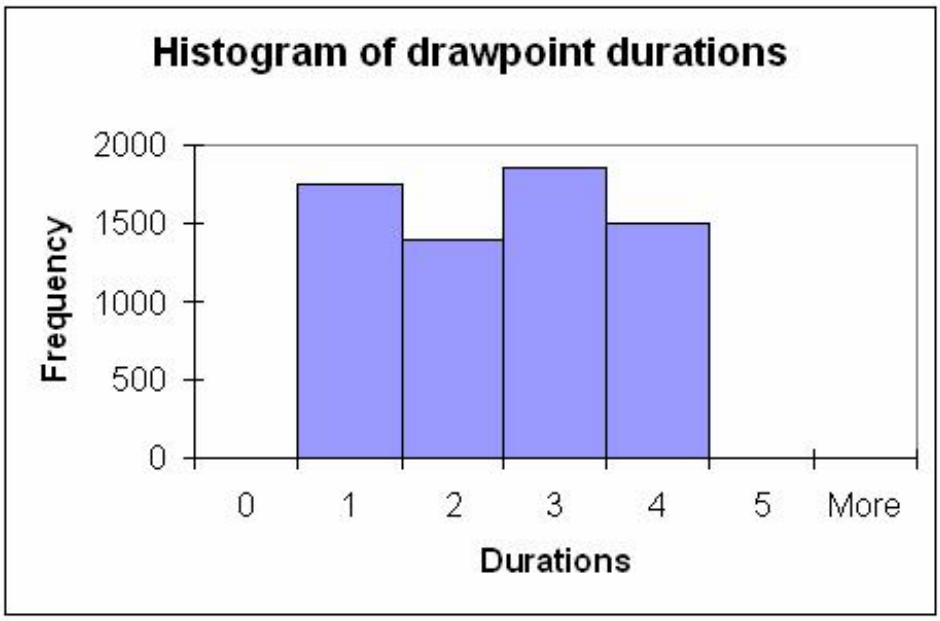


Figure 3.37: Histogram of Durations for Data Set RT2

The total value of each draw point, discounted to the period it is opened is also required as input to the models. The results in this thesis used the Tot_Dol column from the provided data and the calculated duration to calculate this input.

The total value is calculated using the PV(present value) function in Microsoft Excel. It is the PV of Duration installments of (Tot_Dol/Duration) payments at the discount rate.

Maximum active draw points: The downstream processing capacity is captured in the models with a maximum active draw points constraint for each period. The maximum active draw points values were chosen along with the daily draw rate to ensure that all draw points could be mined in 20 periods and are shown below in Table 3.9.

Discount rate: A discount rate of 10% was supplied by Gemcom

Other data: The coordinates of the draw points were converted into tunnel and draw point numbers consistent with the grid system used in the models.

In order to establish the grid, the data was projected onto a new coordinate set. The new axis R runs parallel to the tunnels and is obtained by rotating the original coordinates through 50°. This is the rotation required

		Constant Draw Rate 270 tons/day
	Annual production rate Mt	Maximum active draw points
Year 1	28,382,400	288
Year 2	54,793,800	566
Year 3	82,387,800	836
Year 4	109,587,600	1112
Subsequent years	109,587,600	1112

Table 3.9: Maximum Active Draw Points for Data Set RT2

to align the first tunnel along the horizontal axis. The new S axis runs in the across draw point axis and is perpendicular to the R axis. It is obtained by rotating the original coordinates through 60°. The procedure followed was as follows

1. Choose a point as the "origin", it will have new coordinates (0,0). Denote the original coordinates of this point as (X_o,Y_o)
2. calculate the distance from each draw point to the origin using the X and Y coordinates
3. calculate the new R coordinate as $distance \times \sin\{90 - 50 - \arcsin\{(Y - Y_o)/distance\}\}$
4. calculate the new S coordinate as $distance \times \sin\{\arcsin\{(Y - Y_o)/distance\} - 60\}$

3.5.2 Models

Five sequence optimization models were run on data sets RT1 and RT2. Four of the models were written in the linear/integer programming framework using the language AMPL and solved using CPLEX solver Version 9.1. The fifth model, a greedy algorithm, was written in Visual Basic. All five models used the same input data, and the same layout of draw points. The models were run on a Intel Pentium (R) D CPU 3.20 GHz machine with 1.99 GB of RAM.

Basic Model: This is the original "across and along" model, i.e., contiguity along and across each tunnel, described in Section 3.4.3. The constraints ensure that in each period, if a tunnel contains open or active draw points that they form a single contiguous segment. There are also constraints to enforce a similar contiguity constraint in the across-tunnel dimension. Recall from Section 3.4.3 that earlier runs have shown that enforcing contiguity in only two directions does not guarantee a single contiguous cave in each period.

Maximize: NPV

Subject to :

Start Once : (3.11)

Global Capacity : (3.12)

Contiguous along tunnels: (3.13) - (3.15)

Contiguous across tunnels: (3.16) - (3.19)

Table 3.10 shows the number of binary variables and constraints for the Basic model. In addition to the set of variables that indicate which draw points are open in each period, there are two sets of binary variables which denote the first open draw point along and across each tunnel. There are four contiguity constraints for each draw point in each period plus two for each tunnel in each period. These form the majority of the constraints for the model. For Data Set RT1 there are a total of almost 14 000 constraints, and this grows to almost 530 000 for Data Set RT2.

Malkin Model: This model is a formulation based on the ideas of Malkin and Wolsey, Malkin (2006). For a convex mine footprint, the model, described in Section 3.4.4, guarantees a single cave in each period by defining a fixed centre point to the cave and only allowing draw points to open if they are connected by open or active draw points to the centre point. A

		RT1	RT2
Binary Variables	$nDPTS \times T \times 3$	9 960	390 600
Constraints			
Start Once	$nDPTS$	332	6 510
Global Capacity	T	10	20
Contiguous Along	$nDPTS \times T \times 2 + nTNLS \times T$	6 970	261 520
Contiguous Across	$nDPTS \times T \times 2 + nTNLS \times T$	6 970	261 520
Total Constraints	$nDPTS + T +$ $nDPTS \times T \times 4 + nTNLS \times T \times 2$	13 922	529 570

Table 3.10: Number of Variables and Each Type of Constraints for Basic Model Runs on Data Set RT1 and RT2

fixed set of axes are constructed from the centre draw point to help with the formulation. The fixed centre point model was used as the increased data preparation for the "wandering axes" variation did not result in improved results in the developmental runs in Section 3.4.4.

Maximize: NPV

Subject to :

Start Once : (3.11)

Global Capacity : (3.12)

Column Connected: (3.24)-(3.25)

Row Connected: (3.26)-(3.29)

Table 3.11 shows that the Malkin model is considerably smaller than the Basic number, both in number of binary variables and number of constraints. The Malkin model has one third of the variables, and approximately one half of the constraints of the Basic model. For Data Set RT1 there are approximately 7 000 constraints, and approximately 130 000 constraints for Data Set RT2.

2Cone Model: The 2 vertices 2 cone model, described in Section 3.4.5, is an attempt to get a single cave that has a diamond shape. The cave is the intersection of two cones or triangles each determined by the decision variable representing the vertex of the cone. Figure 3.31 illustrates the concept.

		RT1	RT2
Binary Variables	$nDPTS \times T$	3 320	130 200
Constraints			
Start Once	$nDPTS$	332	6 510
Global Capacity	T	10	20
Column Connected	$nDPTS \times T$	3 320	130 200
Row Connected	$nDPTS \times T$	3 320	130 200
Total Constraints	$nDPTS + T + nDPTS \times T \times 2$	6 982	266 930

Table 3.11: Number of Variables and Each Type of Constraints for Malkin Model Runs on Data Set RT1 and RT2

Maximize: NPV

Subject to :

Start Once : (3.11)
Global Capacity : (3.12)
One top vertex in each period: (3.31)
One lower vertex in each period: (3.32)
Cone Definition : (3.33) - (3.34)
Open Intersection of Cones : (3.35) - (3.37)
Previous Cave Contained in Current Cave: (3.38) - (3.39)

The 2Cone model is the largest of the three integer programming models, as can be seen in Table 3.12. There are five sets of binary variables, including the two for the vertices over the extended footprint. This model has approximately 21 000 binary variables for Data Set RT1, and just over 1 000 000 for Data Set RT2. A large number of constraints are needed to define the cones and their intersection resulting in a model with approximately twice as many constraints as the Basic model.

ColGen: The column generation technique, described in Section 3.4.6, uses a LP model to choose from a set of cave patterns to form a maximum value, feasible sequence over all periods. A separate algorithm, not limited by the LP/IP framework, uses the dual prices generated in the LP to add new candidates to the set of cave patterns. In this case, the algorithm was based on the 2 Cone concept and found the highest value candidates by enumeration.

		RT1	RT2
Binary Variables	$nDPTS \times T \times 5 + eTNLS^2 \times T \times 2$	21 100	1 035 160
Constraints			
Start Once	$nDPTS$	332	6 510
Global Capacity	T	10	20
One of Each Vertex	$T \times 2$	20	40
Cone Definition	$nDPTS \times T \times 2$	6 640	260 400
Cone Intersection	$nDPTS \times T \times 3$	9 960	390 600
Previous Cone	$nDPTS \times (T-1) \times 2$	5 976	247 380
Total Constraints	$nDPTS + T \times 3 + nDPTS \times T \times 7 - nDPTS \times 2$	22 938	904 950

Table 3.12: Number of Variables and Each Type of Constraints for 2Cone Model Runs on Data Set RT1 and RT2

Greedy: The greedy model uses the cave generation algorithms from the ColGen model to generate an initial set of cave patterns that form a feasible sequence and is described in Section 3.4.7. Since there were no dual values available, the total value of the draw point is used.

3.5.3 Results

Results on Data Set RT1

Table 3.13 compares the total NPV and number of caves from running the models discussed above on Data Set RT1. As the Basic model has the most relaxed constraints, it is expected to give the highest NPV so will be used as a benchmark for the other models.

	Total NPV	# of Caves	% of max NPV
Basic	$ 3,507,820	>2	100
Malkin	$ 3,457,480	1	98.6
2Cone	$ 3,314,810	1	94.5
ColGen	$ 3,333,938	1	95
Greedy	$ 3,333,520	1	95

Table 3.13: Results for Data Set RT1

The highest NPV is from the Basic model, however inspection of the results shows that this model does not actually result in a single contiguous cave. Note that the Malkin result is (3,457,480/3,507,820 = 0.986) 98.6% of the Basic model result and the 2Cone result is (3,314,810/3,507,820 = 0.945) 95% of the Basic model result.

Somewhat surprising is the success of the Greedy model to achieve such a good result;(3,333,520/3,507,820 = 0.950) 95.0% of the Basic model result. Subsequent iterations of the ColGen model, from the Greedy initial solution, did result in improvement of the objective value, but it is almost negligible. (3,333,938/3,507,820 = 0.950) 95.0% of the Basic model result was achieved with ColGen model.

Table 3.14 reports the differences in active cave size over the ten periods for the different models. There is very little difference between the models with the exception of the Basic model. There are 37 draw points with negative value that must be mined for structural reasons. Some of the models may delay these until the last period and therefore discount this loss. This may explain why the Basic model only opens 132 of the possible 140 draw points in the second period.

As can be seen from the results in Table 3.15, the models took from 4 seconds to almost an hour to run. Listed are both the times for the complete set of Malkin models, with a run with each of the 332 draw points as the fixed centre point, and the single run that produced the highest value pattern. The Basic model ran far faster than the 2Cone and Malkin models,

	Number of draw points active in each period									
	1	2	3	4	5	6	7	8	9	10
Basic	70	132	205	284	284	284	284	284	283	254
Malkin	70	140	206	284	284	284	284	284	284	241
2Cone	70	140	198	284	284	284	284	283	283	223
ColGen	70	140	204	284	284	284	284	276	284	232
Greedy	70	140	205	284	284	284	284	279	279	232

Table 3.14: Cave Development Results for Data Set RT1

	Total NPV	Solution Time
Basic	$ 3,507,820	4 seconds
2Cone	$ 3,314,810	3 050 seconds ≈ 50 minutes
Malkin: all draw points		2 416 seconds ≈ 40 minutes
Malkin (maximum value)	$ 3,457,480	2 seconds

"Malkin: all draw points" is the aggregation of 332 runs of the Malkin model, once for each draw point as the fixed centre point.
'Malkin (maximum value)" is the single run that produced the highest value

Table 3.15: Solution Times:

however the resulting multiple caves are undesirable. In this case, the 2Cone model took 10 minutes longer to run than the Malkin model, but produced a higher NPV.

We now look at the mine shapes generated by solving these models on data set RT1. The figures show how well the results conform to the objective of a single cave as well as the preferred diamond shape. We also compare the progression of the caves over time to see how the results from the different models vary.

Basic:

The following sequence was determined using the Basic model. As has been found in the past, this model does not guarantee a single contiguous cave. The results show that there is a high value group of draw points on the right side that is chosen in the first period. It also shows that there is a valuable group on the left side that is also chosen in the first period, despite not being adjacent to the group on the right.

