Danmarks Tekniske Universitet

Project 3 - 02526 Mathematical Modeling

Sunday 29th March

Authors:
Mads Esben Hansen
Marcus Lenler Garsdal
Nicolaj Hans Nielsen

Study No:
s174434
s174440
s184335

Contents

1 Introduction

In the coming years we will see the real effects of global warming. It will leave much land inhospitable - in some areas due to drought and others due to flood caused by the rise of sea level. We might soon be forced to take drastic methods into account and here filling of the Qattara Depression with sea water might be one of them.
It has already been proposed several times [3] often with the same main issue; how to dig the tunnel?
From 1964 and on wards Friedrich Bassler led a project in which they planned to remove the sand with use of nuclear explosives. In this project we expand on this idea, and analyze the outcome of such methods implemented in different ways using Operations Research.

2 Methodology

2.1 Distance and interpolation

We a given a set of data with latitude, longitude and height for the channel. First, we want to calculate the distance between each point using the formulas below. We denote latitude, longitude and height; lat, lon & H, and then we can calculate the distance between two points, 1 & 2, as:

$$\begin{aligned}
\Delta_{\text{lon}} &= \text{lon}_2 - \text{lon}_1 \\
\Delta_{\text{lat}} &= \text{lat}_2 - \text{lat}_1 \\
a &= (\sin\frac{\Delta_{\text{lat}}}{2})^2 + \cos\text{lat}_1 \ \cos\text{lat}_2 \cdot (\sin\frac{\Delta_{\text{lon}}}{2})^2 \\
c &= 2\ \text{atan2}(\sqrt{a}, \sqrt{1-a}) \\
d &= \text{R}\, c
\end{aligned}$$

Here R is the radius of the earth and here we use the radius at the equator = 6378km.
With the calculated distances and their height, we interpolate to obtain points every 250 meters. We interpolate linearly using the built-in Matlab function; Interp1.

2.2 Mixed Integer Linear Problem

Initially we formulate the problem as a mixed integer linear problem. Here we introduce a binary variable for each of the calculated n points along the channel $x_i,\ i = 1, ..n$ which indicate whether we plant a nuke or not. We want to minimize the number of nukes hence we introduce:

$$\min \sum_{i=1}^{n} x_i$$

subjected to

$$\begin{aligned}
R_i &\geq H_i + CHD \ \ i = 1, ..., n \\
R_i &= K_2\, x_{i-2} + K_1\, x_{i-1} + K_0\, x_i + K_1\, x_{1+i} + K_2\, x_{i+2} \ \ i = 1, ..., n \\
x_i &\in 0, 1 \ \ i = 1, ..., n
\end{aligned}$$

The first constraint ensures that we remove enough dirt. The amount of dirt removed, R_i at position i should be greater than or equal to the height, H_i plus the necessary channel depth, CHD. We set $CHD = 10m$.
The second constraint accounts for influence of bombs on neighbouring positions. Therefore K_0 denotes the dirt removed at the position the nuke exploded, K_1 denotes the dirt removed 1 position away and K_2 is the amount of dirt removed 2 positions away. We assume that the dirt removed from one explosion does not replenish nearby positions. The solution to the MILP above, we label as the direct solution.

2.3 Refinements

2.3.1 Smooth channel

We now introduce a new objective function to enhance the flow of the channel:

$$\min \sum_{i=1}^{n} |R_i - H_i - CHD|$$

With this function we will minimize the depth underneath the CHD along the channel which in turn might make it smooth. In the implementation, we already have the constraint $R_i \geq H_i + CHD$ hence we will not take the absolute value i.e:

$$\min \sum_{i=1}^{n} R_i - H_i - CHD$$

2.3.2 Placement constraints

The engineers now worry that if bombs are placed directly next to each other and there is a tiny mis-timing, one of the bombs might be destroyed by the shock wave from the other. Therefore we don't want to place bombs right next to each other and we introduce another constraint:

$$x_i + x_{i+1} \leq 1 \;\; i = 1, ..., n-1$$

2.3.3 Dial a yield

Some nuclear explosives can be adjusted to give a specific yield and now we want to investigate if it would reduce the number of explosives needed and make the channel more smooth. We are given the following settings:

