KY1: The MMU High Altitude Problem

Kieran Molloy

January 2020

Abstract

Perform MILP Optimisation on a 0-1 Knapsack problem presented by High Energy Atmospheric Research Group on which experiments are taken to maximise grant income.

1 Introduction

The University High Energy Atmospheric Research Group (HEARG) is to launch a high-altitude balloon mission which is to carry a number of experiments into the upper atmosphere where various different features of the high altitude environment can be studied. The balloon will carry a platform which will provide a structure to support a number of different experiments and will supply them with electric power. There is not enough capacity for all the projects suggested by re-searchers within the department, and it is necessary to choose a collection of experiments which will not exceed the current draw that can be provided by the platform or be too heavy. A list of proposed experiments is given below, together with the weight and the current requirement for each one. The experiments are evaluated in terms of their potential to produce further grants. The objective of the project is to generate the greatest possible amount of further grant money which will be used to finance further projects. This paper will advise on which experiments should be taken on the flight based on an optimisation function.*KY1-Reference* 2019

2 Analysis

2.1 Details of the Problem

The platform can carry experiments with a total mass not exceeding 14.0 kg and is able to provide up to 45mA of electric current for them. Eleven different possible experiments have been proposed for the mission (see table 1), but there is neither the mass capacity nor the current generation to carry all of them. It is therefore necessary to choose a list of experiments which will be carried by the platform. Each experiment has been evaluated in terms of its mass and the current it will draw from the power pack. The expected benefit of the experiment has also been estimated in terms of the value of additional

funding that will be granted to HEARG if the experiment is successfully launched (see table 1). The standard limit of 45 mA can be extended by plugging additional power packs into the platform. This will produce 10 mA per power pack, but each power pack weighs 2 . 0 kg. Should the university install any power packs, and if so, how many?*KY1-Reference* 2019

id	Experiment Name	Mass(kg)	Current(mA)	Grant(£m)
1	UVS High Altitude UV Spectrometer	3.3	17	90
2	OPP Ozone Partial Pressure Machine	2.1	15	105
3	CFC Spectrometry Analyser	1.8	8	60
4	IRIS IR Imaging Spectrometer	4.2	25	90
5	NOX Nitrogen Oxide Concentration Logger	2.4	16	60
6	HDS Hydrocarbon Detection System	1.5	12	70
7	CRIS Cosmic Ray Imaging Spectrometer	3.6	24	130
8	GBD Gamma Burst Detector	1.3	5	45
9	HEP High Energy Particle Calorimeter	3.6	8	65
10	WV Water Vapour Concentration Device	1.9	14	40
11	ECS Exotic Compounds Mass Spectrometer	2.8	30	80

Table 1: Experiments that could be taken

2.2 Formulation

Since this is a 0-1 Knapsack problem with an addition we can base the model on that, W is the maximum capacity and N is the set of all potential objects, in addition we need x_i, v_i and w_i where x_i is the binary variable on whether we take i or not, v_i is an arbitrary value and w_i is the capacity constraint. The condition is that you cannot take multiple of i.Taha 2017 This basic model is as follows:

$$\begin{aligned} \underset{X}{\text{maximise}} \quad & \sum v_i x_i \\ \text{subject to} \quad & \sum w_i x_i \leq W, \ i \in N, \\ & x_i \succeq 0, \ i \in N. \end{aligned} \tag{1}$$

We modify the basic problem for the papers purposes by redefining v as g, and adding in the extra current problem as defined here.

$$\begin{aligned} \alpha + y w_{batt} \leq W_{max}, \ i \in N, \\ \beta - y a_{batt} \leq A_{max}, \ i \in N, \end{aligned} \tag{2}$$

Where α is the standard weight constraint and β is a modified weight constraint, modelling current.

variable	description
x_i	1 = take experiment i, 0=else
y	number of battery packs taken
N	set of potential experiments i
W	set of experiment weights w_i
G	set of experiment grants g_i
A	set of experiment currents a_i
A_{max}	Maximum Current
W_{max}	Maximum Weight
a_{batt}	Battery Weight
w_{batt}	Battery Current

Table 2: Modelled Variables

2.2.1 MILP

Combining Eq.1 and Eq.2

$$\begin{aligned} \underset{X}{\text{maximise}} \quad & \sum g_i x_i \\ \text{subject to} \quad & \sum w_i x_i + y w_{batt} \leq W_{max},\ i \in N, \\ & \sum a_i x_i - y a_{batt} \leq A_{max},\ i \in N, \\ & x_i \succeq 0,\ i \in N. \end{aligned}$$

3 Results

3.1 Weight = 14

This sections results can be generated by running *KY1_weight14.m* with *getOptimalSolution.m* in the same directory. When the max weight of the ship is 14, the maximum value from grants is 370, this

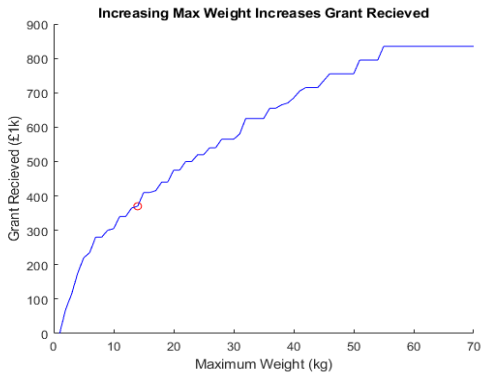
$$\begin{aligned} z = 370, \quad & y = 2, \\ x_1 = 1, \quad & x_2 = 1, \quad x_3 = 1, \quad x_4 = 0, \quad x_5 = 0, \quad x_6 = 1, \\ x_7 = 0, \quad & x_8 = 1, \quad x_9 = 0, \quad x_{10} = 0, \quad x_{11} = 0. \end{aligned}$$