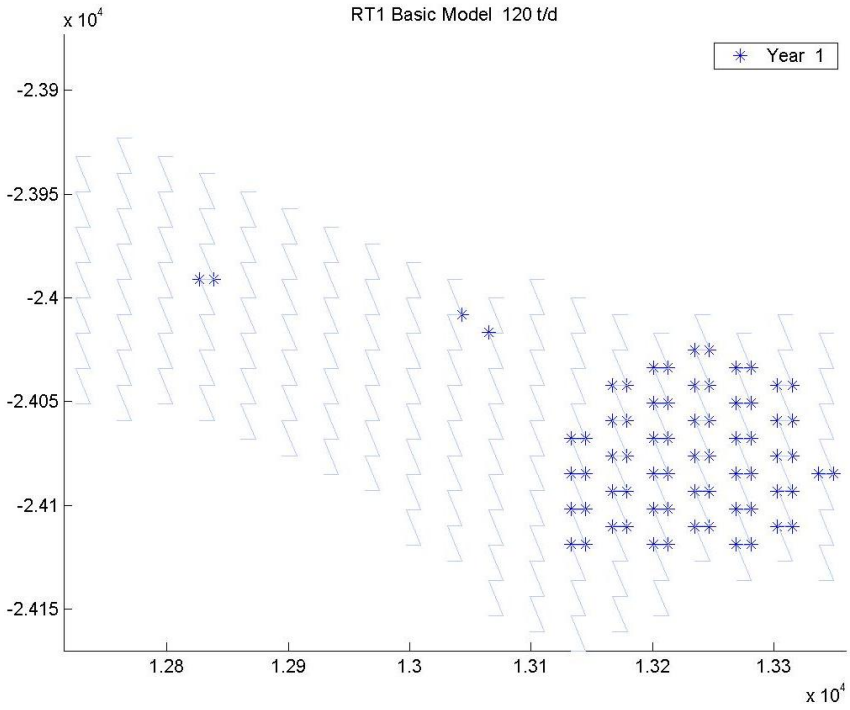


Figure 3.38: Results of Basic Model on Data Set RT1 - Period 1

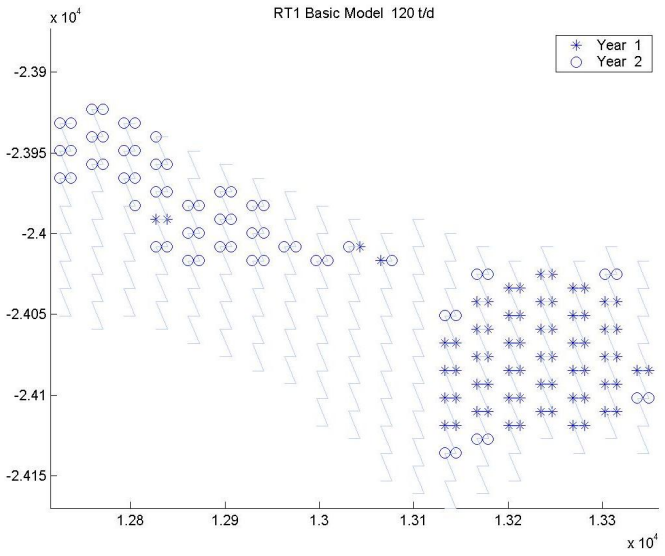


Figure 3.39: Results of Basic Model on Data Set RT1 - Period 2

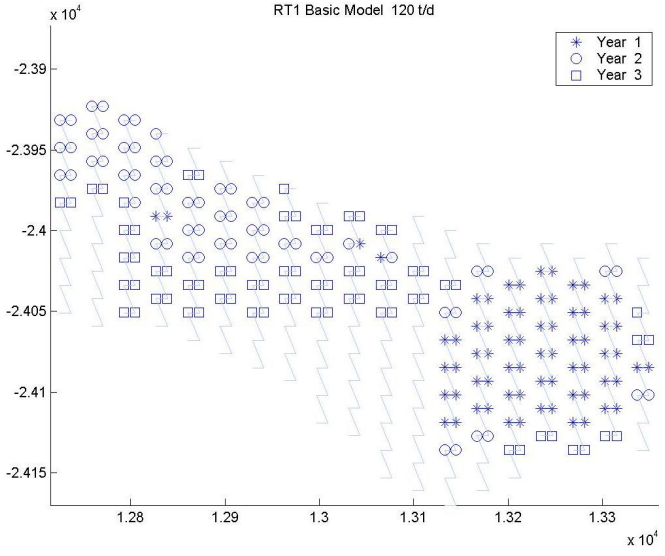


Figure 3.40: Results of Basic Model on Data Set RT1 - Period 3

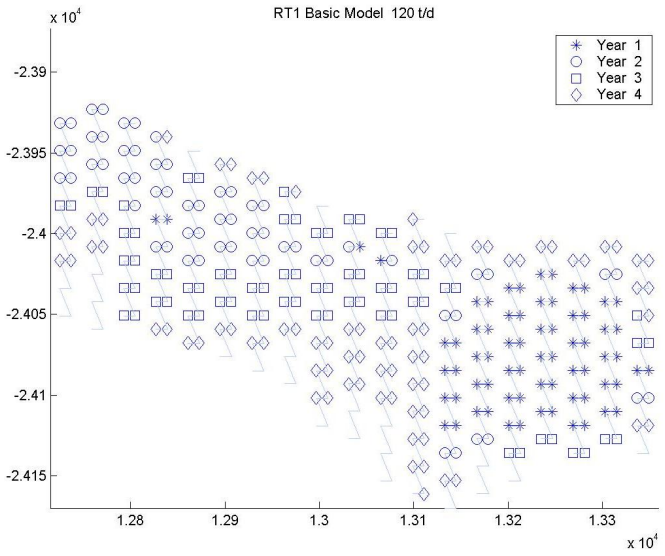


Figure 3.41: Results of Basic Model on Data Set RT1 - Period 4

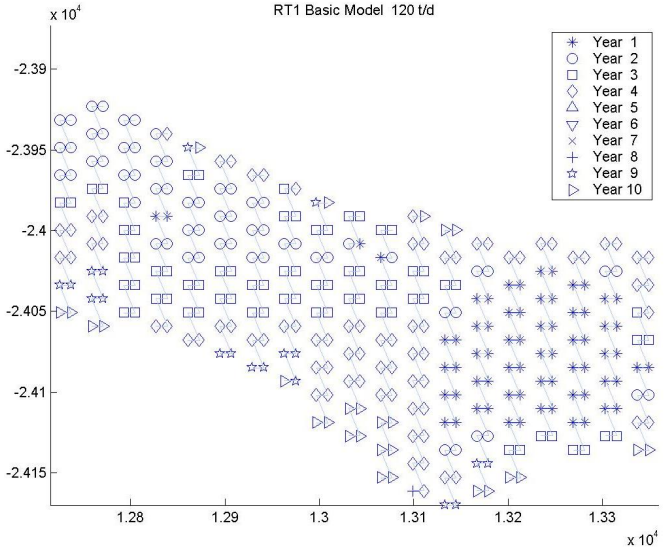


Figure 3.42: Results of Basic Model on Data Set RT1 - Period 10

Malkin:

The Malkin model was run repeatedly, changing the centre point over all draw points to find the centre point yielding the highest NPV. The run using this centre point was chosen for reporting and the pattern is shown below. As with the Basic model, in the first period draw points on right are chosen with a movement towards the left in the second period. The caves for years 2 and 3 extend along the across-tunnel axis and do not look diamond like. We can see that the constraints are met, in each period there are no gaps between any open draw point and either axis. In the first period, the cave is a diamond shape, but the axes of the diamond do not align with the designated axes. In the second and third period, the lack of symmetry around the axes results in a non-diamond shape.

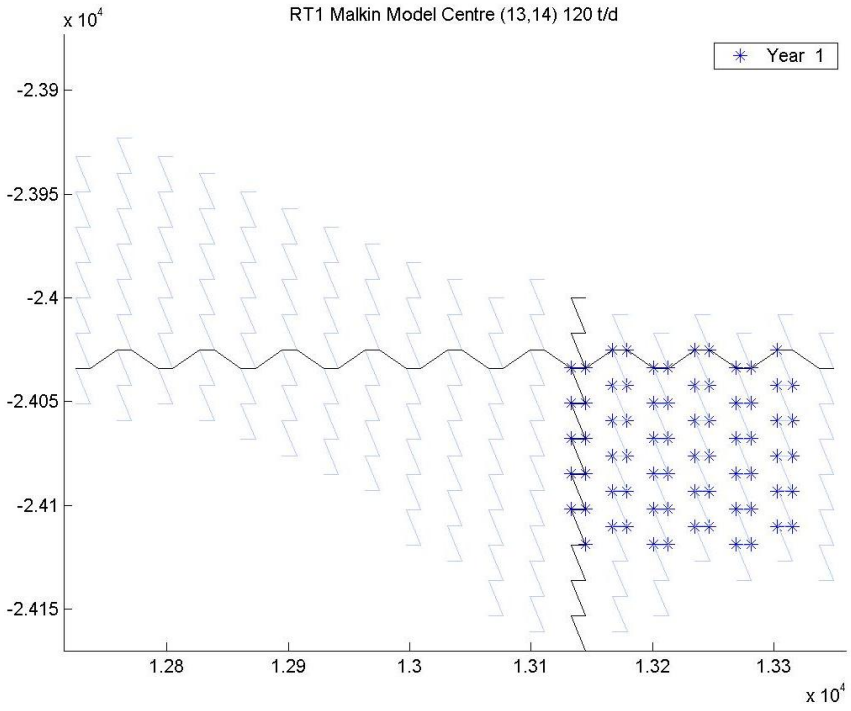


Figure 3.43: Best Results of Malkin Model for Data Set RT1 - Period 1

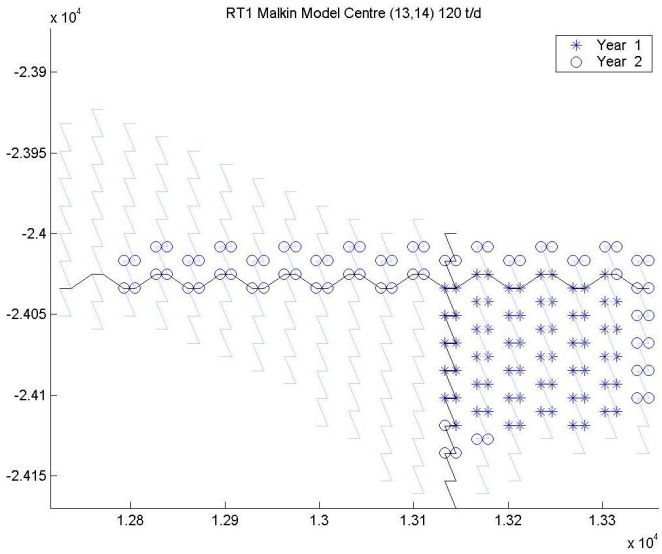


Figure 3.44: Best Results of Malkin Model for Data Set RT1 - Period 2

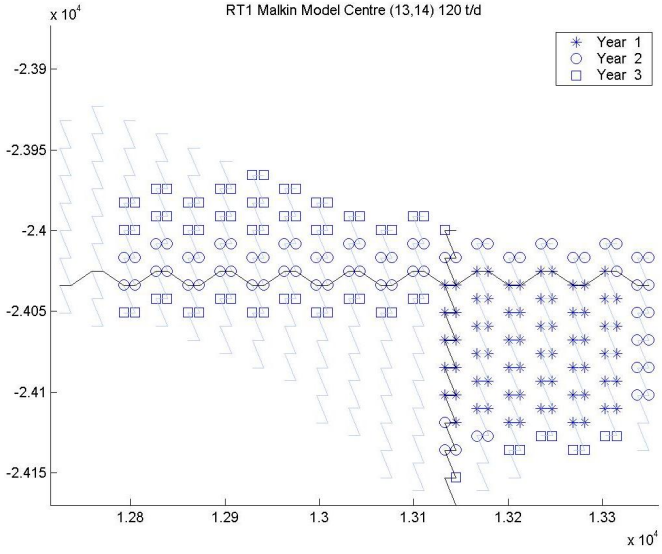


Figure 3.45: Best Results of Malkin Model for Data Set RT1 - Period 3

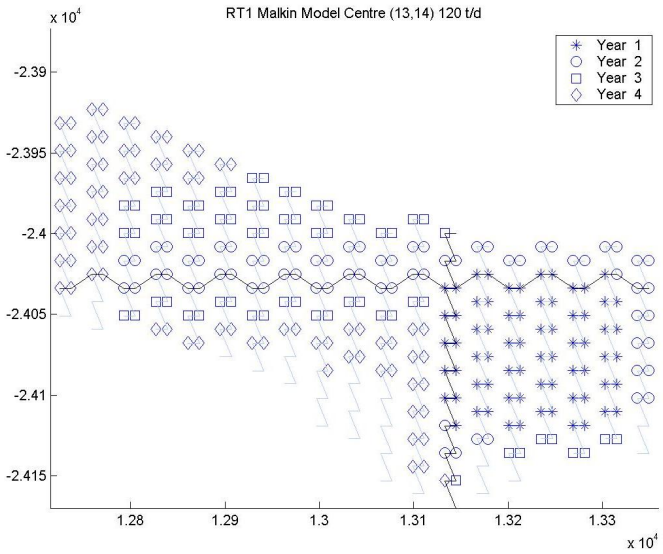


Figure 3.46: Best Results of Malkin Model for Data Set RT1 - Period 4

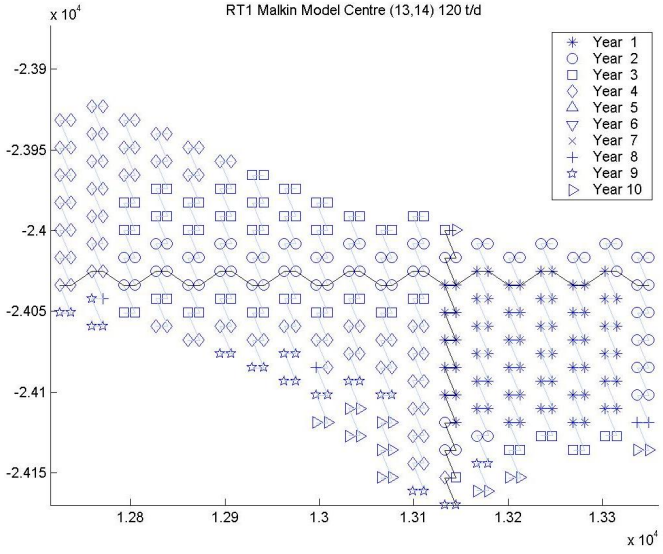


Figure 3.47: Best Results of Malkin Model for Data Set RT1 - Period 10

2Cone:

The 2Cone model generated a single cave in a diamond pattern. The first period cave is very similar to that chosen in the Basic model and the highest value Malkin model. After choosing the high value draw points in the first period, the model moved to the right before moving left through the negative value draw points.