Setting	K_0	K_1	K_2
1	300	140	40
2	500	230	60
3	1000	400	70

To solve this for each position x_i, we make three binary variables for each setting,q:

$$x_{i,q} \;\; i = 1, .., n \;\; q = 1, 2, 3$$

For each position, we can only introduce at most one bomb and only with one setting therefore:

$$\sum_{q=1}^{3} x_{i,q} \leq 1 \;\; i = 1, ..n$$

Now we only have to modify a constraint to be able to use the different settings:

$$R_i = \sum_{q=1}^{3} K_{2,q}\, x_{i-2,q} + K_{1,q}\, x_{i-1,q} + K_{0,q}\, x_{i,q} + K_{1,q}\, x_{1+i,q} + K_{2,q}\, x_{i+2,q} \;\; i = 1, ..., n$$

3 Results

3.1 Impact on world oceans

The average depth of the Qattara Depression is about $60m$, and the area is $19605km^2$ [3]. This yields a total volume of:

$$60m = 0.06km$$
$$V_{QD} = 19605km^2 \cdot 0.06km = 1176.3km^3$$

The total area of the worlds oceans around $3.619 \cdot 10^8 km^2$ [2]. We can now compute the total effect the it will have on the oceans if the Qattara Depression was filled with water. This is assuming the area of the oceans will not be changed significantly by this.

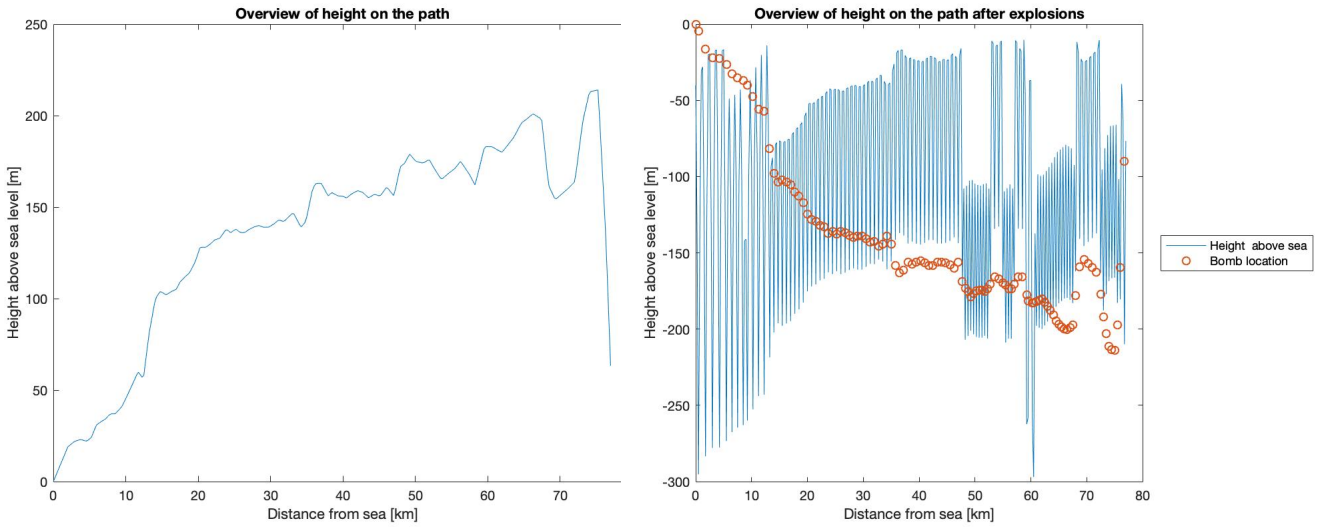
$$A_{oceans} = 3.619 \cdot 10^8 km^2$$
$$\Delta_h = \frac{V_{QD}}{A_{oceans}} = \frac{1176.3km^3}{3.619 \cdot 10^8 km^2}$$
$$\Delta_h = 3.250345399 \cdot 10^{-6} km \approx 0.325cm$$

