Dig-limit optimization

Mixed integer linear programming for grade control in open pit mining

J.W. Buist

Civil engineering and geosciences

TU Delft

Dig-limit optimization

Mixed integer linear programming for grade control in open pit mining

by

J.W. Buist

to obtain the degree of Bachelor of Science
at the Delft University of Technology,

Student number: 4274849
Project duration: April 23, 2018 – June 25, 2018
Thesis supervisors: Dr. ir. T. Wambeke, TU Delft
Ir. J. R. van Duijvenbode, TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Abstract

In open-pit mining on a bench level, the dig-limit optimization problem is deciding whether to classify a Selective Mining Unit (SMU) as waste or whether to classify it as ore. SMU's with high ore grades are profitable for the mining operation, but due to equipment size a whole frame of SMU's has to be classified as either ore or waste. In this thesis the dig-limit optimization problem will be solved using Mixed Integer Linear Programming (MILP). The model proposed in this thesis takes the *frame constraint* and the *average grade constraint* into account. The main results of the MILP model are in table 1. The results show that MILP can successfully be used to determine the optimal value of a mining operation, however the determination of the dig-limit for larger grids leads to exponential computation time.

Grid size	**Frame size**	**Runtime**	**Objective value**	**No. of constraints**	**No. of variables**
13x15	2x2	1.35 s	915.18	976	3315
26x30	4x4	1239.20 s	3741.85	13261	50700

Table 1: Main results for 13x15 and 26x30 grid

Keywords: open-pit mining; grade control; dig-limit optimization; mixed linear integer programming; selective mining unit.

Contents

List of Figures

List of Tables

List of Acronyms

ILP Integer Linear Programming. vii, 3–5

MILP Mixed Integer Linear Programming. iii, 2, 3, 5, 7, 8, 13, 21, 22

SMU Selective Mining Unit. iii, 1, 7–10, 15, 20

1 Introduction

The subject of this thesis will be introduced in this first chapter. First the problems with grade control in mining will be described. After this, the purpose of dig-limit optimization will be described. Then a brief introduction in optimization problems will be given along with methods to solve these problems. At the end of this chapter, the goals and objectives of this thesis will be given.

1.1. Grade control in mining

In mining operations the grade is defined as the proportion of the product wanted (for example gold) inside the ore. A location with relatively high grades is therefore a profitable place for a mine. On a smaller scale grades also differ. Even inside a mine there is a difference between the grades. For the mining operation it is useful to keep the grades as high as possible, because that is the most profitable. [1]

1.2. Dig-Limit optimization

Dig-limits are determined in most open-pit mines on a bench level. [2] Dig-limits classify the material to be mined between higher graded ore and lower graded waste. Ore can be send to the processing plant and waste to the dump. These dig-limits are used to keep the grade in the processing plant high enough to reach production targets and to maximize the profit of the mine. This all depends on the cut-off grade. [3] When the grades in the processing plant are too low (below the cut-off grade) the cost of processing will be too high too make a profit. In this case it is better to dump the low grade material. This is usually cheaper than processing it. Therefore the dig-limits must be chosen carefully. [4]

The dig-limits do not only depend on the cut-off grade. Many more factors have to be kept in mind. The most important constraint to the dig-limit is the mining equipment. On bench scale, the grades are known for each Selective Mining Unit (SMU), but the equipment can only mine accurately a frame of SMU's. In this thesis this will be called the *frame constraint*. [5]

Another constraint to the dig-limit is the mineral processing. For this to be feasible a certain minimal average grade is needed. Therefore the material sent to the processing plant has to be chosen carefully. In this thesis this will be called the *average grade constraint*. This can be assumed as it is reasonable that one bench is mined in a short period of time and will determine the average grade of the processing. [6]

In past days, most of these dig-limits were drawn by hand, by a geologist. Drawing of a dig-limit can be handled as a mathematical optimization problem. The optimal dig-limit will be the dig-limit with the most profit for the mine. This is a complex optimization problem and advanced solving techniques have to be used to solve it. [4]

1.3. Optimization problems and methods to solve them

An optimization problem is the problem of finding the best solution out of all the feasible solutions. The best solution is most often the maximum or either the minimum solution depending on the problem. Common applications are scheduling problems, mixing problems or other problems with a lot of solutions. [7]

The easiest way to solve an optimization problem is to calculate all the solutions and pick the highest or lowest one. However, this becomes very hard when there are many variables and it is really time inten-

sive. Therefore a lot of methods were developed to solve optimization problems. These methods can be divided between optimization algorithms, iterative methods and heuristics. The optimization algorithms are designed mainly for linear and quadratic programming, for example the Simplex Algorithm designed by George Dantzig. [8] The iterative methods are used to solve non-linear problems. Common examples are Newton's methods and the finite difference method. [9] A heuristic is a technique to solve a difficult problem faster, which trades optimality, completeness, accuracy, or precision for speed. [10] Examples of heuristics are genetic algorithms or simulated annealing. [11]

1.4. Goals and objectives

The main goal of this research is to determine the optimal dig-limit on a bench scale with the use of mathematical optimization. The optimal dig-limit is the dig-limit for which the mine will make the most profit. The dig-limit will be limited by two constraints: the frame constraint and the average grade constraint.

As stated before there are many methods to solve an optimization problem. This thesis will limit itself to the use of Mixed Integer Linear Programming (MILP) to calculate the dig-limit optimization.

1.4.1. Research questions

The main question that this thesis will try to answer is:

> Can the method of mixed integer linear programming be used for dig-limit optimization, grade control and therefore profit maximization in a mine on a bench scale?

Next to the main question this thesis will also try to answer the following sub-questions:

1. Suppose the frame constraint can be implemented to the MILP for maximizing profit. How do the results change when the frame size and shape are changed?
2. Presume the average grade constraint can be added to the model. What is the influence of the average grade constraint on the dig-limit optimization program?
3. How many variables and constraints are needed to successfully implement the constraints into the model?
4. How long does it take to calculate the dig-limit optimization with MILP?

1.4.2. Outline

In this chapter, the problem of dig-limit optimization was introduced. Before the dig-limit optimization model can be created, more research on MILP is needed. Therefore, chapter 2 will focus on the method of MILP. When the MILP method has been made clear, the methodology of the model will be given in chapter 3. This chapter will describe all the maths needed to come up with the dig-limit optimization model. After this, in chapter 4, the results of the dig-limit optimization model will be given. Finally, chapter 5 will conclude this thesis, discuss the results and give answers to the questions posed in section 1.4.1.

2

Literature review

In this chapter the method of Mixed Integer Linear Programming will be explained. This is the method that will be used to solve the dig-limit optimization problem. The dig-limit optimization problem is explained in chapter 1.

2.1. Mixed Integer Linear Programming

To find the optimal solution for an optimization problem, Mixed Integer Linear Programming (MILP) can be used. This is a form of integer programming, in which not all the variables are constrained to be integers. Integer programming is mainly applied for problems where the variables can only represent integers or where the variables represent a decision. For example it is not possible to schedule 1.3 taxis to pick-up people and the decision can be to send taxis or not to send taxis. More detailed examples will be given in this chapter. [8]

2.1.1. Definition

An Integer Linear Programming (ILP) is expressed as: [12]

$$\begin{aligned} Minimize/Maximize: \quad & \mathbf{c}^T\mathbf{x}, \\ Subject \quad to: \quad & A\mathbf{x}+\mathbf{s}=\mathbf{b}, \\ & \mathbf{s}\geq 0, \\ & \mathbf{x}\geq 0, \\ & \mathbf{x} \quad is \quad in \quad \mathbb{Z}^n. \end{aligned}$$

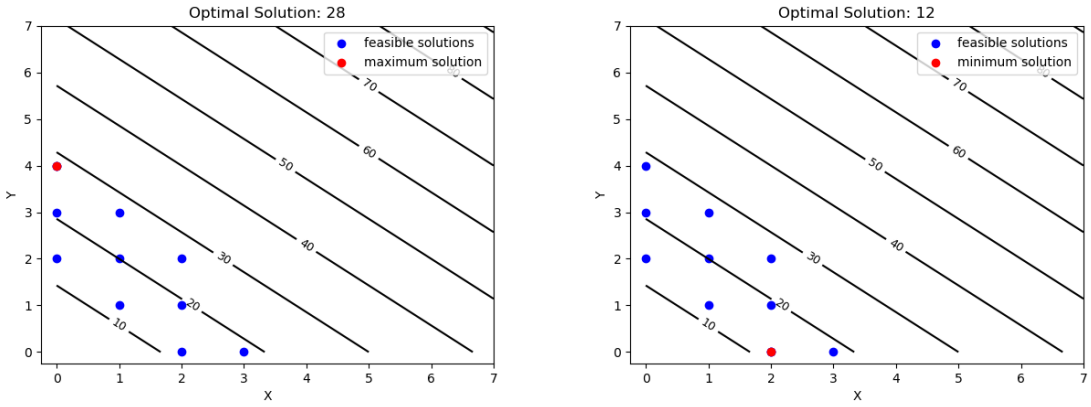
In this equation the first line is called the linear objective function. This can be a maximization or minimization depending on the problem. This objective function is subject to one or more constraints. The goal of this problem is finding the best integer value of X for which the objective function is maximized. When X can also take non-integer values this becomes a MILP problem. [8]

2.1.2. Simple Example

An easy example in the standard ILP form for a optimization problem is:

$$\begin{aligned} Maximize: \quad & Z=6X+7Y, \\ Subject \quad to: \quad & 4X+5Y\leq 20, \\ & 10X+7Y\leq 35, \\ & 3X+4Y\geq 6, \\ & X,Y\geq 0, \\ & X,Y\in\mathbb{Z} \end{aligned}$$

In this case, the objective function Z is to be maximized under multiple constraints and X and Y are the integers. The objective function Z can be viewed as a contour map for integer coordinates X and Y. The



(a) Maximization problem

(b) Minimization problem

Figure 2.1: The feasible solutions of an simple example of an ILP problem

height of a certain coordinate is then the value of the objective function. Besides being integer, X and Y are bound to three other constraints in this case. The feasible solutions of this maximization problem are scattered in figure 2.1a together with the contour map of the objective function. From all these solutions the highlighted solution in ($X = 0, \quad Y = 4$) is the highest. Therefore this is is the best integer solution to this problem.