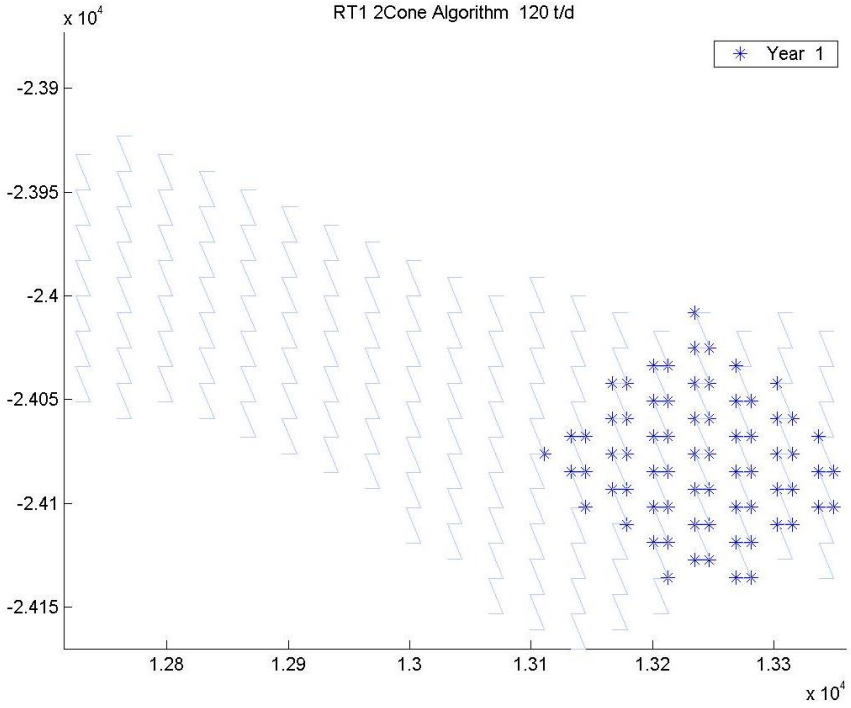


Figure 3.48: Results of 2Cone Model for Data Set RT1 - Period 1

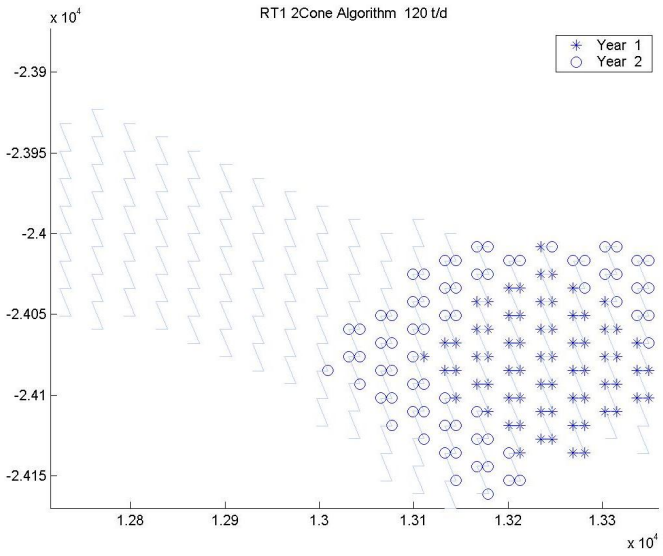


Figure 3.49: Results of 2Cone Model for Data Set RT1 - Period 2

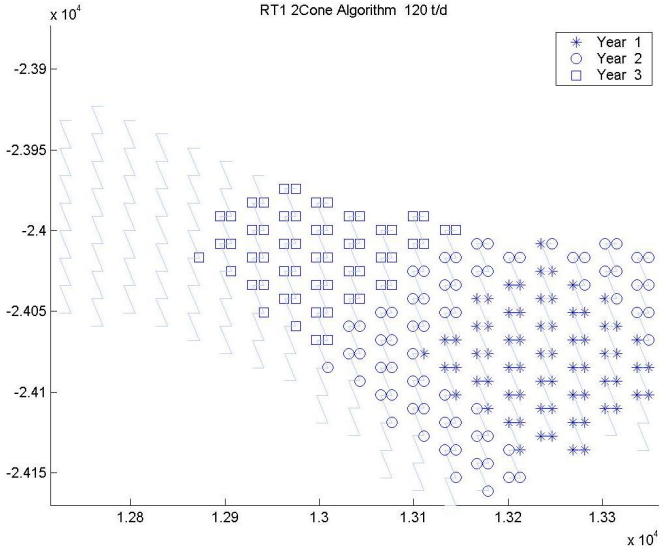


Figure 3.50: Results of 2Cone Model for Data Set RT1 - Period 3

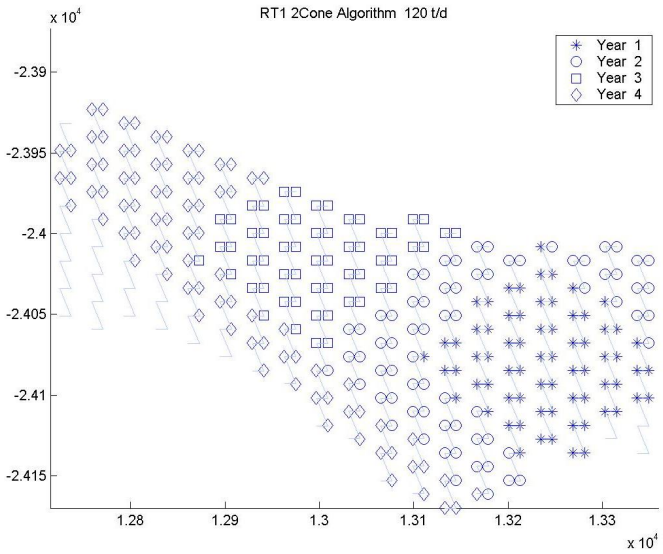


Figure 3.51: Results of 2Cone Model for Data Set RT1 - Period 4

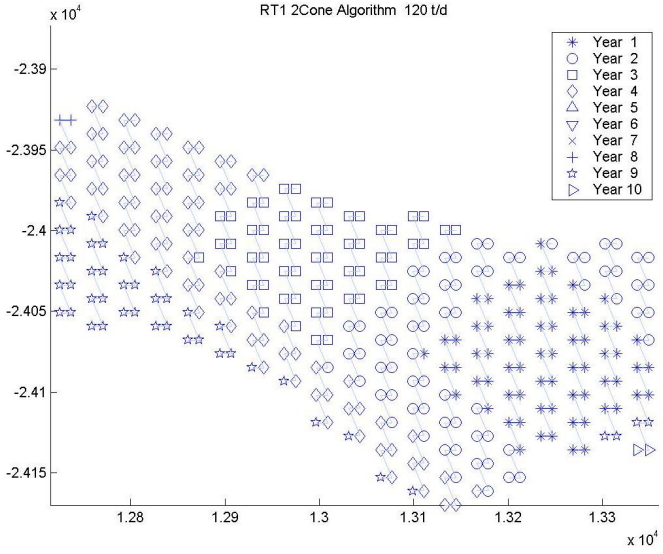


Figure 3.52: Results of 2Cone Model for Data Set RT1 - Period 10

ColGen:

The ColGen Model also generated a single cave in a diamond pattern, but the first period cave was not as centrally located as the other models. The first period cave includes some high value draw points but extends over to the right hand edge. Over the remaining periods the cave is extended left, then ends with draw points at the bottom of the layout.

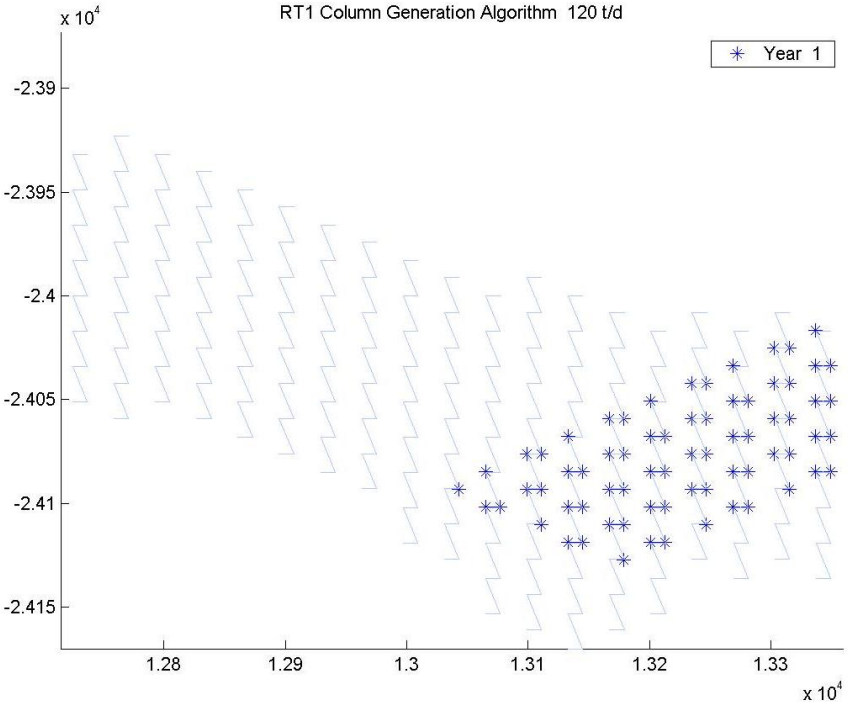


Figure 3.53: Results of ColGen Model for Data Set RT1 - Period 1

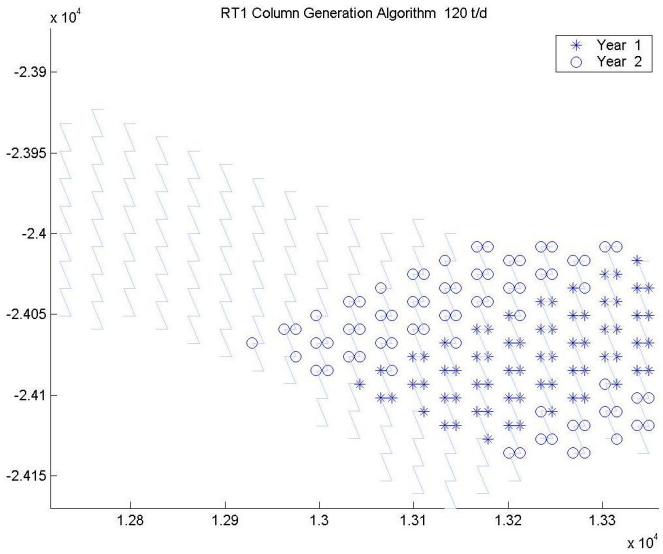


Figure 3.54: Results of ColGen Model for Data Set RT1 - Period 2

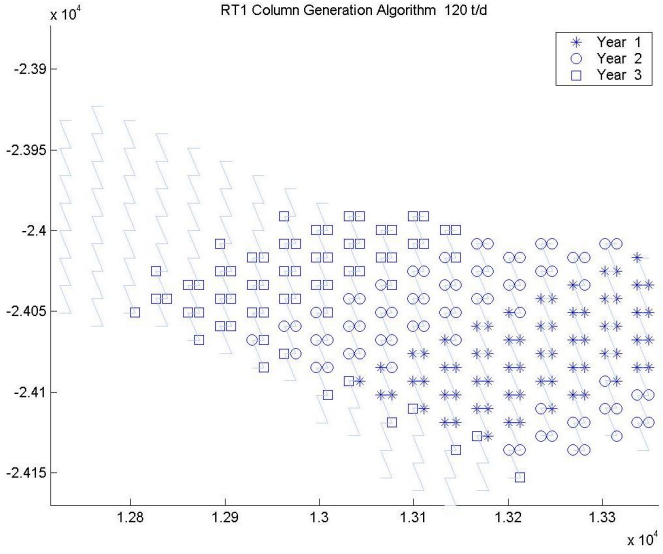


Figure 3.55: Results of ColGen Model for Data Set RT1 - Period 3

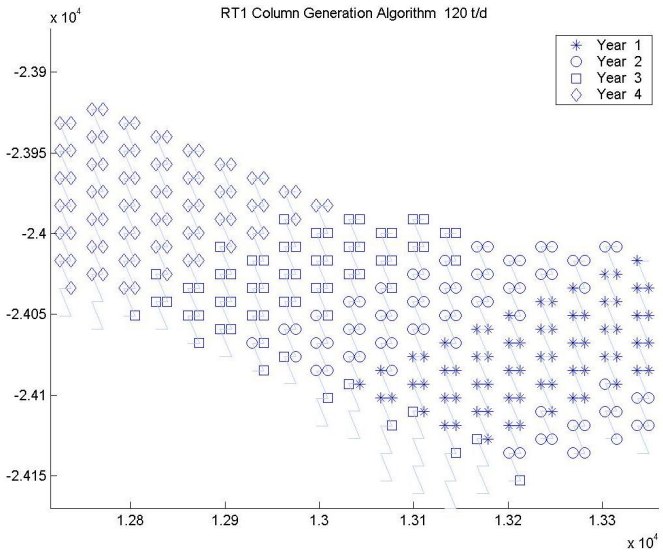


Figure 3.56: Results of ColGen Model for Data Set RT1 - Period 4

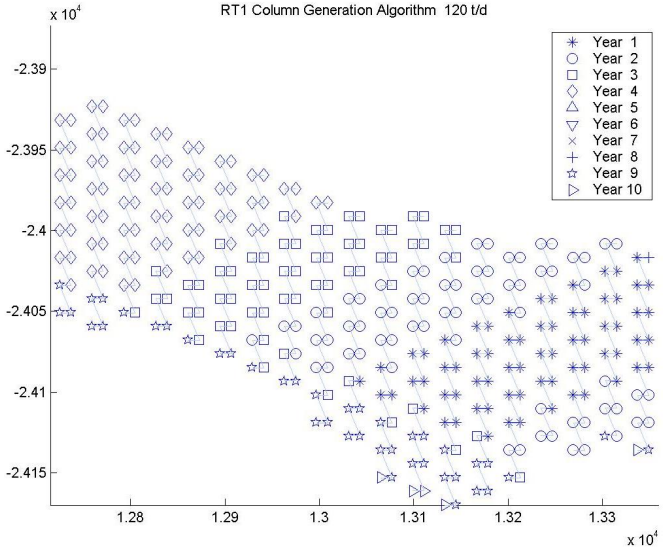


Figure 3.57: Results of ColGen Model for Data Set RT1 - Period 10

Greedy:

As noted above, there is very little difference between the Greedy algorithm results, and the improvements made by subsequent iterations of the ColGen model. Differences do not appear until year 4 and only on the edges of the cave.