3.2 Creating the channel

The first thing we want to compute is the minimum number of bombs required to create the channel. To do this we only apply the constraints, which ensures the channel is at least 10m deep everywhere - as described in the methodology section.
We find that the minimum number of bombs required are 111.

$$Bombs_{min} = 111$$

We can now plot the bomb locations and the impact on the channel:



(a) Interpolated height above sea level along channel

(b) Direct solution

We see that this solution does not yield a very smooth channel. There are places where the channel is very(!) deep, and others where it is quite shallow. Therefore we want to smoothen the channel. This is done by changing the objective function so that we now minimize the "unnecessary" amount of dirt remove i.e. minimize all dirt removed that makes the channel more than $10m$ deep. We found that this solution required 111 bombs, exactly as last time.

$$Bombs_{smooth} = 111$$

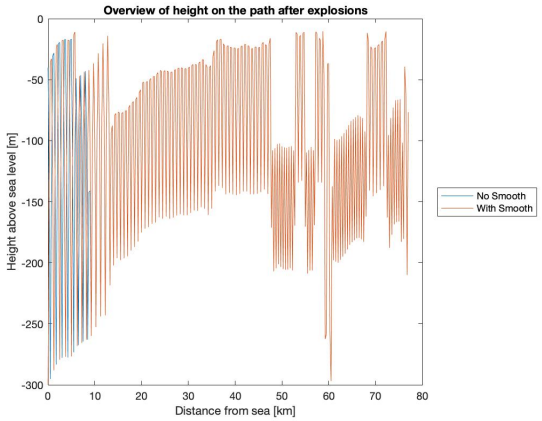


Figure 2: Smoothest channel only using one type of bomb.

When we introduce the objective function that minimizes the depth underneath the CHD, the most noticeable difference is the missing spike at ≈ 57 km. Otherwise this model does not seem to make the channel more smooth.

For practical reasons and for the sake of security we do not want to place to bombs next to each other. We therefore now want the solution where no bombs are placed next to each other; as before we again minimize the amount of unnecessary amount of dirt remove rather than just the amounts of bombs used. We found that this problem with no neighbors also required 111 bombs.

$$Bombs_{NN} = 111$$

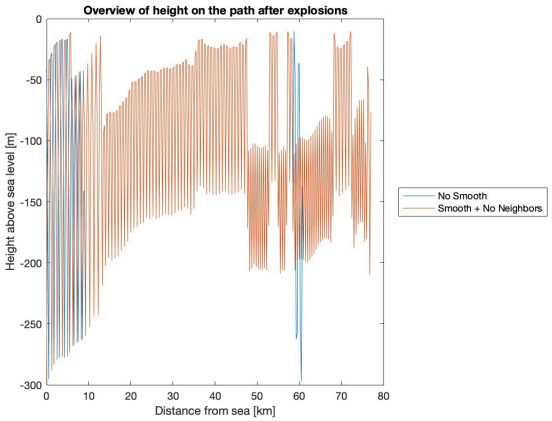


Figure 3: Smoothest channel where no bombs are next to each other.

We see that there is no significant difference when we introduce the constraint on no bombs next to each other. Therefore we are able to meet the engineers concern without any significant changes in channel flow.

We are now told the bombs have 3 different payload setting - until now we have only used type 1. We therefore want to solve the previous problem - smooth but no neighbors - utilizing this new property. We find the solution to this problem is to use 97 bombs of type 1 and 7 of type 2.

$$Bombs_I = 97$$
$$Bombs_{II} = 7$$
$$Bomb_{total} = 104$$

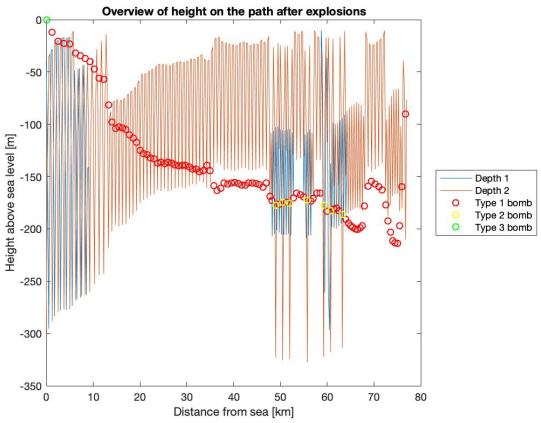


Figure 4: Smoothest channel where no bombs are next to each other and the bomb can have 3 different payloads.