The same problem can be handled as a minimization problem. This gives the following output shown in figure 2.1b. Regarding the maximization problem, the feasible solutions are the same, as the constraints are the same. In this case the best solution is the lowest solution. This solution is located at ($X = 2, \quad Y = 0$).

2.1.3. The Bank Robber

To explain the use of ILP better a problem with some more context is needed. Suppose a criminal is robbing a bank. When he enters the vault, he has to make a choice. The vault is filled with gold bars and stacks of 50 euro bills. The gold bars will yield a lot of money on the black market, but they are heavy. The banknotes are light, but not as valuable as the gold bars. On top of that he has to put the loot in a suitcase he brought to escape the bank without being noticed. He has only five minutes in the vault and it takes longer to put a gold bar in his suitcase than a stack of bills. Will he choose to only take gold, only take banknotes or will he choose a combination?

Symbol	Value	Meaning
N_b	≥ 0	Number of banknotes
N_g	≥ 0	Number of gold bars
P_b	€5.000,–	Price of stack of bills
P_g	€30.000,–	Price of gold bar
V_b	$1.05 * 10^{-5} \quad m^3$	Volume of stack of bills
V_g	$6.86 * 10^{-5} \quad m^3$	Volume of gold bar
W_b	$0.80 * 10^{-3} \quad kg$	Weight of stack of bills
W_g	$1 \quad kg$	Weight of gold bar
T_b	$2 \quad s$	Time to pack stack
T_g	$10 \quad s$	Time to pack gold bar
$V_{suitcase}$	$4.81 * 10^{-2} \quad m^3$	Volume suitcase
$W_{maximum}$	$23 \quad kg$	Maximum weight suitcase
$T_{maximum}$	$300 \quad s$	Maximum time

Table 2.1: Parameters bank robber problem

This can be formulated as an integer problem. The integers that can be varied are the number of bills and the number of gold bars. The objective function of the criminal is clear: he wants to maximize his profit. His profit is the sum of the bills and the gold bars he takes from the vault. In his work the criminal is constrained

by his suitcase. He can not take more from the vault than fits in the suitcase. He is also constrained by his own strength. He can not put more weight in his suitcase than he can carry. At last he is constrained by the time, because he only has five minutes to fill his suitcase.

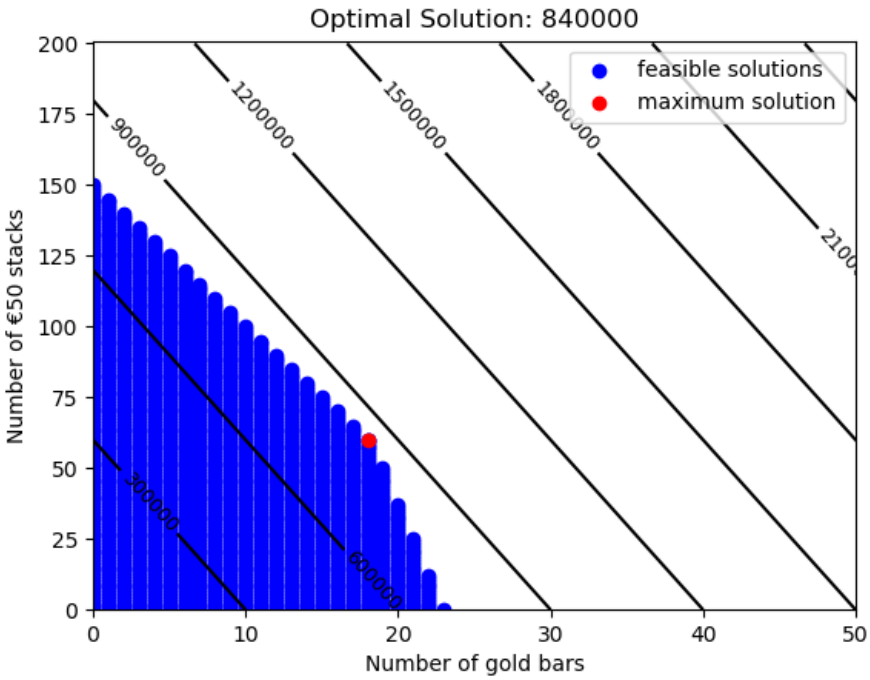


Figure 2.2: Feasible solutions pool of the bank robber problem

This problem is mathematically stated as:

$$\begin{aligned} Maximize: \quad & profit = N_b P_b + N_g P_g, \\ Subject \quad to: \quad & N_b V_b + N_g V_g \leq V_{suitcase}, \\ & N_b W_b + N_g W_g \leq W_{maximum}, \\ & N_b T_b + N_g T_g \leq T_{maximum}, \\ & N_b, N_g \geq 0, \\ & N_b, N_g \in \mathbb{Z} \end{aligned}$$

The explanation of the symbols in these equations is given in table 2.1. For the problem is assumed that there are only stacks with 100 banknotes of €50, so-called €5000 straps and 1 kilogram gold bars.

The solution of this problem is visualized in figure 2.2. Again, the objective function is plotted as a height map. All the feasible solutions are shown and the optimal solution is shown. The optimal solution occurs when the thief takes 18 gold bars and 60 €5000 straps. In this case, the profit of the robber is €840.000. This ILP problem would convert to a MILP problem if one of the variables was not constrained to be integer. When the criminal would bring a saw to the vault and cut the gold bars, he would not be constrained any more to take integer values of gold bars, but he could take any real number of gold bars. For example 2,4 gold bars. This would change the problem into a MILP problem. [12]

2.2. Solving techniques

The problems given as example were not so difficult. It did not cost much time to calculate all the feasible solutions. They could be solved visual and had only a few constraints. The dig-limit optimization problem will have a lot of constraints and may not be so easy to solve. Luckily there are other techniques than just calculate all the feasible solutions and pick the best one.

MILP problems are most of the time solved using a branch-and-bound algorithm. [13]. This is a tree based search algorithm for the best solution. It considers the solution pool as a rooted three with all the solutions at

the root. The algorithm iterates through branches of the three and checks them against the upper bound and lower bound of the optimal solution. The branch is discarded if it is outside of the bounds and the branch is further explored if it is inside the bounds.[14] The branch and bound method was first proposed in 1960 and has been used since. [15]

3

Methods

This chapter focuses on the dig-limit optimization model proposed in this thesis. This Mixed Integer Linear Programming model will solve the problem introduced in chapter 1 with the MILP techniques explained in chapter 2. This chapter will introduce the model situation and all the relevant parameters, indices, sets and variables. These will be used to create the objective function and the constraints.

3.1. Model situation

The main thing that is required as input for this model is a grade distribution. All the other variables are constant, only the grades depend on location. The model will convert the grade distribution to a dig-limit. The Walter lake dataset is used as grade distribution. [16]

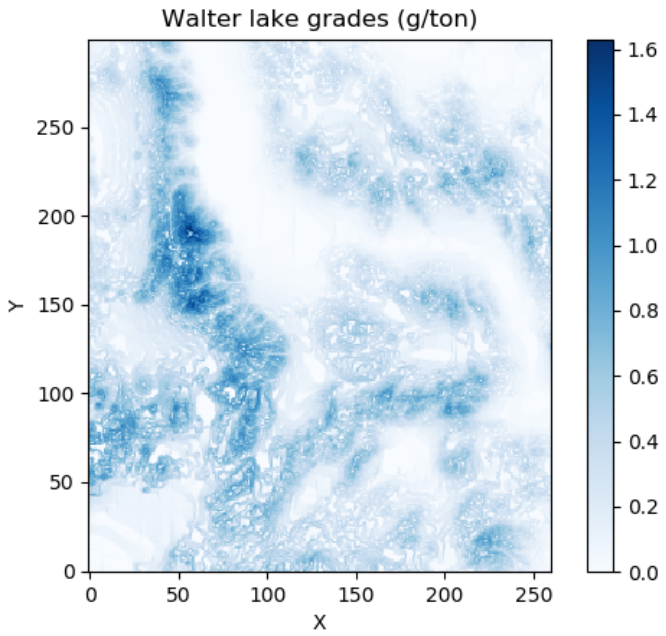


Figure 3.1: Grades in Walter lake dataset [16]

This dataset consists of a rectangular grid of 260 by 300 blocks with elevation data. When divided by thousand the elevation values become realistic grades for an open pit gold mine. [17] This dataset is often used in the field of geostatistics and mining. Therefore it also applicable for this thesis [16]. Each grid point is than a SMU. The resulting values are plotted in figure 3.1.

Using MILP on a 260 by 300 blocks dataset is very time intensive and therefore the Walter lake dataset was reblocked to a 26 by 30 blocks dataset which holds for every block a mean value of 10 by 10 blocks from the original dataset and to a 13 by 15 blocks dataset which holds for every block a mean value of 20 by 20 blocks from the original dataset. The reblocked datasets are shown in figure A.1 and figure A.2 in appendix A.

3.2. Parameters

Next to the grade distribution, more input parameters are required. These are of simpler form (single value) but just as important. All these parameter are stated in table 3.1. X, Y, N_x and N_y depend on the SMU size and the frame size.