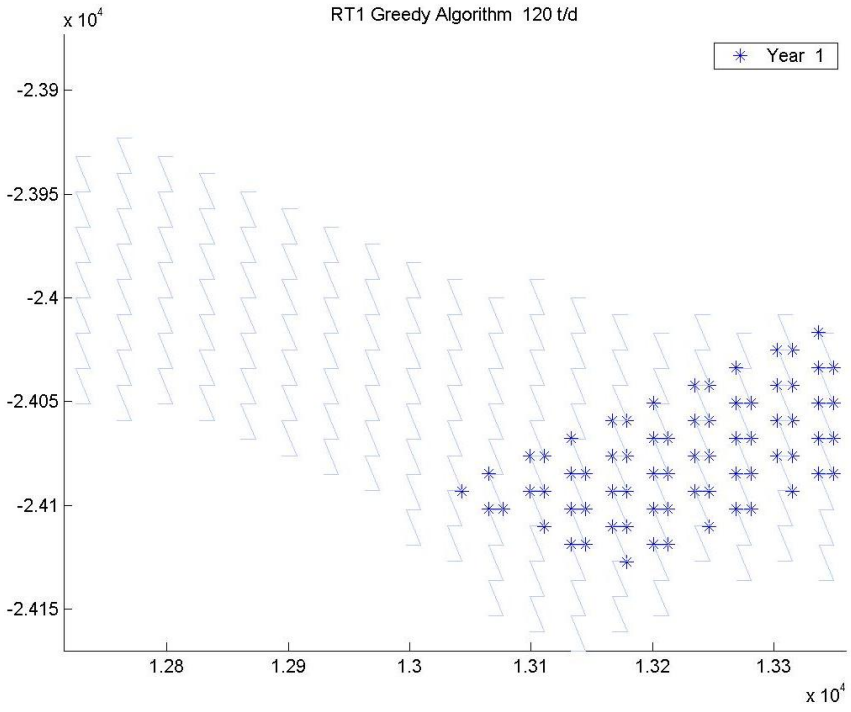


Figure 3.58: Results of Greedy Model for Data Set RT1 - Period 1

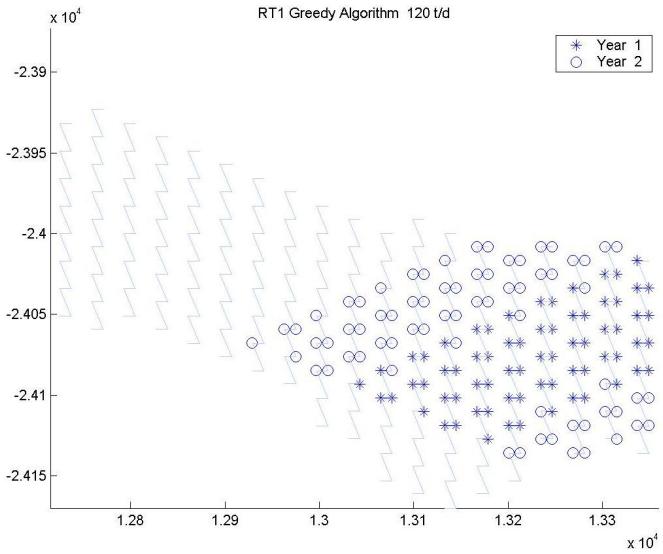


Figure 3.59: Results of Greedy Model for Data Set RT1 - Period 2

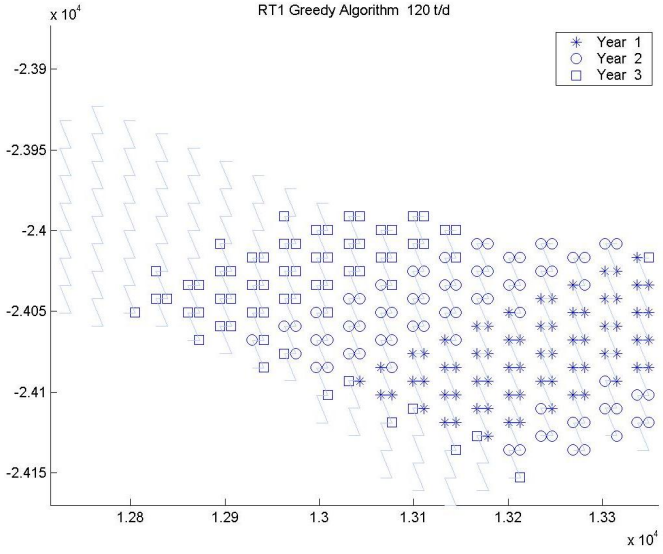


Figure 3.60: Results of Greedy Model for Data Set RT1 - Period 3

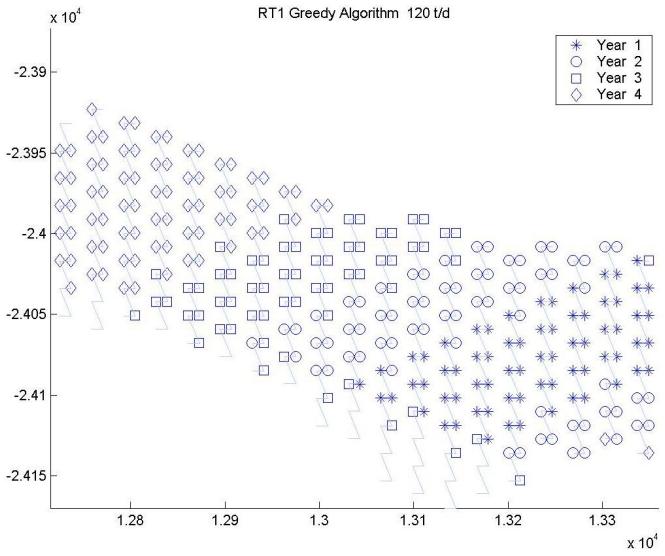


Figure 3.61: Results of Greedy Model for Data Set RT1 - Period 4

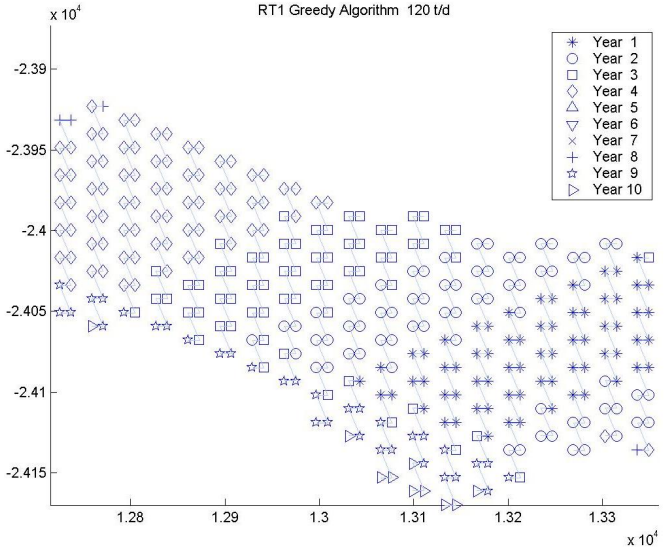


Figure 3.62: Results of Greedy Model for Data Set RT1 - Period 10

Comparison of first two periods over all models

This section is included to facilitate comparison of the results for the first two periods across all models. As can be seen below, all models start with the draw points in tunnels 13 through 19; tunnels numbered from left to right (Figure 3.33). The results are very similar, with the exception of those for the Basic model which moves to tunnels 1 through 10 in the second period and results in multiple caves.

While all models created a diamond-shaped cave in the first period, the 2Cone, ColGen and Greedy models are more successful at generating diamond-shaped caves in the second period. Combining this with the NPV value results suggests the 2Cone model performs better than the other models.

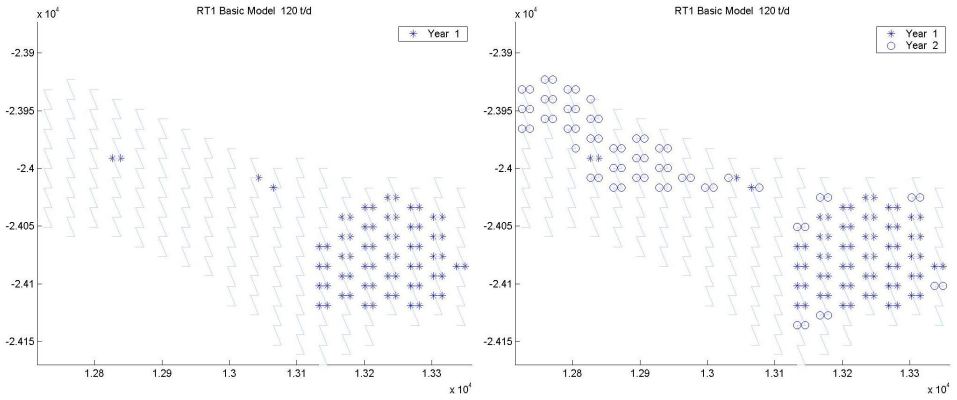


Figure 3.63: Opening Patterns for First Two Periods for Basic Model

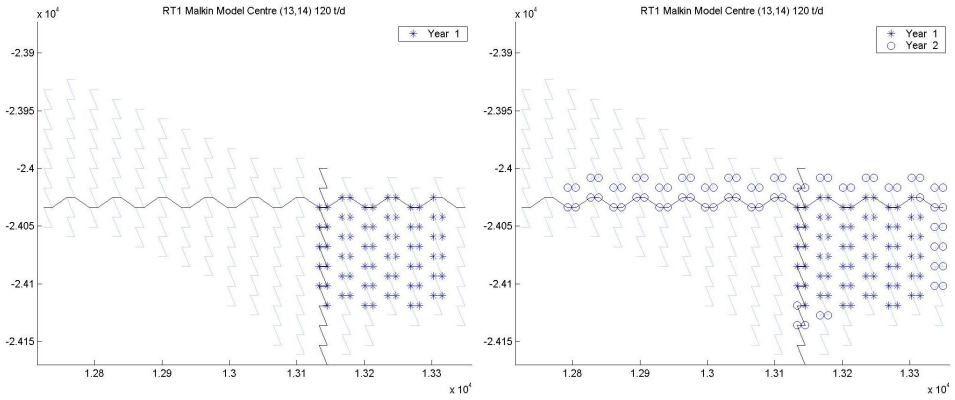


Figure 3.64: Opening Patterns for First Two Periods for Malkin Model

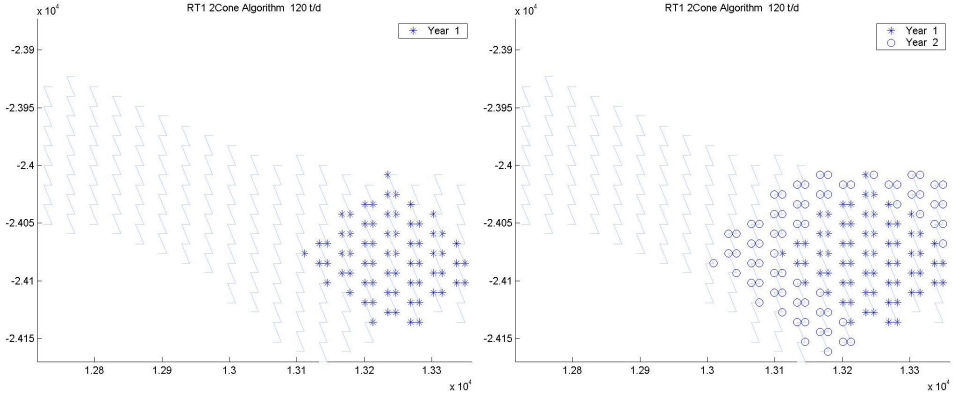


Figure 3.65: Opening Patterns for First Two Periods for 2Cone Model

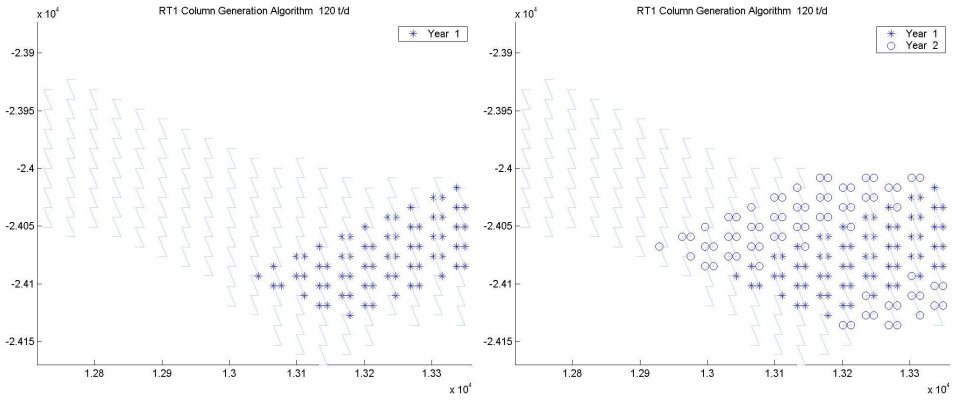


Figure 3.66: Opening Patterns for First Two Periods for ColGen Model

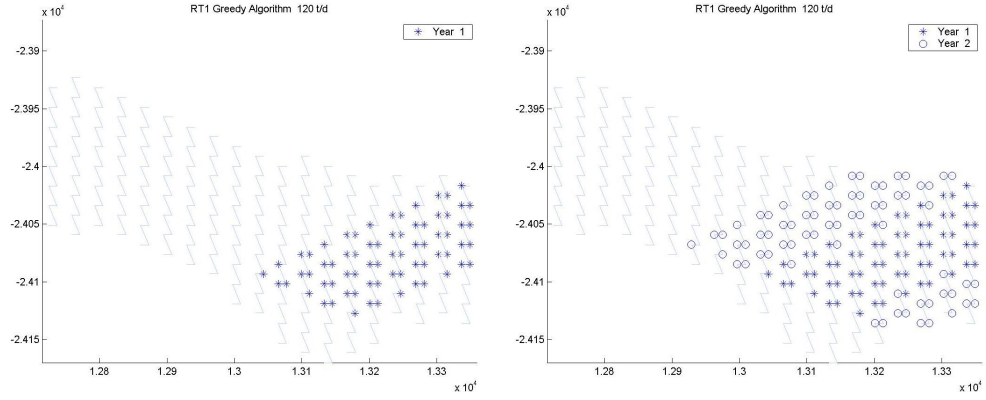


Figure 3.67: Opening Patterns for First Two Periods for Greedy Model

Sensitivity:

This data set was chosen for this project as it was thought by planners that its relatively small size provided few opening sequences and would not lead to much sensitivity to sequence patterns. To test sensitivity to sequence, some alternate scenarios, intended to be different from those generated above, were run as follows:

1. Start at LHS (2Cone): Force mining to start on the left-hand side of footprint, i.e., Tunnel 5 if counted from the LHS. This was done by forcing all the draw points in tunnel 5 to start in period 1 or period 2. All the results above started with the high value draw points in the centre or right of centre of the layout. The choice of tunnel 5 is distinct from the centre point of the model results above, and allows for a diamond shaped first period cave, without running into the edge of the footprint.

2. Start on Edge (First or Last Tunnel) (2Cone): Force mining to start on an edge (Tunnel 1 or Tunnel 19) instead of the centre of the footprint. As mentioned in Section 3.4.4, some mining sequences originate at the edge of a mining footprint so that the equipment used to remove the mined rock does not cross an active mining front. If the sequence does not start on a footprint edge, often the mined rock must be carried on a level below the mining to cross the mining front. Sequences that start on a footprint edge are of interest to understand the tradeoffs between these two approaches.

3. Start on Edge (Any) (2Cone): Force mining to start on an edge (Tunnel 1 or Tunnel 19 or "top" or "bottom")

4. Look for Minimum NPV (2Cone): run the model to minimize NPV. Since the easiest way to do this is delay starting all draw points until the end, or not open any at all, a new constraint was added to require at that least "maximum number of active draw points - 4" must be active in each period. This sequence can provide a measure of the worst case scenario of a badly chosen sequence, and demonstrate the potential value of the models on this data set.

5. Lowest value Malkin: The Malkin model solution with the fixed centre point yielding the lowest NPV. Again, this is a worst case scenario to show effect of chosen sequence on value.

6. Force mining to start in the middle (2Cone). This was done by forcing draw points in tunnels 10 and 11 (numbered from LHS) to start in period 1 or period 2. It was chosen to reproduce the published sequence (Calder (2000)) for this data set. Figure 3.68 below shows the sequence that was published for the data set.