The solution is given as the depth 2 in the plot. We see that while it is the optimal solution given our constraints it is not very smooth. It does, however, minimize the amount of unnecessary dirt removed which is what we were asked to do.

4 Discussion

For the first and simple model it was found that it initially requires 111 nuclear bombs to dig the tunnel. The bombs are spread out evenly, and the channel depth fulfills the required constraints. This model assumes that dirt removed at one position does not replenish nearby positions when blown up. In reality it would be expected that the individual bombs would relocate the dirt to nearby positions, which would render a lot of the work done by bombs useless. Another uncertain aspect of the model is that it is solved by a resolution of 250 meters throughout the 70km tunnel. To improve the model in this regard, one could introduce functions to describe the dirt removed relative to the distance from the explosion. This would allow the problem to be solved with a greater resolution. It could also be considered to look into stochastic programming, to better describe the dirt removed at each location, assuming that the spread would be random from explosion to explosion.
For the model optimizing channel smoothness, the results are perhaps not as great as one would expect from minimizing the depth underneath the CHD. The biggest difference between this model iteration and the previous, is a smoothing of the channel depth around 57km. The number of bombs remain the same, though being placed differently, and the lowest point of the tunnel is still located between 200-300m. We could instead use a objective function that minimizes the sum of absolute differences between to neighboring points. Further, to asses the smoothness one should perhaps also consider the horizontal displacement of dirt because it is probably going to be greater than the vertical as it was seen in for instance the Sedan test [1].

For one of the final iterations of the model it is implemented that bombs cannot be located next to each other. Solving the model shows that this does not significantly change the results for channel smoothness, and the same amount of bombs is used. It can therefor be implemented without problem to meet the engineers concerns. In the final iteration of the model, it was tested if introducing dial-a-yield to the implementation could improve smoothness of the channel. The results showed no significant gain in smoothness, although it did use fewer bombs than previously.
A big step to improve the model would be to discuss the resolution of the model. Instead of a few large bombs, one could try to implement a similar model using many and smaller bombs and analyse how these results would compare against the nuclear bombs. This of course removes the proposition from Friedrich Bassler, but would be a great tell if nuclear bombs are really worth it compared to other explosive methods. One could argue that it would be better to use normal explosives, as this would reduce the impact on nearby wildlife and humans inhabitants in terms of radiation. The later effects of nuclear fallout would also be avoided.
However, using smaller bombs would result in having to use more bombs which would increase the complexity of the problem. As it is solving the final problem with 3 settings took quite a while, even using state of the art solvers. A way to significantly reduce the complexity would be a dynamic programming approach. This approach would also be able to handle non-convex problem such as solving for a sum of absolute values.

5 Conclusion

In this project we reviewed the idea of filling the Qattara Depression by means of nuclear explosives. We formulated the scenario as a mixed integer linear problem and remodelled it several times to asses the bombs needed, for safety in case of mistiming of the bombs, the smoothness of the channel and with a bomb with three different payloads. The most smooth result is obtained with consistent payload and with the aim of minimizing the depth underneath 10 meters, however, we use fewer bombs when different payloads are used. We speculate both the resolution of 250 meters in the problem and if the problem is best modelled as MILP. Further, due to the unwanted impact of radiation, we suggest that other ways to dig out the tunnel are investigated.

6 References

References

[1] Wikipedia. *Sedan (nuclear test) — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Sedan\%20(nuclear\%20test)&oldid=938536486. [Online; accessed 29-March-2020]. 2020.

[2] Wikipedia contributors. *Ocean — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Ocean&oldid=945854590. [Online; accessed 29-March-2020]. 2020.

[3] Wikipedia contributors. *Qattara Depression — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Qattara_Depression&oldid=947562503. [Online; accessed 28-March-2020]. 2020.