Symbol	Units	Meaning
X	$\in \mathbb{Z}^+$	Amount of SMU's in X direction
Y	$\in \mathbb{Z}^+$	Amount of SMU's in Y direction
N_x	$\in \mathbb{Z}^+$	Frame size x-axis
N_y	$\in \mathbb{Z}^+$	Frame size y-axis
M	ton	Mass of SMU
$G_{i,j}$	g/ton	SMU grade at (i,j)
$G_{average}$	g/ton	Average grade demanded by processing plant
P	$€/ton$	Price
R	%	Recovery
C_p	$€/ton$	Processing cost
C_m	$€/ton$	Mining cost

Table 3.1: Model parameters

3.3. Indices and sets

Indices and sets are used to state the summations in the objective function and in constraints. Indices i and j form the corresponding sets which span up the grid of SMU's. Secondly, indices fx and fy for the sets which span up each possible frame for each SMU. At last, indices α and β form an offset index for each frame. These indices and corresponding sets are given in table 3.2. [18]

Symbol	Range	Meaning
i	$0 : i : X$	SMU index x-axis
j	$0 : j : Y$	SMU index y-axis
fx	$0 : fx : N_x$	Frame index x-axis
fy	$0 : fy : N_y$	Frame index y-axis
α	$0 : \alpha : N_x$	Frame index offset x-axis
β	$0 : \beta : N_y$	Frame index offset y-axis

Table 3.2: Model indices

3.4. Decision variables

Symbol	Size	Variable type	Meaning
$x_{i,j}$	$X * Y$	Binary	Ore/waste classification
$t_{i,j,fx,fy}$	$X * Y * N_x * N_y$	Integer	Sum of x values inside a frame
$a_{i,j,fx,fy}$	$X * Y * N_x * N_y$	Binary	1 if valid waste frame
$b_{i,j,fx,fy}$	$X * Y * N_x * N_y$	Binary	1 if valid ore frame
$v_{i,j,fx,fy}$	$X * Y * N_x * N_y$	Binary	Valid frame classification

Table 3.3: Decision variables

Ultimately, the goal of the linear program is to determine the optimal ore/waste classification. Therefore the linear program will have to be able to change this variable. Next to the ore/waste classification there are more variables of which the value needs to be determined by the program. These variables are called *decision variables*. The decision variables required for the model are given in table 3.3.

The most important variable is the ore/waste classification ($x_{i,j}$), as this is the goal of the model. The other two decisions variables are $t_{i,j,fx,fy}$ and $v_{i,j,fx,fy}$, where $t_{i,j,fx,fy}$ is the sum of the values of $x_{i,j}$ for each possible frame and where $v_{i,j,fx,fy}$ takes a positive value if the value of $t_{i,j,fx,fy}$ is valid. This will be explained further in section 3.6.1

3.5. Objective function

As mentioned before, the goal of the model is to determine the optimal dig-limit. This is the dig-limit for which the mine will make the most profit by sending the material to the appropriate location (waste dump or processing plant). Therefore the objective function will have to be formulated as a maximization of the profit.

The total profit can be divided between costs and benefits. The sum of these two is the profit. In the considered mining operation there are two main costs: the mining costs and the processing costs. The mining costs apply to all SMU's as the whole bench needs to be mined. The processing costs, however, only apply to the blocks that are chosen by the dig-limit to be processed.

$$\begin{aligned} &maximize: \\ &profit = \sum_{i}^{X}\sum_{j}^{Y} M(x_{i,j} * G_{i,j} * P * R - x_{i,j} * C_p - C_m) \end{aligned} \tag{3.1}$$

The only benefit of the mining operation is the sold ore. Thus the processed blocks times the price, the recovery and the grade.

The costs and benefits combined lead to the equation for the profit. The maximization of this equation will be the objective function of the model. This is stated in equation 3.1. [4]

3.6. Constraints

As stated in the introduction, the objective function will be limited by two constraints. The *frame constraint* and the *average grade constraint*. This section will explain how these constraints work and how they will be modelled.

3.6.1. Frame constraint

The frame constraint is the most complicated constraint that will be used in the model. A so-called frame is the minimal range of the mining equipment. When the grade distribution is known in a higher resolution then the frames are, this limits the dig-limit to bigger blocks then the SMU size. When the frames are three by three in SMU size this already gives nine different possible frames for each SMU. An example of three out of the nine possible frames is given in figure 3.2. The yellow square is the SMU where the three frames apply to.

In the model, the decision variable $t_{i,j,fx,fy}$ holds for each frame on each SMU the sum of $x_{i,j}$ values. This is mathematically expressed in equation 3.2. [4] The if statement is needed to eliminate the frames crossing the border of the grid. For the border crossing frames $t_{i,j,fx,fy}$ is given the value of 1.

$$\begin{aligned} if \quad & i - fx + N_x \leq X, \quad j - fx + N_y \leq Y, \quad i - fx \geq 0, \quad j - fy \geq 0: \\ & t_{i,j,fx,fy} = \sum_{\alpha}^{N_x}\sum_{\beta}^{N_y} x_{i-fx+\alpha, j-fy+\beta} \\ else \quad & : \\ & t_{i,j,fx,fy} = 1 \end{aligned} \tag{3.2}$$

The values for $t_{i,j,fx,fy}$ are in the range from 0 to $N_x * N_y$. The frame constraint demands the frames to be entirely sent to the waste dump or to be completely sent to the processing plant. Therefore, only the frames with value 0 (only waste) or value $N_x * N_y$ (only ore) have to be taken in account. The border crossing frames will automatically be considered as an invalid frame as there value is always 1.

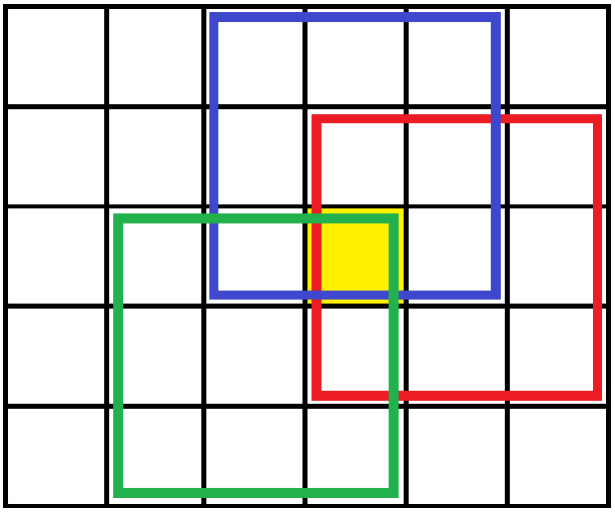


Figure 3.2: A SMU with three of the nine three by three frames displayed

To accomplish this two new decision variables are introduced: $a_{i,j,fx,fy}$ and $b_{i,j,fx,fy}$. They respectively check if a frame is only waste or only ore. The statements creating these are given in equation 3.3 and equation 3.4.

$$\begin{aligned} if \quad & t_{i,j,fx,fy} = 0 : \\ & a_{i,j,fx,fy} = 1 \end{aligned} \tag{3.3}$$

$$\begin{aligned} if \quad & t_{i,j,fx,fy} = N_x * N_y : \\ & b_{i,j,fx,fy} = 1 \end{aligned} \tag{3.4}$$

Then a frame is converted to a valid frame if either $a_{i,j,fx,fy}$ is valid or if $b_{i,j,fx,fy}$ is valid. This is done in equation 3.5.

$$v_{i,j,fx,fy} = a_{i,j,fx,fy} \quad or \quad b_{i,j,fx,fy} \tag{3.5}$$

In the final solution only valid frames can be used. Therefore each SMU has to have a valid frame. This is applied to the model by equation 3.6. [4]

$$\sum_{fx}^{N_x} \sum_{fy}^{N_y} v_{i,j,fx,fy} \geq 1 \tag{3.6}$$

3.6.2. Average grade constraint

The other constraint to the model is the average grade constraint. This constraint demands all the material classified as ore to have a minimal average grade. This is often a demand by the processing plant in order to efficiently run the processes.

This constraint is added to the model by calculating the average grade and demanding it to be higher than a certain average grade. This is stated in equation 3.7

$$\frac{\sum_{i}^{X} \sum_{j}^{Y} x_{i,j} * G_{i,j}}{\sum_{i}^{X} \sum_{j}^{Y} x_{i,j}} \geq G_{average} \tag{3.7}$$

3.7. Solver

In order to be able to solve this model quickly, a solver is used. The solver used in this thesis is Gurobi. This solver uses all the techniques explained in section 2.2. Gurobi offers one of the fastest solvers available. [19] Gurobi is a commercial optimization solver which also provides academic licences. [20]

The Gurobi Python interface was used to solve the model. The written program can roughly be divided in three parts that build the model in Gurobi. First, declaring of the Gurobi variables, second, the objective function and third, implementation of the constraints.

Declaring the Gurobi variables is done by programming the decision variables from 3.3 into the Gurobi syntax. The variables are all named so they can be recalled from the model. The objective function is written analogous to equation 3.1. The double summation is replaced by Gurobi's `quicksum()` function which is a faster alternative to normal summations. [21] The same `quicksum()` function is used to implement the average grade constraint. Equation 3.7 could be rewritten to a single line of code.

Declaring the variables, generating the objective function and implementing the average grade constraint is quite simple, but implementing the frame constraint needs some more code. The frame constraint from equation 3.2 is defined for each i, j, fx and fy. Therefore a nested loop is needed to implement this. The following algorithm implements the frame constraint analogous to the maths in section 3.6.1:

for all SMU's **do**
  for all Possible frames in SMU **do**
    if Frame is not crossing borders **then**
      t is the sum of x values inside a frame
    else
      t is a invalid frame
    end if
    if t is a valid frame **then**
      v is also valid
    end if
  end for
  There must be at least one valid frame for each SMU
end for

A full runnable code for a $13x15$ grid with frame constraint and average grade constraint implemented is given in appendix B. This code should work with the right packages installed. Important parts of the code are highlighted with comments.

4

Results

This chapter will show the results of the Mixed Integer Linear Programming model. The model will give solutions to the dig-limit optimization problem introduced in chapter 1. The model will use the technique of MILP which is explained in chapter 2. The content of this model is defined in chapter 3.

In this chapter, first, the results of the MILP with the frame constraint will be discussed. Next, the same situation will be reviewed with intermediate solutions from the solver. After this, the results of changing the frame size and shape will be shown. Hereafter, the average grade constraint will also be added to the model which gives new results. Next, the final results will be compared with a hand solution. The next section will focus on the computation time of the MILP model and will also give insights on the number of constraints and variables. Finally, a sensitivity analysis on the model will be described.