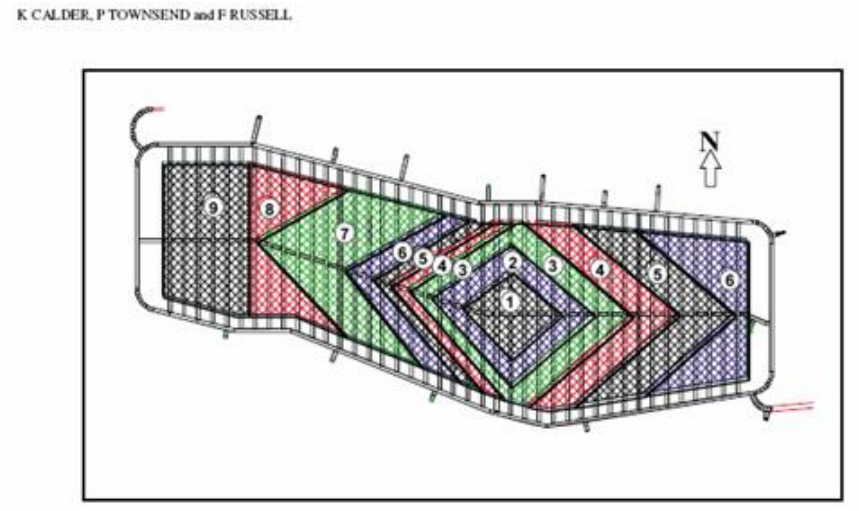


Figure 3.68: Published Sequence Optimization for Sample Data Set

Table 3.16 reports the results of the sensitivity runs. The 2Cone model from the earlier runs was used as a benchmark to compare the values of these alternate runs, based on its high NPV value and diamond-shaped caves.

The results in Table 3.16 do show that the total NPV of caves produced on this data set is sensitive to the opening sequence with many of the alternates 10% below the 2Cone value from the original runs. The results also show a variation in solution times. The individual Malkin run is faster than most of the 2Cone results. The more constrained the 2Cone model is, the faster the run completed. While the longest 2Cone solution time of 8 hours is not fast by any measure, it would be possible to run this model overnight, especially for the scale of costs involved with mine planning.

There is not much variation in the number of active draw points in each cave between the runs; some runs did not open a full 70 draw points in the first period. See Table 3.17

The caves from the first two years are presented below in Figure 3.69 through Figure 3.73 to confirm that the objectives of each scenario were met.

	Total NPV	% of 2Cone NPV	Solution time
Basic	$ 3,507,820	105.8	4 sec ≈ 0.001 h
2Cone	$ 3,314,810	100	3 050 sec ≈ 0.8 h
Start at LHS (2Cone)	$ 3,025,550	91.3	694 sec ≈ 0.2 h
Start on Edge (First or last tunnel)(2Cone)	$ 2,932,900	88.4	2 598 sec ≈ 0.7
Start on Any Edge (2Cone)	$ 3,304,930	99.7	1 564 sec ≈ 0.4 h
Minimize (2Cone)	$ 2,809,050	84.7	29 132 sec ≈ 8 h
Malkin (low value)	$ 3,063,040	92.4	303 sec ≈ 0.08 h
Start in centre (2Cone)	$ 3,136,930	94.6	149 sec ≈ 0.04 h

Table 3.16: Results of Sensitivity Runs:

	Number of draw points active in each period									
	1	2	3	4	5	6	7	8	9	10
Basic	70	132	205	284	284	284	284	284	283	254
2Cone	70	140	198	284	284	284	284	283	283	223
Start at LHS	66	135	202	284	284	284	284	284	283	234
Start on Edge (First or last tunnel)	67	137	202	284	284	284	284	284	279	267
Start on Edge (Any)	69	138	205	284	284	284	284	282	283	237
Minimize	67	137	202	280	280	280	280	260	238	251
Malkin (low value)	70	140	206	284	284	284	284	284	284	268
Start in centre	70	137	205	283	283	283	283	279	263	205

Table 3.17: Cave Sizes in Sensitivity Runs:

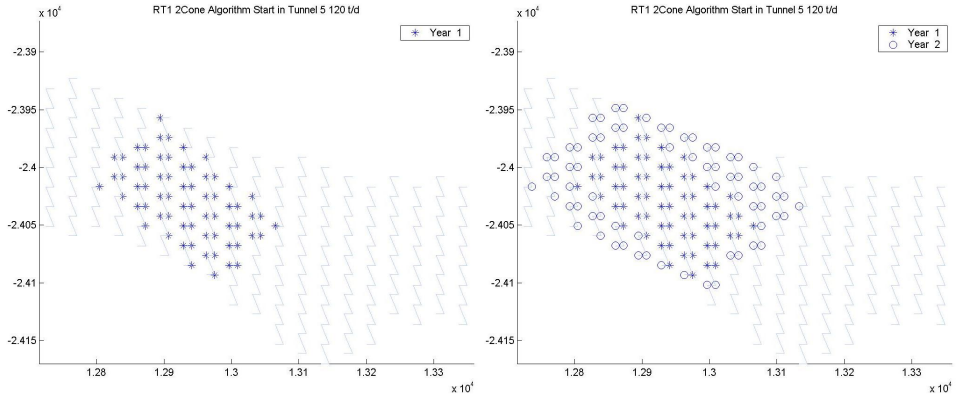


Figure 3.69: Results of 2Cone Model Starting at Tunnel 5

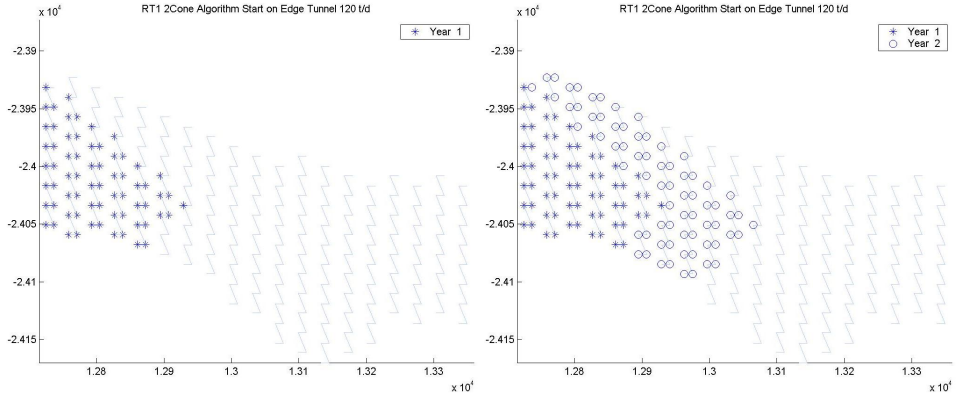


Figure 3.70: Results of 2Cone Model Starting on Edge Tunnel

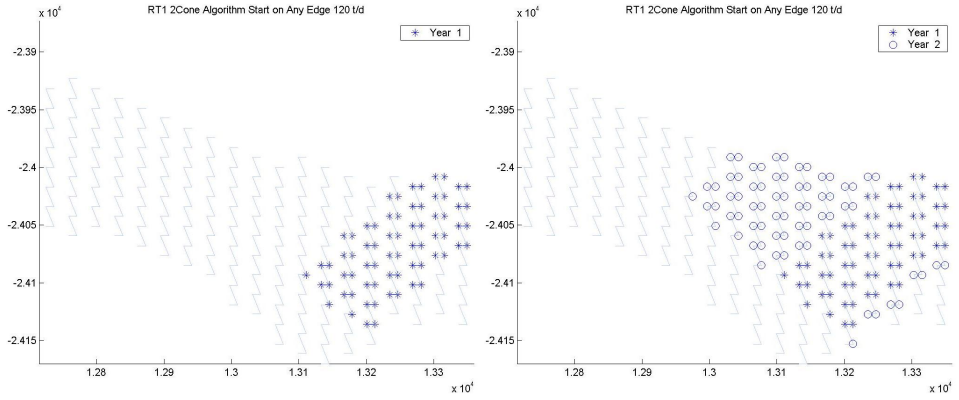


Figure 3.71: Results of 2Cone Model Starting on Any Edge

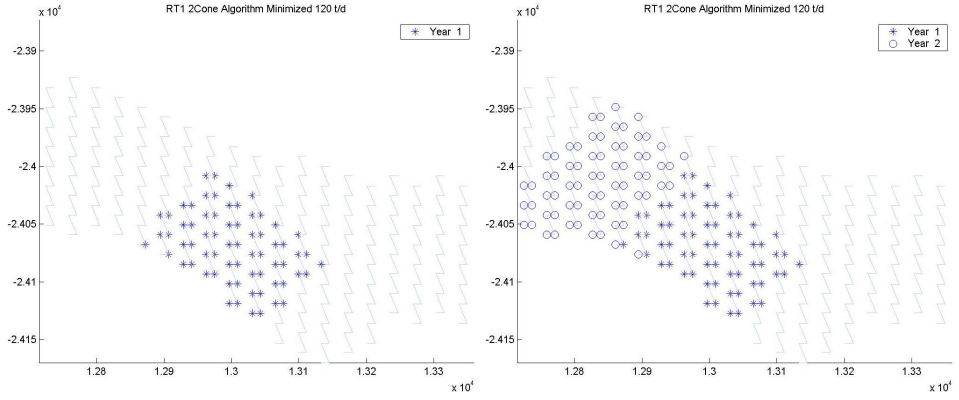


Figure 3.72: Results of 2Cone Model Minimized

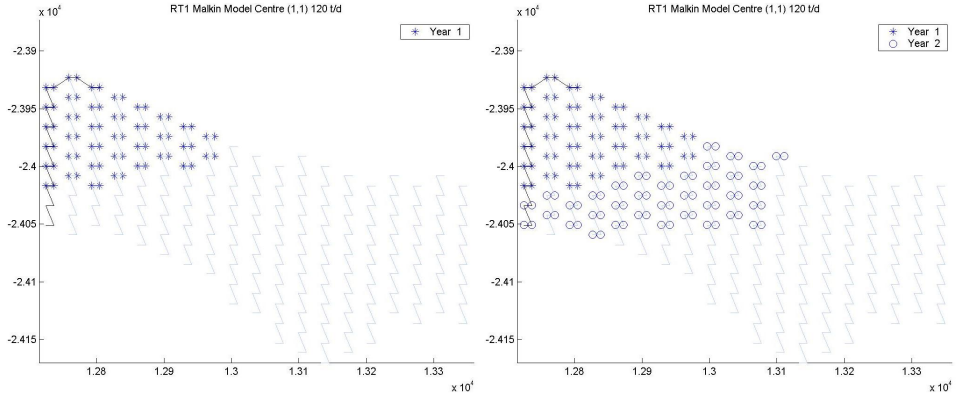


Figure 3.73: Results of Minimum Malkin

Start at centre This run was an attempt to reproduce the published sequence for this data set. Figure 3.74 below is an overlay of the sequence for the first 3 periods with the published sequence. While the published sequence starts in the middle of tunnel 5, as opposed to the bottom in the model result, both move to open the draw points in the tunnels to the right before moving towards the tunnels on the left. The value of the published sequence is not available for comparison.

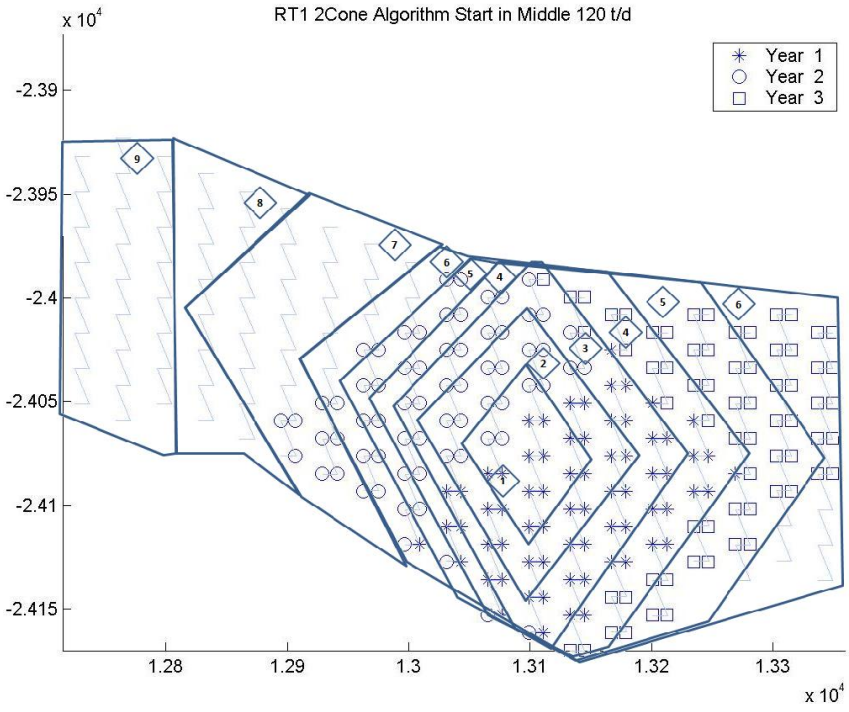
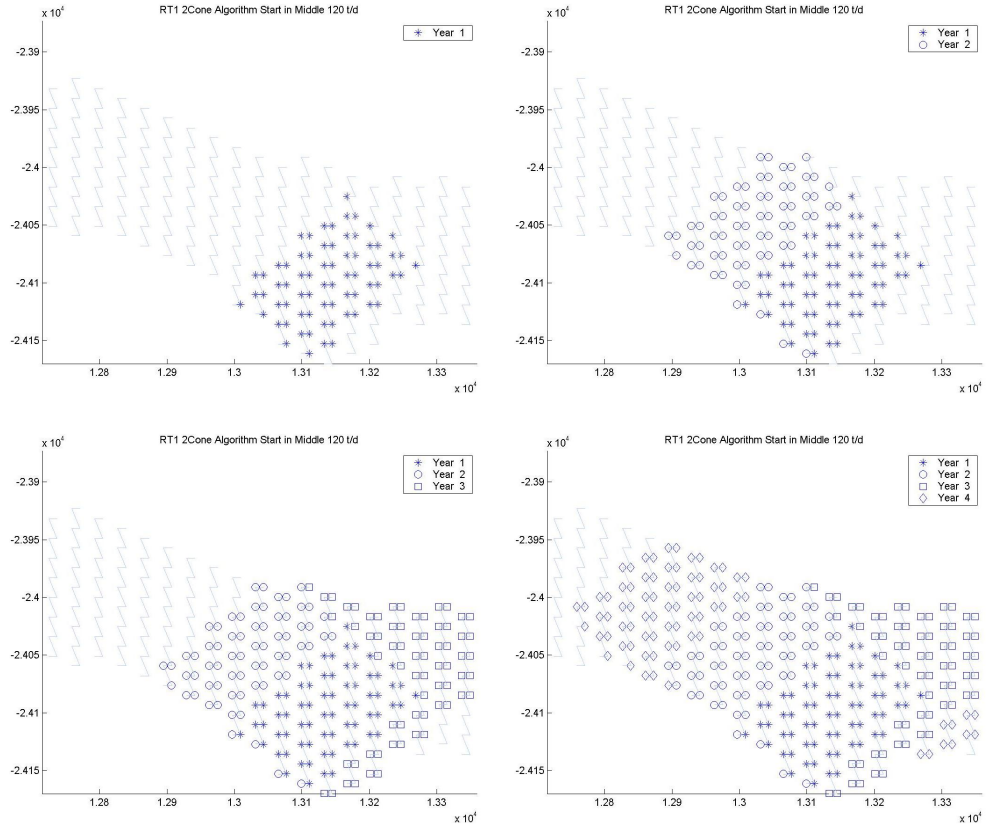


Figure 3.74: Overlay of Published Results and First 3 periods of Generated Results

The following figures show the generated results without the overlay of the published results. As mentioned, the first period cave is in the middle tunnels, but touches the lower edge of the mining footprint. In the second period it moves to the left, and after that grows in both directions.