Appendices

A Matlab implementations

```
clear all;
data = readmatrix('inter2.txt');


%% Plot height as a function of distance (we use the accumulated dist)
```

```
plot(data(:,1), data(:,2))
title('Overview of height on the path')
xlabel('Distance from sea [km]')
ylabel('Height above sea level [m]')


%% JULIA STUFF 1 - Problem  3 "nukeDemo.jl"

results_q3 = readtable("results_q3.csv");
R1 = table2array(results_q3(:,end));
bombs1 = table2array(results_q3(:,1));

% hvis du skifter mellem resultaterne skal du vende R-vektoren
post_data = data;
post_data(:,2) = post_data(:,2) - R1;

plot(post_data(:,1), post_data(:,2))
title('Overview of height on the path after explosions')
xlabel('Distance from sea [km]')
ylabel('Height above sea level [m]')

hold on
scatter(data(:,1).*bombs1, -data(:,2).*bombs1)
legend("Height  above sea", "Bomb location", 'Location', 'EastOutside')

%% Julia stuff 2 - problem 4 "nukeDemo2.jl"

results_q4 = readtable("results_q4.csv");
R2 = table2array(results_q4(:,end));
bombs2 = table2array(results_q4(:,1));


post_data2 = data;
post_data2(:,2) = post_data2(:,2) - R2;

plot(post_data(:,1), post_data(:,2), post_data2(:,1), post_data2(:,2))
title('Overview of height on the path after explosions')
xlabel('Distance from sea [km]')
ylabel('Height above sea level [m]')
legend("No Smooth", "With Smooth",'Location','EastOutside')

%% Julia stuff 3 - problem 5 "nukeDemo3.jl"

results_q5 = readtable("results_q5.csv");
R3 = table2array(results_q5(:,end));
bombs3 = table2array(results_q5(:,1));

post_data3 = data;
post_data3(:,2) = post_data3(:,2) - R3;

plot(post_data(:,1), post_data(:,2), post_data3(:,1), post_data3(:,2))
title('Overview of height on the path after explosions')
xlabel('Distance from sea [km]')
ylabel('Height above sea level [m]')
legend("No Smooth", "Smooth + No Neighbors",'Location','EastOutside')


%% Julia stuff 3 - problem 6 "nukeDemo4.jl"

results_q6 = readtable("results_q6.csv");
R4 = table2array(results_q6(:,end));
bombs4 = table2array(results_q6(:,1:3));
n4 = sum(bombs4,'all');

post_data4 = data;
```

```
post_data4(:,2) = post_data4(:,2) - R4;

plot(post_data(:,1), post_data(:,2),post_data4(:,1),post_data4(:,2))
title('Overview of height on the path after explosions')
xlabel('Distance from sea [km]')
ylabel('Height above sea level [m]')
hold on
scatter(data(:,1).*bombs4(:,1), -data(:,2).*bombs4(:,1), 'r')
hold on
scatter(data(:,1).*bombs4(:,2), -data(:,2).*bombs4(:,2), 'y')
hold on
scatter(data(:,1).*bombs4(:,3), -data(:,2).*bombs4(:,3), 'g')
legend("Depth 1", "Depth 2", "Type 1 bomb", "Type 2 bomb", "Type 3 bomb", 'Location', 'EastOutside')
```