4.1. Maximize profit with MILP and frame constraint

This section will show the results of the MILP model in combination with the frame constraint. The parameter values had to be determined first before the model could be run. The chosen values are tabulated in table 4.1. These values were not chosen for being as realistic as possible but were chosen to create the need for a dig-limit. For example, when the price is chosen really high, the model chooses to classify all the blocks as ore. On the other side, the model chooses to classify all the blocks as waste if the processing costs are too high. They were chosen, thus, to be applicable to the Walter Lake dataset. [16]

Symbol	Value	Units
M	1	ton
P	80	€$/ton$
R	100	%
C_p	15	€$/ton$
C_m	5	€$/ton$

Table 4.1: Parameter values

The model was run on two different downsized versions of the Walter Lake dataset. One with the new shape of $13x15$ and the other with the new shape of $26x30$. The values of the downsized grids were calculated as the mean of the original values in the range of a single new block. A frame size of $2x2$ was used for the $13x15$ model and a frame size of $4x4$ was used for the $26x30$ model.

The results of these dig-limit optimizations are plotted in figure 4.1 and figure 4.2. For both results the left plot displays the grade distribution of the reblocked Walter Lake dataset. The right plot shows the dig-limit, where the blue blocks mark the material as ore and the white blocks mark the material as waste. Both result look similar, but the higher resolution grid provides a little more detailed result.

The final value of the objective function was also determined for these grids. The $13x15$ grid has an objective of €**915**,**18** and the $26x30$ grid an objective of €**3741**,**85**. The parameter values from table 4.1 were used for both grids. The objective value of the $13x16$ grid is around 4 times smaller than the objective value of the $26x30$ grid. When the objective of the smaller grid is multiplied by 4 this gives an objective value of €**3660**,**72**.

This is slightly less than the result of the objective value of the 26x30 grid. It turns out a double resolution does not add very much detail to the solution, but the bigger grid can make better boundary decisions and therefore more profit.

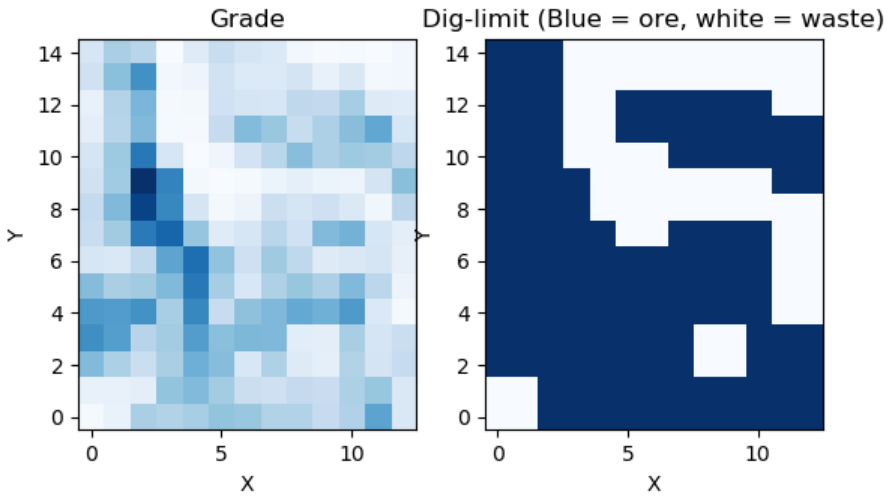


Figure 4.1: Dig-limit for 13x15 grid with 2x2 frame

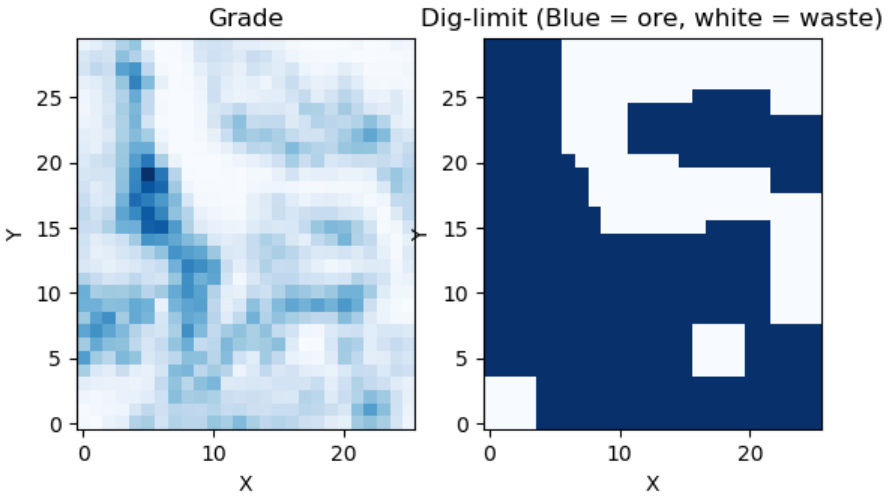


Figure 4.2: Dig-limit for 26x30 grid with 4x4 frame

Note: Most of the other calculations in this chapter will be done on these two grids. They can be considered the standard or reference situations.

4.2. Intermediate solutions

To get an idea how the Gurobi solver works to the final solution given in section 4.1, intermediate solutions were plotted. These can be found in figure 4.3. The variable values were the same as in section 4.1 and can be found in table 4.1. The plots show the feasible solutions found by the solver until it reaches the optimal solution. This was calculated for the 13x16 grid with 2x2 frames. It was also calculated for the 26x30 grid with the 4x4 frames. The intermediate solutions of the bigger grid can be found in figure A.4 in appendix A. The

runtime required to create the intermediate solutions is also noted there. The runtime will be discussed in section 4.6.

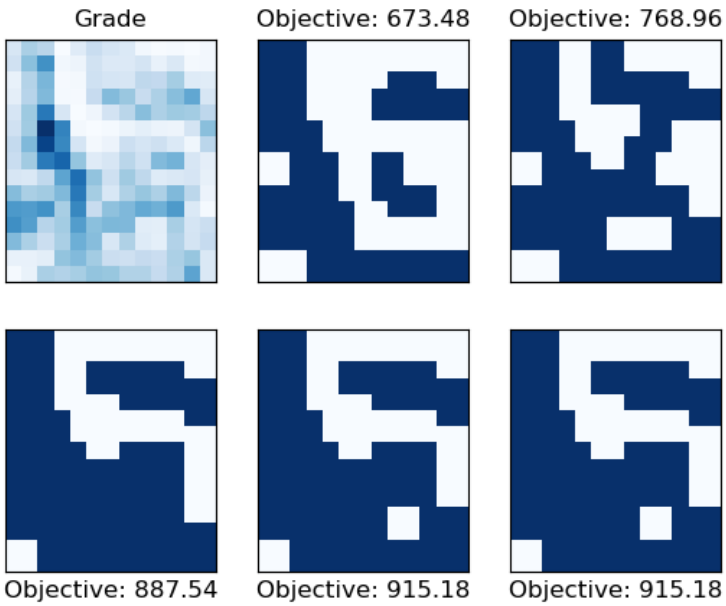


Figure 4.3: Intermediate solutions for 13x15 grid with 2x2 frame

From figure A.4 in appendix A can be concluded the solver's branch and bound techniques get quickly in range of the optimal solution, but it takes a lot of steps (and time) to find the optimal solution. The optimal solution for the smaller grid is quickly found and there are not many feasible solutions found in between.

4.3. Changing the frame size and shape

In the previous sections the frames were all square and modestly sized. This section will show the results of changing the size and the shape of the frame. All the frame sizes are again measured in amount of SMU's. Results of these different frames are in table 4.2.

Grid	Frame size	Frame area	Objective value
13x15	2x2	4	€ 915.18
	3x3	9	€ 717.10
	4x4	16	€ 576.98
	2x4	8	€ 698.89
	2x8	16	€ 522.86
	4x2	8	€ 682.15
	8x2	16	€ 436.47
26x30	4x4	16	€ 3741.85
	5x5	25	€ 3199.03
	6x6	36	€ 2479.59
	4x8	32	€ 2814.22
	4x16	64	€ 1745.86
	8x4	32	€ 2663.64
	16x4	64	€ 1745.86

Table 4.2: Objective values for different frame sizes and shapes

The results of the different frame sizes and frame shapes follow a logical trend. The bigger the frame, the lower the objective value. When the frames get bigger, the model has to select low grade blocks more

often and thus the objective value gets lower. Another conclusion from these results is that when rectangular frames are used it is more profitable to use the long side of the frame in the Y direction. The results for the $26x30$ grid in combination with the $4x16$ and $16x4$ frames are the same, this is because they both select the whole grid as ore.

4.4. Maximize profit with MILP, average grade constraint and frame constraint

The other constraint that will be discussed in this thesis was the average grade constraint. To create interesting solutions the model was run with the frame constraint and the average grade constraint. Again it was done with the parameter values from table 4.1. It was done for five different situations with increasing average grade demand from the processing plant. The values for these situations are tabulated in table 4.3. This values are all a certain percentage above the mean grade from the grade distribution of the input grid. This was all done for the $13x15$ grid and the $26x30$ grid with square frames of respectively $2x2$ and $4x4$ just as the frames in section 4.1.

No.	$G_{average}$	% above mean grade
#1	$0.347 \quad g/ton$	25%
#2	$0.417 \quad g/ton$	50%
#3	$0.486 \quad g/ton$	75%
#4	$0.556 \quad g/ton$	100%
#5	$0.625 \quad g/ton$	125%

Table 4.3: Average grade situations

The results from this test are plotted in figure 4.4 and figure 4.5. Visually, the results are not much different. Just as in section 4.1 the dig-limits look the same but are more detailed for the bigger grid.

No.	Obj. 13x15 grid	Obj. 13x15 grid x 4	Obj. 26x30 grid	Absolute difference
#1	€ 915.18	€ 3660.72	€ 3616.58	€ 44.14
#2	€ 699.98	€ 2799.92	€ 2970.70	€ 170.78
#3	€ 297.28	€ 1189.12	€ 1608.55	€ 419.43
#4	€ −27.53	€ −110.12	€ −123.10	€ 12.98
#5	€ −306.02	€ −1224.08	€ −920.66	€ 303.42

Table 4.4: Average grade tests objectives

The objective values are displayed in table 4.4. The objective value of the $13x15$ grid multiplied by four was also calculated. This can be found in the same table. For two of the tests the objective values of the $13x15$ grid multiplied by four are almost the same as the objective values of the $26x30$ grid. Two other tests give big differences and the other test lies in between. The figures corresponding to the tests with big differences also show more differences than the tests with comparable values.