RT1 2Cone Algorithm Start in Middle 120 t/d
Year 1
RT1 2Cone Algorithm Start in Middle 120 t/d
Year 1
Year 2
RT1 2Cone Algorithm Start in Middle 120 t/d
Year 1
Year 2
Year 3
RT1 2Cone Algorithm Start in Middle 120 t/d
Year 1
Year 2
Year 3
Year 4

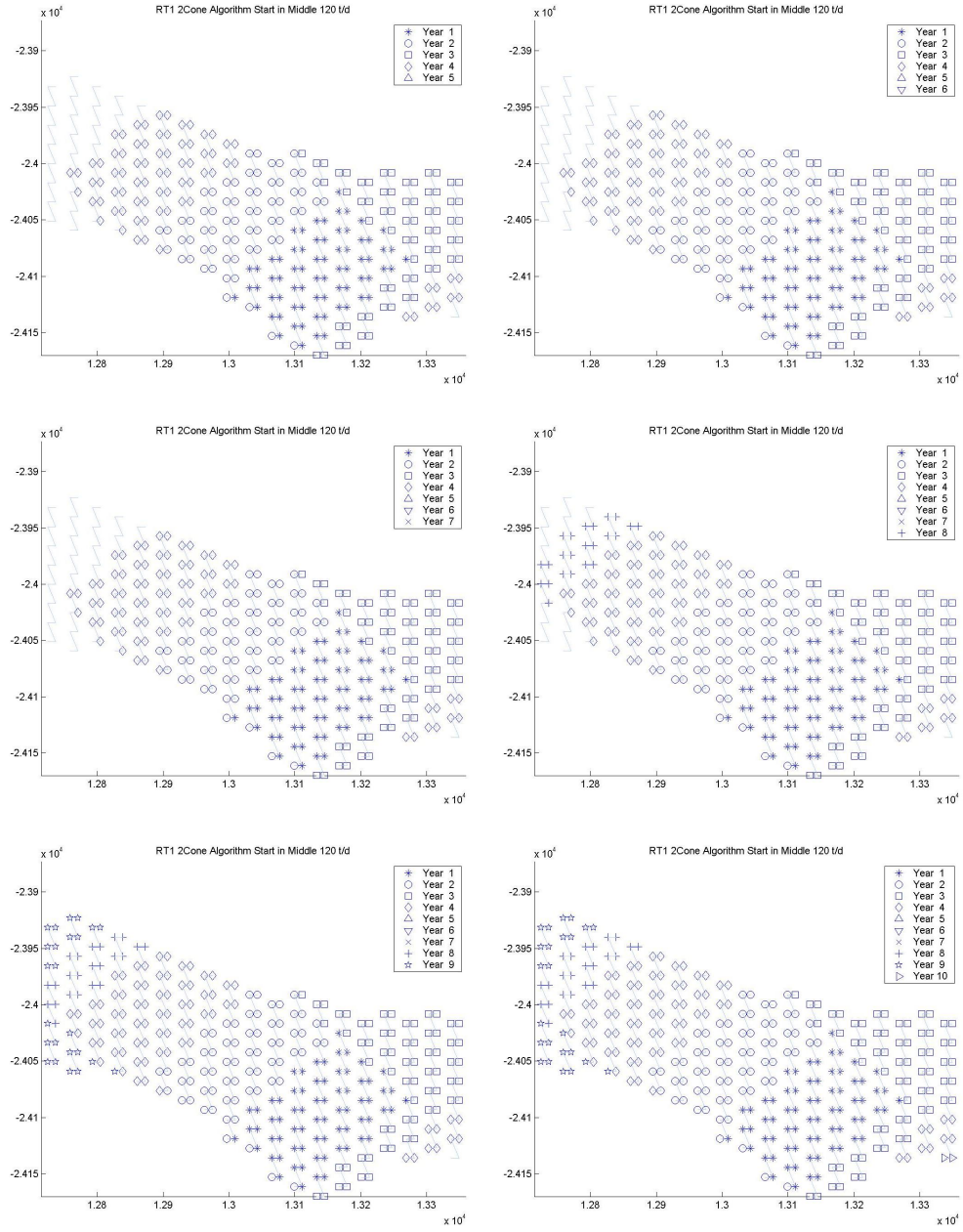


Figure 3.75: Results of 2Cone Model Starting at Centre

Results on Data Set RT2

Table 3.18 reports the results of the models discussed above. Due to the length of time of each run it was not feasible to run the Malkin model on all 6510 possible centre points. Instead a sampling was taken based on the draw points selected in the first period of the Basic model. The 2Cone model did not run to completion on this data set. After two days of calculations the results were not returned and it appeared the maximum memory was exceeded. Similarly, there are no results for the ColGen model as there were no results after several days of running and no intermediate results were returned.

	Total NPV	Number of Caves	% of Basic NPV	Solve Time (sec)	
Basic	$ 11,062,600	2	100	55 719.6	≈ 19 hr
Malkin (29,80)	$ 10,934,200	1	98.8	5 372.8	≈ 1.5 hr
Malkin (23,85)	$ 10,877,400	1	98.3	4 299.7	≈ 1 hr
Malkin (26,95)	$ 10,861,900	1	98.2	3 649.2	≈ 1 hr
Malkin (35,70)	$ 10,831,700	1	97.9	33 128.1	≈ 9 hr
Malkin (25,100)	$ 10,803,000	1	97.6	4 867.4	≈ 1 hr
Malkin (39,50)	$ 10,706,000	1	96.8	25 849.4	≈ 7 hr
2Cone				infinite	
ColGen				infinite	
Greedy	$ 10,441,300	1	94.4	> 6 days	

Table 3.18: Results for Data Set RT2:

The highest NPV is from the Basic model. However, inspection of the results, see Figure 3.76 shows that this model produces two contiguous caves. The Malkin run with the highest result is (10,934,200/11,062,600= 0.988) 98.8% of the Basic model result and the Malkin run with the lowest result is (10,706,000/11,062,600= 0.968) 96.8% of the Basic model result, demonstrating that the choice of centre point can improve objective value.

As with data set RT1, the Greedy model achieved a very good result achieving (10,441,300/11,062,600= 0.944) 94.4% of the Basic model result.

The models took considerably longer to run on this data set than on Data Set RT1, as expected. The Basic model took almost a day to complete, and the Greedy model almost a week. Since the ColGen model initializes with the Greedy model, it is not surprising that it did not complete. The Greedy algorithm evaluates the caves formed by each pair of draw points,

roughly 42 million combinations for data set RT2. Clearly there is some room to improve this algorithm and as a result, improve the speed of the ColGen model. While individual Malkin runs appear much quicker, many runs would be needed to find the best centre point.

Table 3.19 shows the number of active draw points in each period for the Basic, the highest and lowest total value from the sample of Malkin runs, and the Greedy algorithm. There are 201 draw points with negative value that must be mined for structural reasons. Some of the models may delay these until the last period and therefore discount this loss. This may explain why the models wait until the last period to open the remaining draw points.

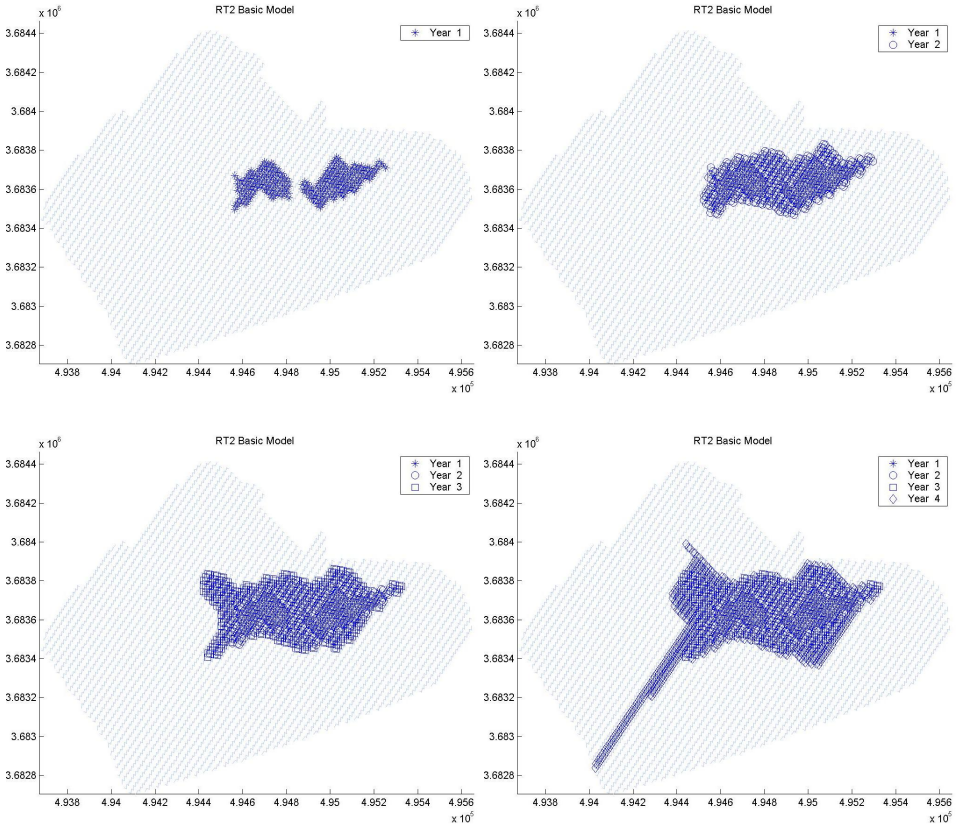
We now look at the mine shapes generated by solving these models on data set RT2. Inspection of the figures provides a count of caves formed. We also compare the shape and progression of the caves over time to see how the results from the different models vary.

Number of draw points active in each period	Basic	Malkin Maximum of sample (29,80)	Malkin Minimum of sample (5,80)	Greedy
1	288	288	278	288
2	566	566	543	566
3	835	831	794	836
4	1090	1087	1067	1117
5	1099	1086	1067	1106
6	1109	1117	1075	1102
7	1095	1102	1067	1086
8	1103	1154	1081	1112
9	1080	1126	1082	1108
10	1064	1073	1092	1089
11	1077	1030	1048	1095
12	1155	1113	1055	1095
13	1139	1113	1137	1104
14	1184	1172	1159	1134
15	1108	1126	1190	1114
16	792	846	1068	906
17	22	21	42	189
18	3	2	0	42
19	1	0	0	35
20	163	116	124	33

Table 3.19: Cave Development Results for Data Set RT2:

Basic:

The following sequence was determined using the Basic model. As has been found on other data sets, this model did not return a single contiguous cave in the first period. The results show that there is are two high value groups of draw points on the right side that are chosen in the first period. It also shows that there is a valuable group on the bottom left side that is not adjacent to those chosen in the first period but which the model seeks in the fourth period.



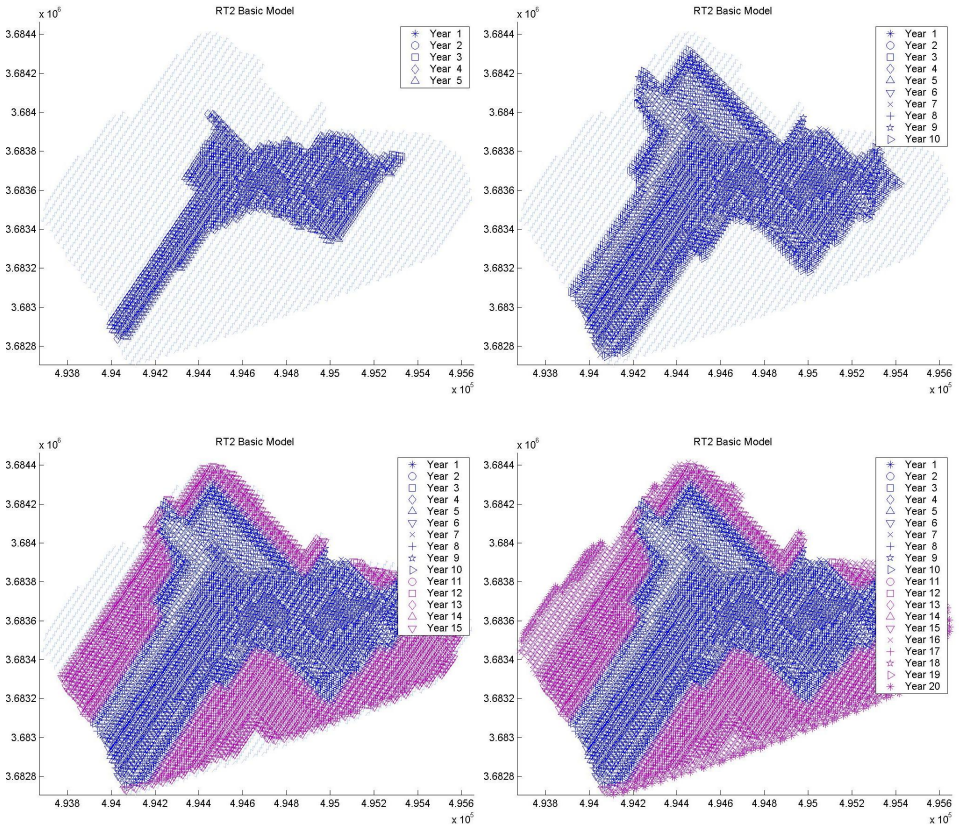


Figure 3.76: Results from Basic Model on Data Set RT2

Malkin:

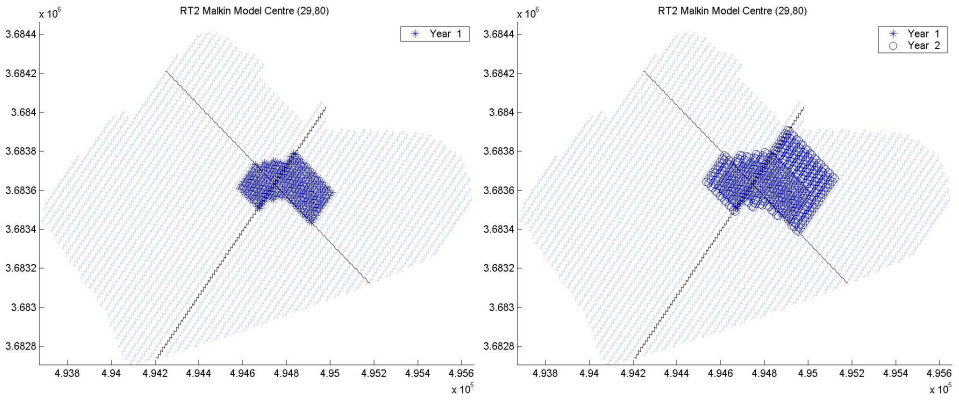
As each run of the Malkin model took between 1 and 9 hours it was not feasible to run the model for each of the 6510 draw points as the starting point. Instead 6 runs were done, each starting in one of the two caves selected in the first period of the Basic model run. Of the runs done, the maximum total NPV was centre point (29,80) with NPV $10,934,200, which was 2.6% higher than the minimum at centre point (39,50) with NPV $10,706,000. While there is not much difference in the total NPV between the Malkin runs, the two centred on (35,70) and (39,50) took much longer time to run than the others. These two runs were the only two centred in the right hand cave generated in first period of the Basic model.