B Julia implementations

B.1 Minimize bombs

```
using GLPK, JuMP, SparseArrays, Gurobi, CSV
using DelimitedFiles
using CSV, DataFrames

include("H_interpolate.jl")

K = [
300 140 40
]


function constructA(H,K)
    n = length(H)
    A = zeros(n,n)
    #for m=1:3
        for i=1:n
            for j=0:3-1
                if i+j ≤ n
                    A[i,i+j] = K[j+1]
                    A[i+j,i] = K[j+1]
                end
                if i-j ≥ 1
                    A[i,i-j] = K[j+1]
                    A[i-j,i] = K[j+1]
                end
            end
        end
    #end
    return A
end

h = length(H)
myModel = Model(with_optimizer(Gurobi.Optimizer))
# If your want ot use GLPK instead use:
#myModel = Model(GLPK.Optimizer)

M = 1800 # Max is: 400 + 1000 + 400
A = constructA(H,K)
CHD = 10

@variable(myModel, x[1:h], Bin)
@variable(myModel, R[1:h], Int)
```

```
@objective(myModel, Min, sum(x[j] for j=1:h ))

# Constraint we must remove HEIGHT(i) + 10m dirt R(i) at location i
@constraint(myModel, [j=1:h],R[j] ≥ H[j] + CHD )
# Constraint the dirt R(i) removed at location i is a sum of all nearby bombs
@constraint(myModel, [i=1:h],R[i] == sum(A[i,j]*x[j] for j=1:h))

optimize!(myModel)

if termination_status(myModel) == MOI.OPTIMAL
    println("Objective value: ", JuMP.objective_value(myModel))
    println("x = ", JuMP.value.(x))
    println("R = ", JuMP.value.(R))
else
    println("Optimize was not succesful. Return code: ", termination_status(myModel))
end

#We export results
PATH = "C:/Users/Garsdal/Desktop/02526 Mathematical Modelling/Qattara/Results/results_q3.csv"
results = hcat(JuMP.value.(x), JuMP.value.(R))
CSV.write(PATH, DataFrame(results))
```

B.2 Smooth

```
using GLPK, JuMP, SparseArrays, Gurobi, CSV
using DelimitedFiles
using CSV, DataFrames

include("H_interpolate.jl")

K = [
300 140 40
]


function constructA(H,K)
    n = length(H)
    A = zeros(n,n)
    #for m=1:3
        for i=1:n
            for j=0:3-1
                if i+j ≤ n
                    A[i,i+j] = K[j+1]
                    A[i+j,i] = K[j+1]
                end
                if i-j ≥ 1
                    A[i,i-j] = K[j+1]
                    A[i-j,i] = K[j+1]
                end
            end
        end
    #end
    return A
end

h = length(H)
myModel = Model(with_optimizer(Gurobi.Optimizer))
# If your want ot use GLPK instead use:
#myModel = Model(GLPK.Optimizer)

M = 1800 # Max is: 400 + 1000 + 400
A = constructA(H,K)
```

```
CHD = 10

@variable(myModel, x[1:h], Bin)
@variable(myModel, R[1:h], Int)
@objective(myModel, Min, sum(R[j] - H[j] - CHD for j=1:h ))

# Constraint we must remove HEIGHT(i) + 10m dirt R(i) at location i
@constraint(myModel, [j=1:h],R[j] ≥ H[j] + CHD )

# Constraint the dirt R(i) removed at location i is a sum of all nearby bombs
@constraint(myModel, [i=1:h],R[i] == sum(A[i,j]*x[j] for j=1:h))

optimize!(myModel)

if termination_status(myModel) == MOI.OPTIMAL
    println("Objective value: ", JuMP.objective_value(myModel))
    println("x = ", JuMP.value.(x))
    println("R = ", JuMP.value.(R))
else
    println("Optimize was not succesful. Return code: ", termination_status(myModel))
end

#We export results
PATH = "C:/Users/Garsdal/Desktop/02526 Mathematical Modelling/Qattara/Results/results_q4.csv"
results = hcat(JuMP.value.(x), JuMP.value.(R))
CSV.write(PATH, DataFrame(results))
```