From the situations with higher average grade values, the more valuable frames can be indicated clearly. When the average grade value gets too high to have a profitable mining operation, it can be observed the model still classifies material as ore. This happens because the model tries to limit the losses. The average grade constraint can be used to find the most valuable frames.

4.5. Comparison with hand solution

The dig-limits are often still chosen manually in mines. A student in geology was asked to draw a dig-limit for the same dataset used in this thesis. This is shown in figure 4.6 together with the solution of the $26x30$ grid. The geology student did not know the exact values that are used in the objective function, but did manage to draw a proper dig-limit. Before he draw the dig-limit by hand, the solution generated by the model was unknown to him.

The drawing follows roughly the same lines the generated solution follows. Easy to classify are the big waste areas in the top and the middle of the grid and the square waste area in the bottom left of the grid. Hard

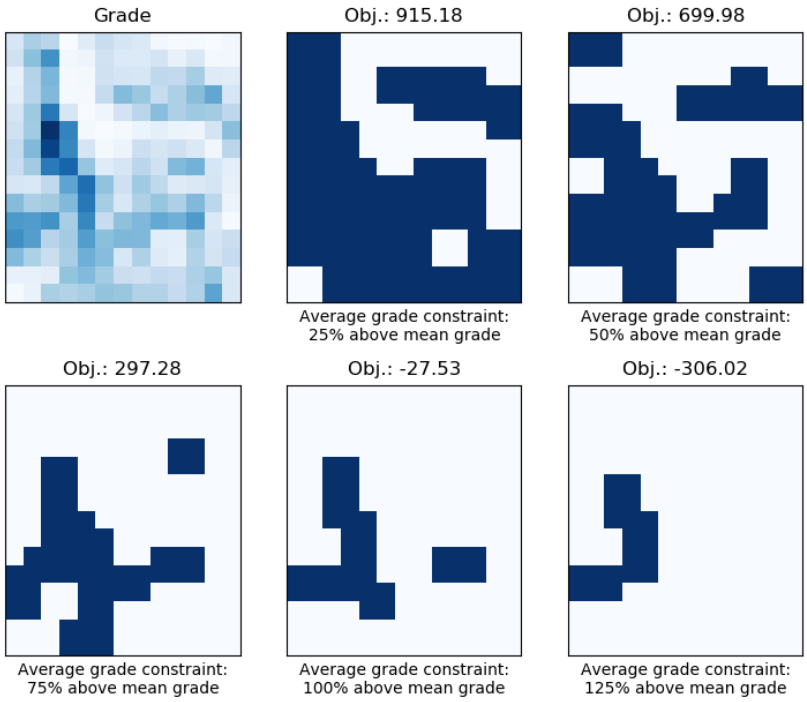


Figure 4.4: Average grade constraint for 13x15 grid with 2x2 frame

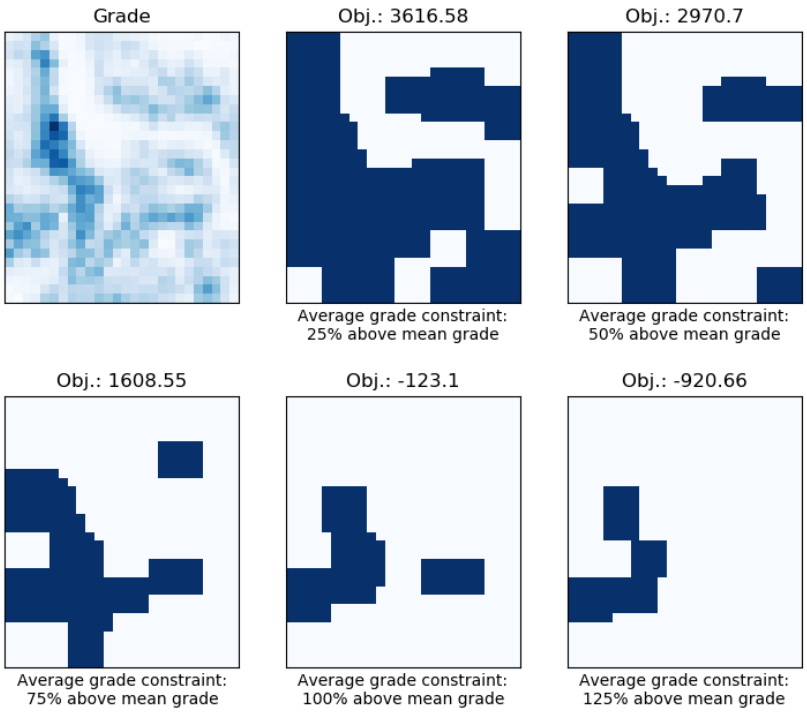


Figure 4.5: Average grade constraint for 26x30 grid with 4x4 frame

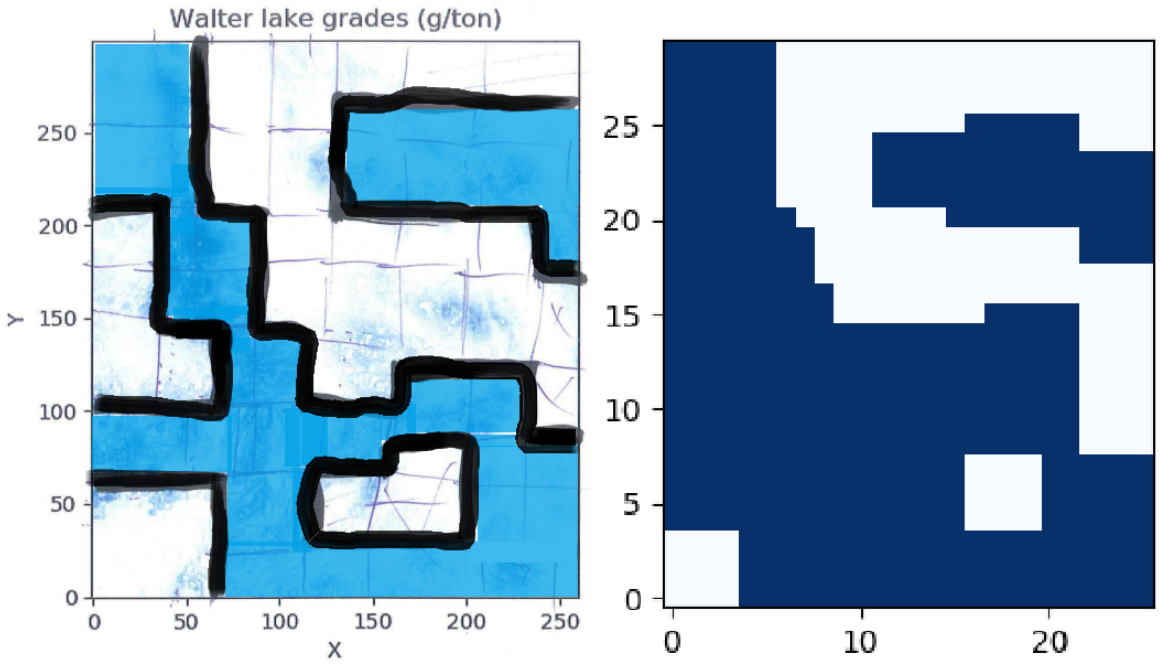


Figure 4.6: Hand solution and computer generated solution

to classify areas are the areas with medium grades. The decision to take it as ore or as waste is not so easy there without knowledge about the processing costs of the mine.

Also, some more students were asked to draw a dig-limit. Their drawings are in figure A.5 in appendix A. They are all students in the field of geology or mining. The variance between these drawings indicate clearly how hard it is to draw a good dig-limit. Hence for a mine geologist, drawing the dig-limit is a precise job.

4.6. Constraints, variables and computation time

The amount of variables and constraints has a big influence on the complexity and the computation time of the model. The number of constraints and variables for different situations are shown in table 4.5. These values are extracted from the dig-limit model.

Grid	Frame	Average grade	No. of constraints	No. of variables
13x15	2x2	*No*	975	3315
13x15	2x2	*Yes*	976	3315
13x15	4x4	*No*	3315	12675
13x15	4x4	*Yes*	3316	12675
26x30	4x4	*No*	13260	50700
26x30	4x4	*Yes*	13261	50700
26x30	8x8	*No*	50700	200460
26x30	8x8	*Yes*	50701	200460

Table 4.5: Number of constraints and variables

The variables in the model are the decision variables from table 3.3. From the *Grid* and *Frame* columns in this table can be concluded that the number of variables in the model is:

$$\#\ \ Variables = 4 * X * Y * N_x * N_y + X * Y \tag{4.1}$$

Equation 4.1 gives indeed the right values if compared with the values extracted from the model in table 4.5. The total number of constraints was also extracted from the model. This number also depends on the grid size and the frame size. The frame constraints adds two constraints per grid point per frame point when it is formulated as in section 3.6.1. The two statements used to implement $a_{i,j,fx,fy}$ and $b_{i,j,fx,fy}$ do not

create additional constraints. To implement these constraint indicators were used. The Gurobi solver can work quicker with these. When the average grade constraint is applied this gives only one extra constraint. All this leads to formula 4.2 for the total number of constraints.

$$\begin{aligned} \#\ \ Constraints &= 2*X*Y*N_x*N_y + X*Y + 1*A \\ With: A &= \begin{cases} If\ \ average\ \ grade\ \ enabled: & 1 \\ else: & 0 \end{cases} \end{aligned} \tag{4.2}$$

The computation time can become large when large grids are input to the model or when the frames are chosen big. Large grids and large frames lead to many constraints and many variables. The time needed to find the optimal solution becomes larger when there are more constraints and variables. All the tests run in this thesis were run on the same computer. Some specifications of this computer can be found in table 4.6.