Starting Tun	Starting Dpt	Total NPV	Number of Caves	Solve Time (sec)	
29	80	$ 10,934,200	1	5 372.8	≈ 1.5 hours
23	85	$ 10,877,400	1	4 299.7	≈ 1 hour
26	95	$ 10,861,900	1	3 649.2	≈ 1 hour
35	70	$ 10,831,700	1	33 128.1	≈ 9 hours
25	100	$ 10,803,000	1	4 867.4	≈ 1 hour
39	50	$ 10,706,000	1	25 849.4	≈ 7 hours

Table 3.20: Results from Selected Runs of Malkin Model on RT2

Due to similarities in the caves generated, graphical results from only the runs centred at (29,80) and (39,50) are shown below.

Tunnel 29, Draw point 80 This run was started in the left hand cave selected in the first period of the Basic model run. As with the Basic model, the cave moves to the right, then heads towards the bottom left of the footprint in period five.



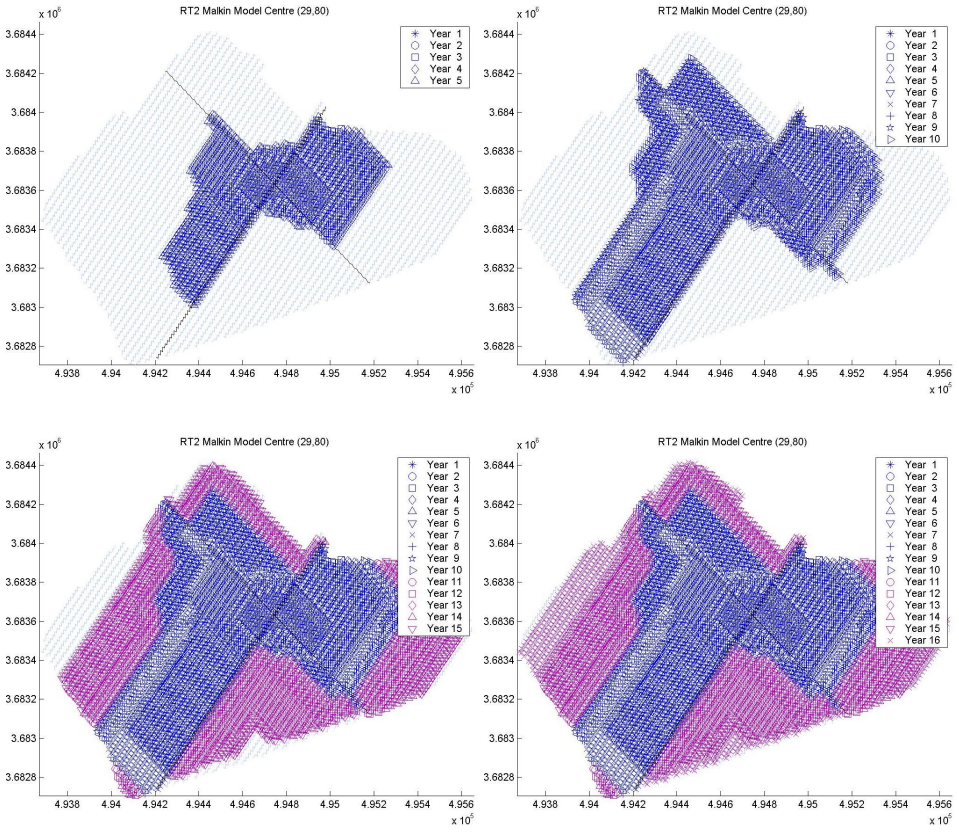


Figure 3.77: Results from Malkin Model (29,80) for Data Set RT2

Tunnel 39, Draw point 50 This run was centred on the upper right corner of the right hand cave selected in the first period of the Basic run on this data set. As with the other Malkin run from the right hand side, Malkin (35,70), the cave moves to the left, almost reaching the edge by period five. The cave then moves to the bottom left of the footprint, which is notably different from all runs other than Malkin (35,70).

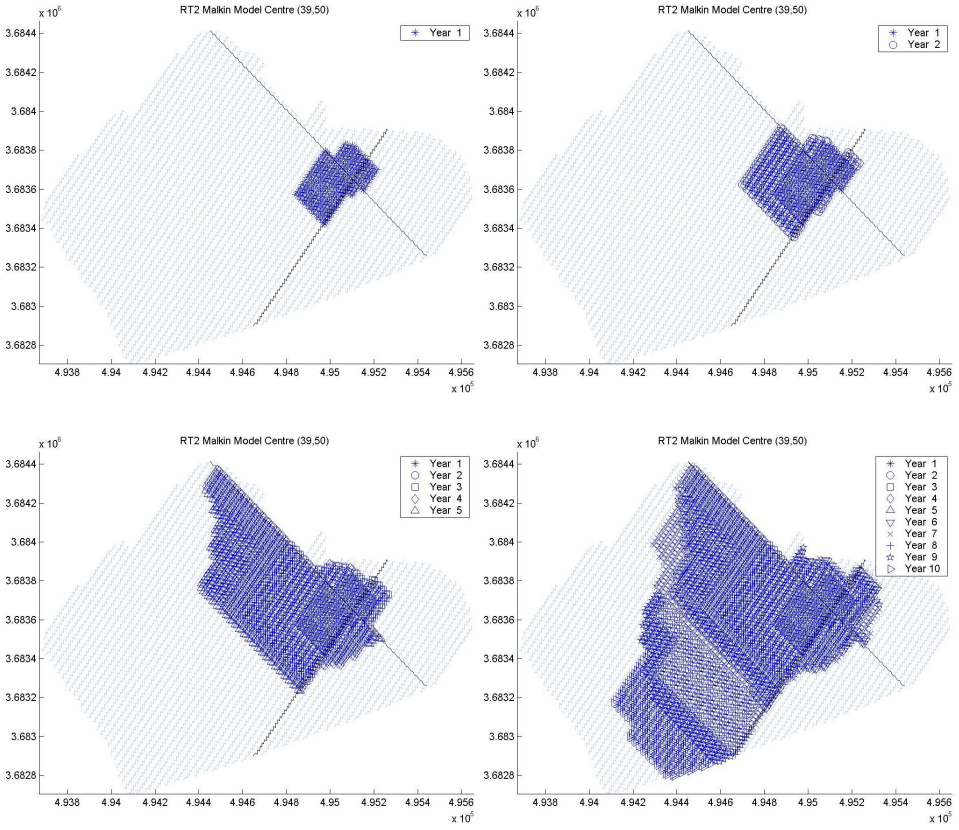
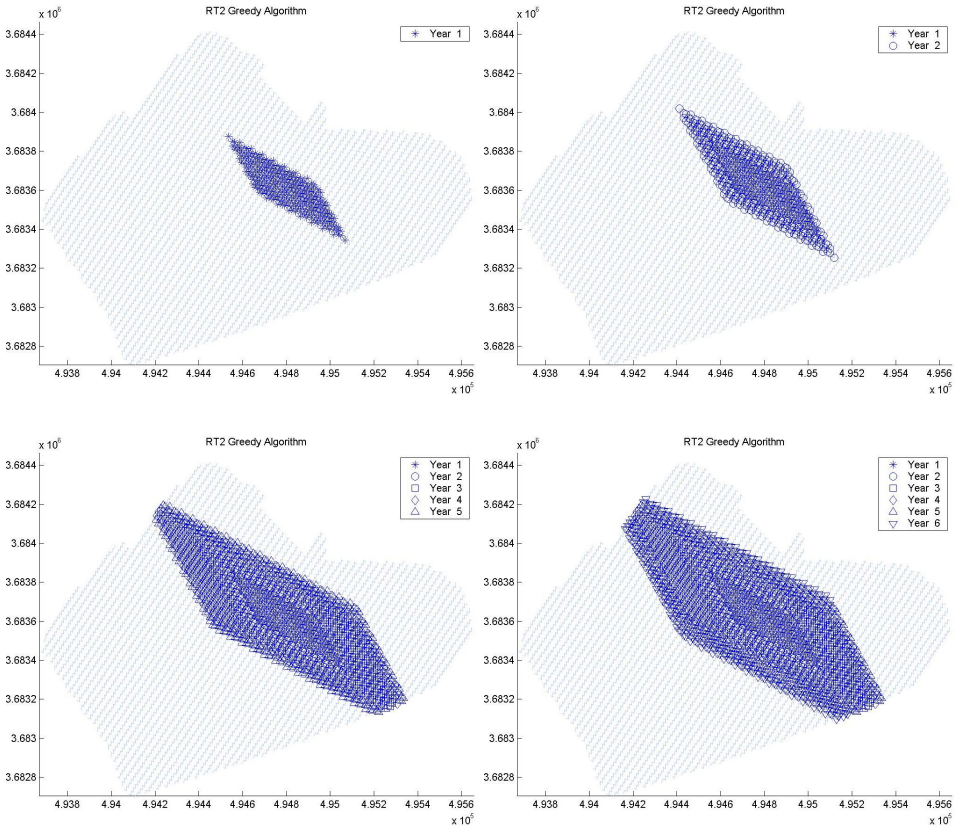


Figure 3.78: Results from Malkin Model (39,50)

Greedy:

The Greedy model returned a diamond-shaped sequence that begins in the same area as the Basic model. The cave grows to the left hand edge in the early periods but the tight shape restrictions prevent it from reaching the bottom left until much later in the run.



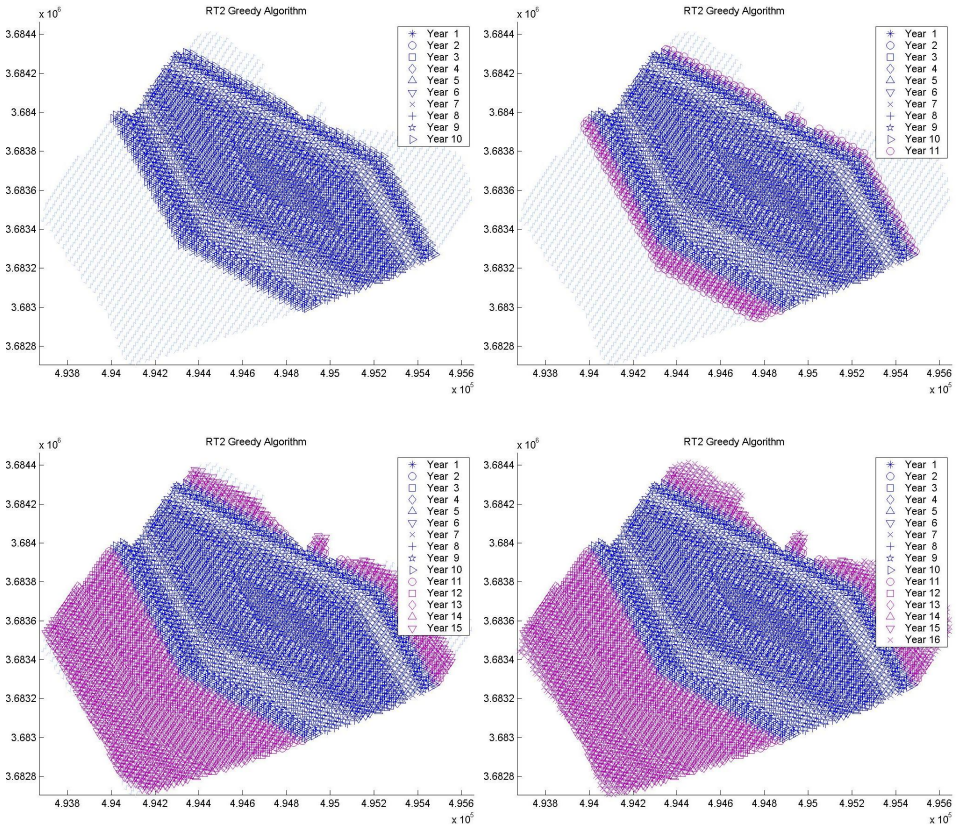


Figure 3.79: Results from Greedy Algorithm on Data Set RT2

3.5.4 Conclusions

The new models demonstrated their contribution on the two data sets. On both data sets the models not only identified high value opening sequences, but also demonstrated sensitivity of total NPV to opening sequences. While the Basic model did not produce a feasible opening in the early periods, it did provide an upper bound on the NPV as well as indicate the sections of the footprint with high value. The Malkin model ran well on both data sets, and provided feasible opening sequences as well as an opportunity to test the sensitivity of the results to different initial opening constraints. The 2Cone model was effective on the smaller data set, but proved to be too large for the larger data set. On the smaller data set, the 2Cone model also proved an effective tool for runs to investigate the sensitivity of the NPV to restrictions on the opening sequence. The Greedy algorithm produced results that were close to those of the other models for Data Set RT1. For Data Set RT2 the total NPV was close to the maximum, but the generated caves were noticeably different from those from the Basic and Malkin model runs. Finally, the ColGen algorithm produced results similar to those of the 2Cone model for Data Set RT1, but, like the 2Cone model, was not able to manage the large number of draw points in Data Set RT2.

From these two data sets, it seems that no one model provides the solutions that a practitioner seeks, but instead each contribute to the understanding of the problem. The Basic model provides an upper bound on the total NPV and indicates high value starting areas. Unfortunately, the frequent generation of multiple caves makes the solution impractical. In large data sets, running the Malkin model on all centre points may be infeasible due to time constraints. However, individual runs can be useful in investigating sensitivity to centre point or other constraints. The 2Cone model also proved useful, and ran in a reasonable time on the smaller data sets. The Greedy algorithm performed reasonably well but as it was not faster than the others it seems less useful than the other models.