B.3 No neighbors

```
using GLPK, JuMP, SparseArrays, Gurobi, CSV
using DelimitedFiles
using CSV, DataFrames

include("H_interpolate.jl")

K = [
300 140 40
]


function constructA(H,K)
    n = length(H)
    A = zeros(n,n)
    #for m=1:3
        for i=1:n
            for j=0:3-1
                if i+j ≤ n
                    A[i,i+j] = K[j+1]
                    A[i+j,i] = K[j+1]
                end
                if i-j ≥ 1
                    A[i,i-j] = K[j+1]
                    A[i-j,i] = K[j+1]
                end
            end
        end
    #end
    return A
end

h = length(H)
myModel = Model(with_optimizer(Gurobi.Optimizer))
```

```
34
35 # If your want ot use GLPK instead use:
36 #myModel = Model(GLPK.Optimizer)
37
38 M = 1800 # Max is: 400 + 1000 + 400
39 A = constructA(H,K)
40 CHD = 10
41
42 @variable(myModel, x[1:h], Bin)
43 @variable(myModel, R[1:h], Int)
44 @objective(myModel, Min, sum(R[j] - H[j] - CHD for j=1:h ))
45
46 # Constraint we must remove HEIGHT(i) + 10m dirt R(i) at location i
47 @constraint(myModel, [j=1:h],R[j] ≥ H[j] + CHD )
48 # Constraint the dirt R(i) removed at location i is a sum of all nearby bombs
49 @constraint(myModel, [i=1:h],R[i] == sum(A[i,j]*x[j] for j=1:h))
50 # We dont allow two bombs next to each other
51 @constraint(myModel, [j=1:h-1], x[j] + x[j+1] ≤ 1)
52
53 optimize!(myModel)
54
55 if termination_status(myModel) == MOI.OPTIMAL
56     println("Objective value: ", JuMP.objective_value(myModel))
57     println("x = ", JuMP.value.(x))
58     println("R = ", JuMP.value.(R))
59 else
60     println("Optimize was not succesful. Return code: ", termination_status(myModel))
61 end
62
63 #We export results
64 PATH = "C:/Users/Garsdal/Desktop/02526 Mathematical Modelling/Qattara/Results/results_q5.csv"
65 results = hcat(JuMP.value.(x), JuMP.value.(R))
66 CSV.write(PATH, DataFrame(results))
```

B.4 3 settings

```
1  using GLPK, JuMP, SparseArrays, Gurobi, CSV
2  using DelimitedFiles
3  using CSV, DataFrames
4
5  include("H_interpolate.jl")
6
7  K = [
8  300 140 40
9  500 230 60
10 1000 400 70
11 ]
12
13
14 function constructA(H,K)
15     n = length(H)
16     A = zeros(n,n,3)
17     for m=1:3
18         for i=1:n
19             for j=0:3-1
20                 if i+j ≤ n
21                     A[i,i+j,m] = K[m,j+1]
22                     A[i+j,i,m] = K[m,j+1]
23                 end
24                 if i-j ≥ 1
25                     A[i,i-j,m] = K[m,j+1]
26                     A[i-j,i,m] = K[m,j+1]
```

```
                end
            end
        end
    end
    return A
end

h = length(H)
myModel = Model(with_optimizer(Gurobi.Optimizer))

# If your want ot use GLPK instead use:
#myModel = Model(GLPK.Optimizer)

M = 1800 # Max is: 400 + 1000 + 400
A = constructA(H,K)
CHD = 10

@variable(myModel, x[1:h,1:3], Bin)
@variable(myModel, R[1:h], Int)

@objective(myModel, Min, sum(R[j] - H[j] - CHD for j=1:h ))

# Constraint we must remove HEIGHT(i) + 10m dirt R(i) at location i
@constraint(myModel, [j=1:h],R[j] ≥ H[j] + CHD )

# Constraint the dirt R(i) removed at location i is a sum of all nearby bombs
@constraint(myModel, [i=1:h],R[i] == sum(sum(A[i,j,q]*x[j,q] for j=1:h) for q=1:3) )

# We can only use one type of bomb of at one spot
@constraint(myModel,[i=1:h], sum(x[i,q] for q=1:3) ≤ 1)

# We dont allow two bombs next to each other
@constraint(myModel, [j=1:h-1], x[j] + x[j+1] ≤ 1)

optimize!(myModel)

if termination_status(myModel) == MOI.OPTIMAL
    println("Objective value: ", JuMP.objective_value(myModel))
    println("x = ", JuMP.value.(x))
    println("R = ", JuMP.value.(R))
else
    println("Optimize was not succesful. Return code: ", termination_status(myModel))
end

#We export results
PATH = "C:/Users/Garsdal/Desktop/02526 Mathematical Modelling/Qattara/Results/results_q6.csv"
results = hcat(JuMP.value.(x), JuMP.value.(R))
CSV.write(PATH, DataFrame(results))
```