Operating system	Windows 7 Premium 64 bit
Processor	Intel Core i7-3630QM
Clock speed	4 x 2.4 GHz
Memory	8 GB, 1600 MHz DDR3 (2 x 4 GB)
Video card	Nvidia Quadro 1100M 2GB GDDR3

Table 4.6: Computer specifications

In figure A.4 in appendix A the runtime to get to each solution is displayed. These solutions are calculated for the $26x30$ grid with a $4x4$ frame. The first solutions show up after a couple of seconds, but in order to get to the optimal solution more than twenty minutes of runtime are needed. The $13x15$ grid only takes a couple of seconds to solve completely. Still, twenty minutes of runtime is for industry purposes fast. However, when the grid size and frame size increase, the runtime increases exponentially. This is not so fortunate for industry purposes, as this makes it difficult to determine the dig-limit for a large grid. Even if the area of the grid is not that large, it is better to take the grid as large as possible, this gives a more precise solution. The runtime for the three different grid sizes is in table 4.7.

Grid	Frame	Runtime	Optimum found
$13x15$	$2x2$	1.35 s	Yes
$26x30$	$4x4$	1239.20 s	Yes
$52x60$	$8x8$	16025.30 s	No (38.3% gap)

Table 4.7: Model runtime for different grid sizes

The model was also run on a $52x60$ grid with a $8x8$ frame to create an even more detailed solution, but this situation would never find the optimal solution. The program got stuck at 38.3 % gap between objective bounds due to a memory error. The best solution found by the program, however not optimal, is shown in figure A.3 in appendix A.

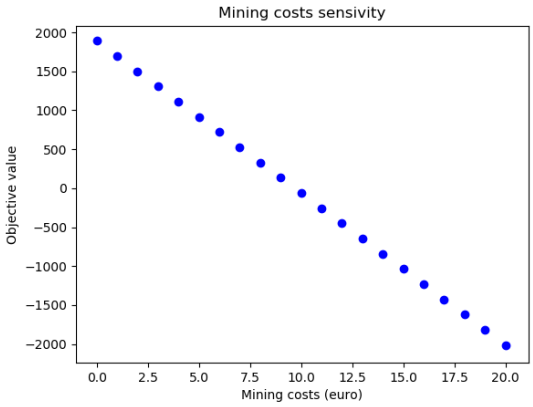
4.7. Sensitivity analysis

Variable	Original value in model	Range for sensitivity analysis
P	€ 80	[0 : 10 : 200]
C_p	€ 15	[0 : 3 : 60]
C_m	€ 5	[0 : 1 : 20]

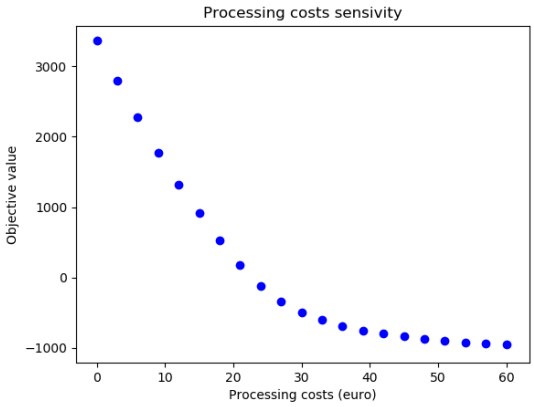
Table 4.8: Range of variables varied for sensivity analysis

A sensitivity analysis was carried out on the $13x15$ grid in the $2x2$ frame set-up. The analysis focused on the economic variables from the models objective function: the gold price, the mining costs and the processing costs. The model was run for a range of input values for each variable to see its influence on the models objective value. In table 4.8 the range of each variable is given. [22]

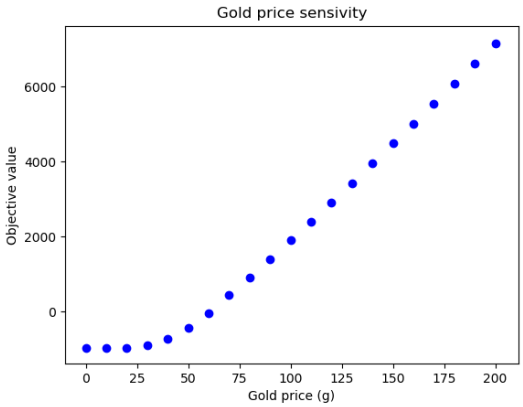
The influence of the economic variables on the models objective value is plotted in figure 4.7. The mining costs are in figure 4.7a, the processing costs in figure 4.7b and the gold price in figure 4.7c.



(a) Objective sensitivity for mining costs



(b) Objective sensitivity for processing costs



(c) Objective sensitivity for gold price

Figure 4.7: Objective sensitivity for the economic variables in the model

From the figure 4.7 can be concluded, the mining costs have a linear influence on the objective value of the model, the processing costs have a partial linear and a partial asymptotic relation with the objective value and the gold price also has a partial linear and a partial asymptotic relation with the objective value. This can be explained with the objective function 3.1 from section 3.5. In this function, the mining costs will always be applied to each SMU, but the processing costs and the gold price apply only when the SMU is characterized as ore. Therefore, when either the gold price is becoming low, or either the processing costs are becoming high, the linear behaviour between the objective value and the gold price or the processing costs ends. From this moment, the behaviour changes to a non-linear behaviour. The graphs go to an asymptote, which can be explained by the moment, the model chooses to select all blocks as waste. At this moment, the only cost made is mining the waste blocks.

5

Conclusion

In the previous chapter, the results from the model proposed in chapter 3 were given. In this chapter, the results will be discussed and a final conclusion will be made.

5.1. Discussion & recommendations

In this section three topics will be discussed and recommendations about these topics will be given.

5.1.1. Grid resolution

In this thesis, the dig-limit optimization model was calculated successfully for a $13x15$ grid and a $26x30$ grid. Larger grid resolutions give more precise results and higher objective values. However, when using a larger grid, the model becomes much more complicated. It was tried to solve the model for a $52x60$ grid, but this resulted in a memory error. The model found feasible solutions, but could not find the optimal solution for the larger grid.

In mining businesses, dig-limit optimization is done on a much larger scale. The grade distribution grids are much more detailed than the grids used in this thesis. The MILP model proposed in this thesis is not able to calculate the dig-limit for such a grid.

Recommendation for further research is to try to make the MILP model work for larger grids. This gives the model more realistic purposes. This can be done by using more computational power, for example use a supercomputer or to significantly decrease the amount of constraints. When formulated different the amount of constraints can maybe be made less. A new objective function with a penalty for invalid frames also might be a solution.

5.1.2. Comparative material

The final dig-limits calculated by the MILP model were visually compared to dig-limits drawn by geology or mining students. The most striking parts of the dataset were almost all the time characterised the same by the computer model as by the students. However, the harder to classify areas were each time classified different by the students.

In this thesis, the model results were only visually compared with other results. The only other results used were dig-limits drawn by geology or mining students. To decide whether the optimal solution of the model really is the optimal solution, it should be compared with other solutions. Not only visually, but also the objective values should be compared.

Recommendation for further research is to compare the model solutions with other solutions. Solutions from other literature or hand drawn solutions with an objective value will be appropriate for comparison.

5.1.3. Statistics

A small sensitivity analysis was carried out on the economic variables in the proposed dig-limit model. This gave some more insight how the economic variables change the objective value.

In the model, more variables than only the economic variables were used. A sensitivity analysis on the other variables would also give insight on the influence of these variables. To check the robustness of the model more statistics are preferred.

For further research is recommended to invest more time into sensitivity analysis and other statistics. This could help in making conclusions about the final results and give the uncertainty of the model.

5.2. Conclusion

In this thesis an answer was sought on the question: 'Can the method of Mixed Integer Linear Programming be used for dig-limit optimization, grade control and therefore profit maximization in a mine on a bench scale?'. Therefore a model for dig-limit optimization was proposed which took the *frame constraint* and the *average grade constraint* into account.

A MILP model was created with an objective function that maximized the mine profit for dig-limit optimization. The frame constraint was succesfully implemented to the dig-limit optimization model. The optimal solution to this model was calculated. The objective values decreased when the frame area became larger. So a higher objective value can be achieved if the frame size is decreased. The average grade constraint could also be added to the model. The average frame constraint affected the model results by selecting less material as ore. This provides an excellent tool to find the most valuable frames. Implementation of these constraints into the model added complexity to the model. The frame constraint added a lot of constraints but the average grade constraint added only one. For a small input grid and a small frame the solution to the model was quickly found, but when the input grid or/and the frame size became larger, the calculation became more time consuming.

Grid size	Frame size	Runtime	Objective value	No. of constraints	No. of variables
13x15	2x2	1.35 s	915.18	976	3315
26x30	4x4	1239.20 s	3741.85	13261	50700

Table 5.1: Final results for 13x15 and 26x30 grid

Two differently sized grade distributions of the same dataset were used as input to the model. The main results calculated in this thesis for the two grids are in table 5.1. From the results on different grid resolutions can be concluded, the larger the grid and frame resolution, the larger the objective value. In other words, using a larger resolution leads to more detailed solutions.

The results showed that the method of MILP could be used to maximize the profit in a mine on a bench level. The frame constraint and the average grade constraint were also successful implemented into the model. The model could not be optimized for a high resolution grid, but showed the optimal results for smaller ones. With the improvements from section 5.1 the results might even become more detailed. Therefore, the method of MILP can be succesfully used for dig-limit optimization and grade control on a bench level in a mine.

Bibliography

[1] M.J. Mahdavipour B. Maleki1, E. Mozaffari. Optimizing the cut off grade in sarcheshmeh copper mine using lane quartet model. *Journal of Mining and Metallurgy*, 52, 2016.

[2] E. Ben-Awuah H. Askari-Nasab, Y. Pourrahimian and S. Kalantari. Mixed integer linear programming formulations for open pit production scheduling. *Journal of Mining Science*, 2011.

[3] Mark Gershon. Mine scheduling optimization with mixed integer programming. 35:4, 04 1983.

[4] Yksel Asli Sari and Mustafa Kumral. Dig-limits optimization through mixed-integer linear programming in open-pit mines. *Journal of the Operational Research Society*, 2017.

[5] M. Kumral Y. A. Sari, N. Germain. Short-term production planning for multi-metal open-pit mines with equipment size constraints. *IMCET*, 2017.

[6] E. Ben-Awuah S. Kalantari H. Askari-Nasab, Y. Pourrahimian. Mixed integer linear programming formulations for open pit production scheduling. *Journal of Mining Science*, 2011.