3.6 Conclusions

In the preceding sections we presented five model formulations to assist the mine planner with the sequence optimization process for planning a block cave mine. The models each have strengths and weaknesses which will be discussed below. This is a complex problem, and this work has not found a simple answer. Instead, a combination of the presented models is proposed to provide insight to the planner.

Model	Number of (Binary) Decision Variables	Number of Constraints
Basic	$nDPTS \times T \times 3$	$nDPTS + T + nDPTS \times T \times 4$ $+nTNLS \times T \times 2$
Malkin	$nDPTS \times T$	$nDPTS + T + nDPTS \times T \times 2$
2Cone	$nDPTS \times T \times 5+$ $eTNLS^2 \times T \times 2$	$nDPTS + T \times 3 + nDPTS \times T \times 7$ $-nDPTS \times 2$

Table 3.21: Integer Programming Formulation Comparison

All models share the common goal of creating a single, contiguous cave, the size of which is regulated over time by a production capacity. The models differ on how they achieve the single cave, as this proved a challenging constraint to implement. All models require the same data preparation, namely the numbering of draw points on a two-dimensional grid with one axis aligned with the planned tunnels, and a second axis perpendicular to the first. The first axis is referred to as "along-tunnel" and the second as "across-tunnel".

Three of the models, Basic, Malkin and 2Cone, use general integer programming methods for the formulation. The ColGen model, based on the column generation technique and a Greedy algorithm round out the set of models.

Table 3.21 compares size of the three models based on integer programming formulations. The Malkin model is the smallest both in number of decision variables, and number of constraints.

Basic Model

The Basic model is the most general of the models. The cave is allowed to grow as long as, in each period, the opened draw points form a single, contiguous group along each tunnel, and a single contiguous group in the across-tunnel direction. With this first model, there is no attempt to create the desired diamond shape, only a single, contiguous cave in each period.

The single, contiguous group is created by specifying a direction to travel along each axis, then, for each row, naming an open draw point following a closed draw point the "First opened" and allowing only one in each period.

Thus, in each row of each axis there is only one transition from closed to open draw points, creating a contiguous group along each row.

Running this model revealed that it does not guarantee a single cave in each period. Apparently, for some data sets, the constraints are needed along more axes than the two defined for this work.

Despite the failure of the formulation to produce a feasible sequence, there is value to this model. The formulation may return a single cave in each period, or even a sequence that can be modified into something useful by the mine planner. If multiple caves are returned, the result can be used as an upper (unreachable) bound against which the sequences created by the skilled mine planner, and other models can be evaluated. The caves generated from the early periods can also be used to highlight high value areas which may be used as input into other models presented.

The Basic model ran to completion on the two data sets tested. It ran in 4 seconds on the smaller data set, but this rose to 19 hours for the larger one. To some, 19 hours may seem unexpectedly long, but in practice it may be acceptable to run it over the weekend if it does not need to be repeated often.

Malkin Model

The Malkin model was based on the 2D Integral Formulation proposed by P. Malkin and L. Wolsey. In this formulation, a centre point is specified which locates a dominant axis in each of the across-tunnel and along-tunnel directions. In each period, a draw point can open if those between it and each dominant axis are open. The formulation is the smallest of those presented as it is simple and does not require any decision variables in addition to the set describing which draw points are open.

For a convex mine footprint, this formulation guarantees a single, contiguous cave in each period. While the dominant axes can suggest a diamond shape, there is nothing in this model that evenly distributes open draw points on either side of the axes. In trials on our data set, this leads to a variety of shapes from a square with the origin of the axes at one corner, to irregular shapes with long, thin protrusions along an axis. Additional work would be required by the mine planner to obtain a usable sequence of caves, and this work would change the total NPV of the sequence.

The obvious challenge with this model is the task of choosing a centre point. In this work we were able to run the model on all centre points in less than an hour for a smaller data set. On the larger data set, individual runs took over an hour making it infeasible to run all 6510 centre points. For the

larger data set we ran the model on a selection of centre points chosen from the caves formed by the Basic model in the first period. The best result from this selection of runs was evaluated against the upper bound formed by the Basic model result.

On the smaller test data set, the best Malkin model achieved 98.6% of the Total NPV of the Basic model, while the best of the selection of runs on the larger data set achieved 98.8% of the Total NPV of the Basic model. The test data sets showed that the strengths of this model are that it can return sequences for a single cave, and they can be close to the upper bound created by the Basic model. This is in contrast to some of the other models that did not return a solution on the larger data set.

This model does provide the mine planner with a way to compare a variety of starting points in the mine footprint. For example, production method constraints may require the cave to begin along an edge of the footprint. Running the model on a selection of points around the perimeter could identify which edge is most favorable. In theory all of the integer programming models can perform this task, but this model is the smallest and individual runs were faster than Basic model runs on our trial data sets.

2Cone Model

In the 2Cone model the cave for each period is defined by the intersection of two cones, or triangles. Each cone has a draw point as its vertex, then grows from the vertex by two draw points per tunnel. Other than the constraints that ensure previous caves are contained within the current one, the vertices are unrestricted across the mining footprint.

This formulation will provide a single cave in each period on a convex data footprint. The shape of each cave is much closer to a diamond shape than those produced by the Basic model and Malkin model, especially in the earlier periods.

The definition of each of the cones uses two sets of decision variables, one for the vertex, and one for the cone. As a result, this was the largest of the integer programming formulations. On the smaller data set, this model took slightly longer to run than running the Malkin model on all centre points. The value of the result was lower than that of the best Malkin model run, 94.5% of the Basic model result, suggesting that the shape constraint was more restricting than the fixed dominant axes constraint. This model did not complete on the larger data set.

The strength of this model is in the creation of diamond shaped caves. This formulation has specified that the cave grows at two draw points per

tunnel, a restriction which may not suit all, if any, mining footprints. We would suggest that this model, or a similar one with a different diamond proportions, could be used in conjunction with one of the above models. The above models can identify draw points that should be in the initial cave. If the cave sequence is not well shaped, the 2Cone model could be used to suggest improvements. With an additional constraint specifying a set of draw points to open in the first period, the 2Cone model can be run to create well shaped caves over all periods. Consultation with the expert mine planner can suggest other diamond proportions that are acceptable.

ColGen Model

The ColGen model uses the Column Generation technique to separate the task of generating feasible caves from that of picking the cave in each period that maximizes the total NPV. The later task still utilizes Integer Programming techniques, but we were free to use other methods to generate the caves. In this work, we used an algorithm based on the 2Cone model to generate the feasible caves for each period.

The technique first requires a set of caves for all periods, then the Master program selects the sequence that maximizes the total NPV. The shadow prices generated by the Master program are then used to generate new caves that will improve the NPV, and so the model iterates until no more improvement is possible.

Creating the set of caves for each period proved complicated as there was not always sufficient information in the shadow prices to create reasonably sized caves that overlapped in consecutive periods. By reasonably sized we mean close to but not exceeding the Global Capacity Constraint. The algorithms added to estimate limits on cave sizes in each period, and generate feasible preceding and succeeding sequences greatly increased the size of the model, and the time to run.

On the small trial data set, the result from the ColGen model was within half a percent of the 2Cone model result. This is not surprising as both models were based on the same cave shape definition. As with the 2Cone model, the larger trial data set proved to be too large, and a solution was not returned.

Greedy Algorithm

The Greedy model is the algorithm that was developed to generate the initial cave sequence for use in the ColGen model. The algorithm enumerates all

caves of maximum size, and chooses the one with the highest total of draw point weights for the first period cave. For each subsequent period, it counts how many draw points have completed, then enumerates all caves of the new size that fully overlap with the cave from the previous period. The cave with the highest incremental draw point weight becomes the cave for that period and the process is repeated. In this work, caves matching the shape of those in the 2Cone model were generated.

This model ran to completion on both the trial data sets. On the smaller data set, the model generated a sequence that achieved $3,333,520 which is essentially the same as the ColGen result of $3,333,938. On this data set, the additional iterations of the ColGen model did not significantly improve on the starting sequence generated by the Greedy algorithm. As mentioned above, the Greedy algorithm / ColGen model result is also very close to that from the 2Cone model. On the larger data set, the Greedy algorithm ran for over 6 days, but did return a solution, unlike the ColGen and 2Cone models. The value of the solution was 94.5% of the Basic model solution, not leaving much room for improvement had the ColGen model run to completion.

In these two trial data sets, the Greedy algorithm performed strongly. If cave shapes that can not be well defined in the integer programming framework are desired, we suggest running this model, modified to the desired shape, and comparing the result to the Basic model benchmark. If there is a large gap, the ColGen model may add some value, but if not, the extra computation may not be worthwhile.

In theory, the Greedy algorithm (and the ColGen model) can be adapted to generate other cave shapes. The results from these trials show that even with the simple, single cave definition, the Greedy algorithm can be slow. This suggests additional, or more complex cave definitions may perform even worse. Depending on the size of the data footprint, and the complexity of the problem, the mine planner can decide if using this model is worth the time to reach a solution.

Future Work

This work is a first attempt to use models to generate opening sequences for planning block cave mining. In the process, we learned, as often happens with real world problems, that writing the rules, or constraints of this task is complex. We have shown that it is possible to write models that can generate sequences, but before any further work is done we recommend consulting expert mine planners to learn their opinion on how useful the models are.

The slow speed of the Greedy algorithm directly impacts the speed of the ColGen model. The Greedy algorithm is very simplistic and evaluates the cave made by each pair of draw points, approximately $nDPTS^2$ of them. Future work could be focused on this algorithm to make it more efficient and faster.

Much further down the road, if useful sequences can be generated, work could be done to combine this program with PC-BC, Gemcom's mine planning software.

In real life, much of the data is stochastic (ore price, discount rate, etc.) and so having a model that takes this into account would be valuable.

The models in this work are all in the integer programming setting. There are many other optimization techniques available, and this problem could be tried on them. For example constraint programming (Marriott 1998) or genetic algorithms and other global optimization algorithms (Weise 2008).

References

Ahuja, Ravindra K., Thomas L. Magnanti, and James B. Orlin. 1993. Network flows: theory, algorithms, and applications. *Prentice Hall.*

Americal Museum of Natural History. (2008). Mining a Kimberlite Pipe. *In The Nature of Diamonds.* Retrieved June 25, 2008 from http://www.amnh.org/exhibitions/diamonds/mining.html.

Barber, J., Thomas, L., & Casten, T. 2000. Freeport Indonesia's Deep Ore Zone Mine. *Proceedings MassMin 2000.* 291.

Calder, K., Townsend, P., & Russell, F. 2000. The Palabora Underground Mine Project. *Proceedings MassMin 2000.* 219–225.

Chen, F. 2000. Optimal Policies for Multi-Echelon Inventory Problems with Batch Ordering. *Operations Research.* **48** no. 3, 376–389.

Dantzig, G. B., & Fulkerson, D. R. 1956. On the Max-Flow Min-Cut Theorem of Networks. *Linear Inequalities and Related Systems* 215–221.

Diering, T. 2000. PC-BC: A Block Cave Design and Draw Control System. *Proceedings MassMin 2000.* 469–484.

Diering, T. 2006. Personal communication.

Garey, M.R., & Johnson, D.S. 1979. Computers and intractability: a guide to the theory of NP-completeness. *W.S. Freeman..*

GEMCOM. 2006. Company website. http://www.gemcomsoftware.com/

Gertsch, R. E. & Bullock, R.L. 1998. Techniques in Underground Mining. *Society for Mining, Metallurgy, and Exploration (U.S.)*

Giannini, L. M., Caccetta, L., Kelsey, P., & Carras, S. 1991. PITOPTIM: A New High Speed Network Flow Technique for Optimum Pit Design Facilitating Rapid Sensitivity Analysis. *AusIMM Proc.* **2** 57–62.

Howard, R. 1960. Dynamic Programming and Markov Processes. *(M.I.T. Press).*

Lee, J., Leung, J. & Margot, F. 2003. Min-up / Min-down Polytopes. *Tepper School of Business* Paper 272, http://repository.cmu.edu/tepper/272

Lerchs, H. & Grossman, I. F. 1965. Optimum Design for Open Pit Mines. *CIM Bulletin.* **58** (January) 47–54.

Lübbecke, M & Desrosiers, J. 2005. Selected Topics in Column Generation. *Operations Research [serial on the Internet].* **53** no. 6, 1007-1023.

Malkin, P. & Wolsey, L. 2006. Possible Formulations of the Mining Problem. *Manuscript.*

Malkin, P. & Wolsey, L. 2006. Personal Correspondence.

Marriott, K. & Stuckey, P. J. 1988. Programming with Constraints: An Introduction. *MIT Press, Cambridge MA*

Newman, A., Rubio, E., Weintraub, A. & Eurek, K. 2010. A Review of Operations Research in Mine Planning. *Interfaces* **40** no. 3, 222-245.

Newman, P. 1987."Indirect Utility Function" The New Palgrave:A Dictionary of Economics. Eds John Eatwell, Murray Milgate and Peter Newman. Palgrave Macmillan. 1987.

Parkinson, A., McCormick, S.T. 2005. Optimal Replenishment with Two Delivery Sizes. *(MSOM Conference Presentation).*

Pinedo, M. 1995. Scheduling: Theory, Algorithms, and Systems. *Prentice Hall..*

Puterman, M. 1994. Markov Decision Processes. *(John Wiley and Sons).*

Rajan, D. & Takriti, S. 2005. Minimum Up/Down Polytopes of the Unit Commitment Problem with Start-Up Costs. *IBM Research Report*

Resolution Coper Mining. (2009). Block Cave Mining. *Block Cave Mining.* Retrieved November 12, 2009, from http://www.resolutioncopper.com/res/ourapproach/BlockCaveMining.pdf

Rubio, E. 2002. Long Term Planning of Block Cave Operations using Mathematical Programming Tools. Masters Thesis, UBC.

Van Vyve, M. 2006. Personal communication.

Veinott, A. 1965. The Optimal Inventory Policy for Batch Ordering. *Operations Research.* **13** 1103–1145.

Weintraub, A., Pereira, M. & Schultz, X. 2008. A Priori and A Posteriori Aggregation Procedures to Reduce Model Size in MIP Mine Planning Models. *Electronic Notes Discrete Math* **30** 297- 302.

Weise, T. 2008. Global Optimization Algorithms - Theory and Applications -. http://www.it-weise.de.

Wolsey, L. 1998. Integer Programming. *John Wiley and Sons..*

Yegulalp, T. M., & Arias, A. J. 1992. A fast algorithm to solve the ultimate pit limit problem. *Proc. 23rd International APCOM Symposium* 391–397.