[7] Christian Truden Philipp Hungerländer. A mixed-integer linear program for the traveling salesman problem with structured time windows. 2017.

[8] Hax Bradley and Magnanti. *Applied Mathematical Programming*. Addison-Wesley, 1977.

[9] T.; Gabbouj M. Kiranyaz, S.; Ince. *Multidimensional Particle Swarm Optimization for Machine Learning and Pattern Recognition*. Springer, 2014.

[10] J. Pearl. Heuristics: Intelligent search strategies for computer problem solving.

[11] Timo Berthold. Primal heuristics for mixed integer programs. Master's thesis, Technischen Universitĺat Berlin, 2006.

[12] K. Papadimitriou, C. H.; Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover, Mineola, NY, 1998.

[13] Gurobi documentation. Mixed-integer programming (mip) - a primer on the basics, . URL http://www.gurobi.com/resources/getting-started/mip-basics.

[14] P. Girodet-G. Hentges G. Ribiere O. Vincent M. Benichou, J. M. Gauthier. Experiments in mixed-integer linear programming. *Mathematical Programming*, 1, 1970.

[15] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1910129.

[16] R. Mohan Srivastava Edward H. Isaaks. *An Introduction to Applied Geostatistics*. Oxford University Press, 1990. ISBN 978-0195050134.

[17] A.J. Richmond and J.E. Beasley. Financially efficient dig-line delineation incorporating equipment constraints and grade uncertainty. *International Journal of Surface Mining, Reclamation and Environment*, 18(2):99–121, 2004. doi: 10.1080/13895260412331295376. URL https://doi.org/10.1080/13895260412331295376.

[18] H. Paul Williams. *Model Building in Mathematical Programming*. Wiley, fifth edition, 2013. ISBN 978-1118443330.

[19] Gurobi documentation. Gurobi 7.5 performance benchmarks, . URL http://www.gurobi.com/pdfs/benchmarks.pdf.

[20] Gurobi documentation. Licensing overview, . URL http://www.gurobi.com/products/licensing-pricing/licensing-overview.

[21] Gurobi documentation. quicksum() function, . URL https://www.gurobi.com/documentation/8.0/refman/py_quicksum.html.

[22] John N. Hooker M. W. Dawande. Inference-based sensitivity analysis for mixed integer/linear programming. *OPERATIONS RESEARCH*, 1998.

A

Supplementary figures

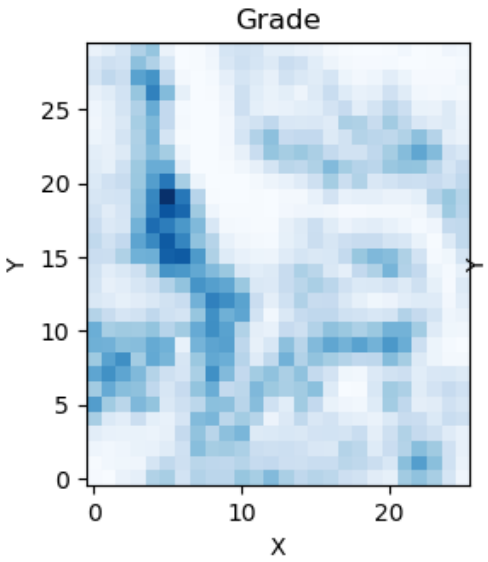


Figure A.1: Reblocked Walter Lake dataset 26x30 blocks

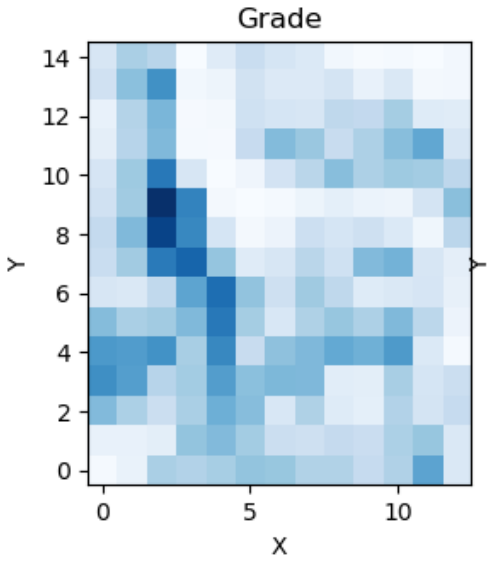


Figure A.2: Reblocked Walter Lake dataset 13x15 blocks

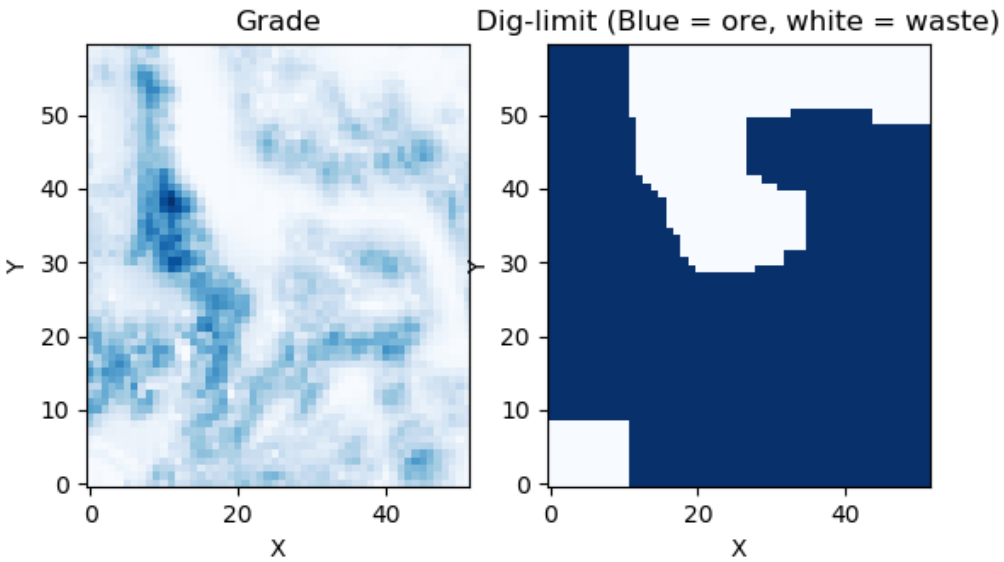


Figure A.3: Not optimal, but feasible, solution for 52x60 grid with 8x8 frames.

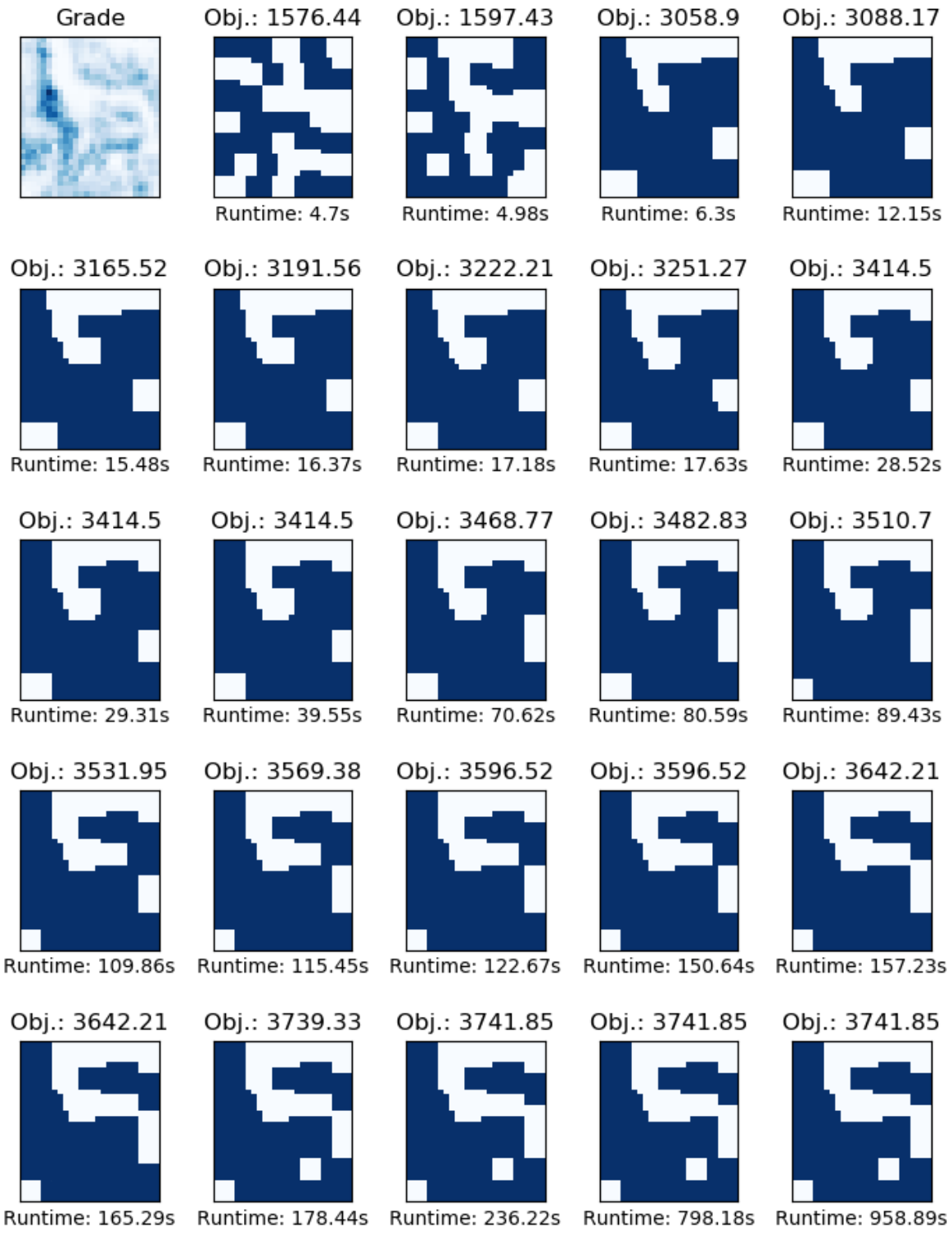


Figure A.4: Intermediate solutions for 26x30 grid with 4x4 frame

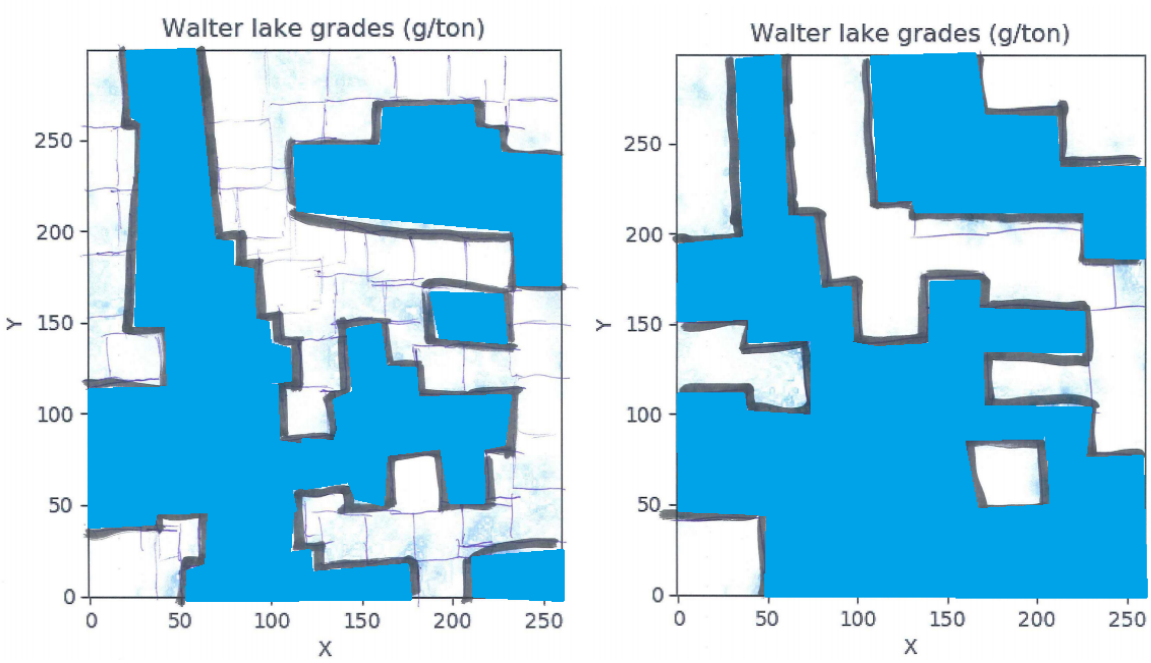


Figure A.5: More hand solutions to the dig-limit problem

B

Python code

Python code for Gurobi model

```
"""
-----------------------------------------------------------------------
Dig-Limit optimization with frame constraint and average grade constraint
using MILP with Gurobi solver (BSc thesis)
Author:          J.W. Buist
Date changed:    11-6-2018

This code should be runable when copied.
Packages needed: numpy, matplotlib and gurobipy
-----------------------------------------------------------------------
"""

# Import packages
import numpy as np
import matplotlib.pyplot as plt
from gurobipy import *

# Grades in numpy array A.
A = np.array(
[[14.980675,71.987125,339.9659,310.45195,346.0071
,400.221375,390.11285,315.477825,315.411675,247.35795
,319.521225,543.96945,149.3161],
[77.990825,77.51495,102.554875,399.4977,442.527375
,356.26615,224.2505,210.5221,253.7021,228.9633
,330.337225,391.4786,146.45435],
[442.159625,330.457075,221.4868,340.40985,492.836675
,430.3381,155.971375,324.265625,123.698075,93.9762
,311.19115,180.49135,248.449425],
[636.02145025,569.626575,294.185275,357.4485,571.355575
,420.481375,454.995725,450.6289,107.503375,101.773225
,340.9947,172.844,224.64855],
[591.55697475,575.80155,622.53929975,342.87525,662.89357425
,239.5807,411.4281,451.08705,520.625675,490.3847
,587.15075,141.027,17.252225],
[434.23275,339.844475,363.7772,448.742625,719.48987575
,352.66335,154.42535,322.924625,394.45025,328.180525
,449.1841,281.955475,56.65765],
[161.5912,143.601975,263.879425,539.51090025,756.87579975
,405.959875,208.966725,369.23965,270.596625,124.88125
```

```
,147.432225,167.709325,83.25535],
[232.510825,362.4823,713.2699015,793.14717525,396.456375
,119.522275,165.05245,289.104125,200.555375,444.194375
,479.50325,160.4131,97.6307],
[255.64635,447.385825,923.275501,662.34137475,171.7934
,22.29155,58.613025,216.725875,171.91315,206.36485
,139.3535,39.72755,283.4608],
[200.0147,367.69522525,997.77357275,678.40102575,18.10355
,5.820825,16.094825,55.615975,95.116075,63.23235
,51.948325,177.905,425.114425],
[166.264175,373.772425,718.3181255,157.775125,2.18055
,41.40645,183.6275,287.01105,427.9896,326.27405
,373.88535,359.247625,274.740575],
[95.634625,297.466175,447.551675,16.643525,13.526325
,246.7501,437.776725,385.597675,241.266325,326.0265
,427.4678,527.830125,159.12335],
[75.2331,307.7282,463.553075,11.579575,21.09645
,202.64245,172.582925,158.286925,274.14915,261.59065
,353.141875,124.019575,115.06985],
[199.89975,422.06585,627.3555755,32.353175,50.468475
,191.77565,137.04905,135.02825,179.7054,73.13545
,145.205375,19.83785,29.28485],
[162.10895,339.013275,291.629275,4.380075,122.463575
,232.42655,172.276425,138.552675,31.05835,7.165575
,16.752375,5.236575,28.886175]])

# declaring situation constants
M  = 1      # tonne
P  = 80     # per g
R  = 1      # 100 procent
Cp = 15     # per tonne
Cm = 5      # per tonne
G  = A/1000 # g per tonne

# Declaring model constants
(X,Y) = np.shape(A)

"""
-----------------------------------------------------------------------
Interesting inputs. (constraints)
Change these for different tests.

Explaination:
Nx and Ny are the x and y sizes of the frame for the frame constraint.
Gc is the average grade for the grade constraint.
-----------------------------------------------------------------------
"""
Nx = 2      # equipment x-size in terms of SMU
Ny = 2      # equipment y-size in terms of SMU
Gc = np.mean(G) * 1

# startup gurobi model
m = Model('milpmine')

# Declaring gurobi variables
x = m.addVars(X,Y,vtype=GRB.BINARY, name = 'x')
v = m.addVars(X,Y,Nx,Ny,vtype=GRB.BINARY, name = 'v')
t = m.addVars(X,Y,Nx,Ny,vtype=GRB.INTEGER, name = 't')
```

```
a = m.addVars(X,Y,Nx,Ny,vtype=GRB.BINARY, name = 'a')
b = m.addVars(X,Y,Nx,Ny,vtype=GRB.BINARY, name = 'b')

"""
-----------------------------------------------------------------------
Declare objective function
-----------------------------------------------------------------------
"""
# Objective function
m.setObjective(quicksum(M*(x[i,j]*G[i,j]*P*R - x[i,j]*Cp - Cm)
                for i in range(X) for j in range(Y)),GRB.MAXIMIZE)

"""
-----------------------------------------------------------------------
Loop to implement the frame constraint.
-----------------------------------------------------------------------
"""
for i in range(X):
    for j in range(Y):
        for fx in range(Nx):
            for fy in range(Ny):
                # Conditions for the frame constraint to hold
                cond1 = i - fx + Nx <= X
                cond2 = j - fy + Ny <= Y
                cond3 = i - fx >= 0
                cond4 = j - fy >= 0
                if cond1 and cond2 and cond3 and cond4:
                    # t is the sum of the values of x inside a frame
                    m.addConstr(t[i,j,fx,fy] == quicksum(x[i-fx+a,j-fy+b]
                    for a in range(0,Nx) for b in range(0,Ny)))
                else:
                    #so that is not 0 or N*N, (-1 -> infeasible solution)
                    m.addConstr(t[i,j,fx,fy] == 1)
                # Convert t to v if it is a valid frame using a and b
                m.addGenConstrIndicator(a[i,j,fx,fy], 1, t[i,j,fx,fy],
                                                        GRB.EQUAL, 0)
                m.addGenConstrIndicator(b[i,j,fx,fy], 1, t[i,j,fx,fy],
                                                        GRB.EQUAL, Nx*Ny)
                m.addGenConstrOr(v[i,j,fx,fy], [a[i,j,fx,fy], b[i,j,fx,fy]])
        # there must be a valid frame for each SMU
        m.addConstr(quicksum(v[i,j,fx,fy] for fx in range(0,Nx)
                                          for fy in range(0,Ny)) >= 1)
"""
-----------------------------------------------------------------------
Add the average grade constraint to the model
-----------------------------------------------------------------------
"""
# Add the average grade constraint
m.addConstr((quicksum(x[i,j]*G[i,j] for i in range(X) for j in range(Y)))
            >= Gc * (quicksum(x[i,j]for i in range(X) for j in range(Y))))

"""
-----------------------------------------------------------------------
Solve the model
-----------------------------------------------------------------------
"""
m.optimize()

```

```
156 # Read solution
157 solution = m.getAttr('x', x)
158 B = np.zeros((X,Y), dtype=int)
159 for i in range(X):
160     for j in range(Y):
161         B[i,j] = int(solution[i,j])
162 obj = m.getObjective()
163 
164 """
165 -----------------------------------------------------------------------
166 Plot grade distribution and dig-limit
167 -----------------------------------------------------------------------
168 """
169 plt.figure()
170 plt.suptitle('Grade distribution -> Dig-Limit')
171 
172 plt.subplot(121)
173 plt.imshow(A, cmap='Blues', origin='lower')
174 plt.title('Grade')
175 plt.xlabel('X')
176 plt.ylabel('Y')
177 
178 plt.subplot(122)
179 plt.imshow(B, cmap='Blues', origin='lower')
180 plt.title(('Optimal solution is: ' + str(round(obj.getValue(),2))))
181 plt.xlabel('X')
182 plt.ylabel('Y')
183 
184 plt.show()
```