From here on, this will be represented as 11101010002 where the first 11 values are the binary representation of $x_i, x \to 11$ and the final value is the number of batteries taken. This is to help understand larger data sets where the best maximum weight of the balloon is to be assessed. If the balloon has a maximum limit of 14kg; HEARG should load 2 batteries onto the balloon and take the following experiments: UVS High Altitude UV Spectrometer, OPP Ozone Partial Pressure Machine, CFC Spectrometry Analyser, HDS Hydrocarbon Detection System, CRIS Cosmic Ray Imaging Spectrometer and GBD Gamma Burst Detector. This allocation optimises the amount of grant money MMU will receive with £370k.

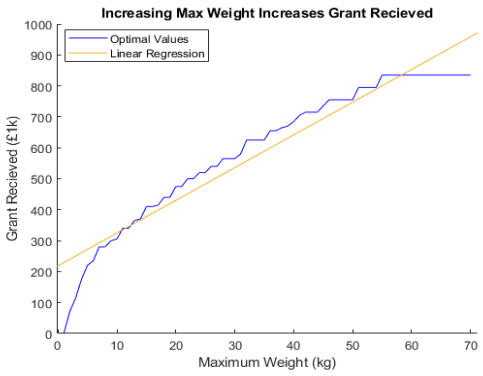
3.2 Optimising Maximum Weight

This sections results can be generated by running *KY1_weightvarying.m* with *getOptimalSolution.m* in the same directory.

For HEARG to generate the highest profit from the grants received the weight that the balloon can carry must be analysed. In Figure.1a the grant amount received is plotted against balloon weight, and in Figure.1b it is clear that there is a relationship between the maximum weight of the balloon and the maximum grant received possible.



(a) Optimal Maximum Weight, W=14



(b) Optimal Maximum Weight Linear Fit

Figure 1: Varying Maximum Weight

Plotted on Figure.1b is a red circle, this is at $A = (x, y) = (14, 370)$. This is the optimal solution discussed above. If HEARG was able to launch a balloon capable of carrying more weight the maximum grant received would increase. If launch costs stayed constant regardless of weight it would make sense to launch a balloon capable of carrying 55kg, as this would yield the total maximum of £835k, this solution requires taking all experiments and additionally 13 batteries. Using the residual values from Figure.2, it could be very beneficial to increase the weight capacity to 15kg, where the actual grant received is 24% higher than the trend. However the trend could be somewhat skewed due to the bottom and top ends of the data. Another local max optimal weight would be 32kg, $Z = 625$ at this point the balloon is capable of taking experiments to increase the grant received by 22.4%, $31kg = 580, 30kg = 565$, as well as $33, 34, 35 = 625$. So

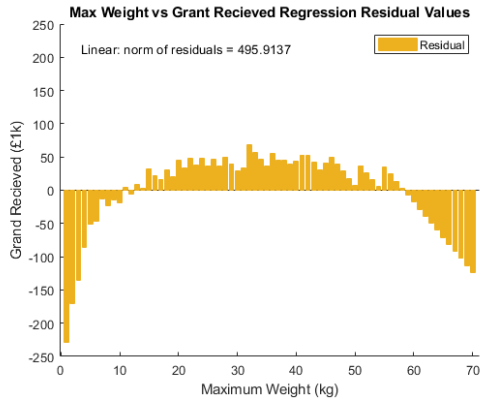


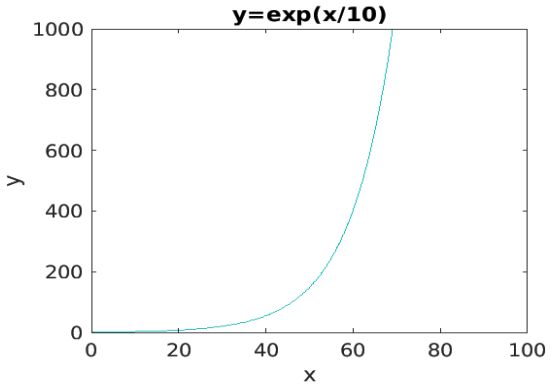
Figure 2: Optimal Maximum Weight Linear Fit Residuals

it would make sense to launch a balloon capable of this amount.

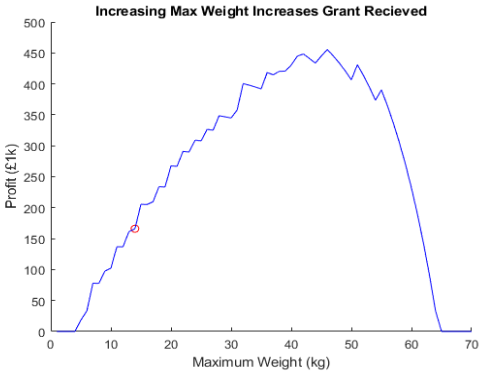
3.3 Optimising Weight Cost

This sections results can be generated by running *KY1_weightcost.m* with *getWeightSolution.m* in the same directory.

As discussed in the previous chapter, having unlimited weight allowance would require the balloon to be capable of carrying 55kg, which gives the solution $z = 835$, taking all experiments and also 13 batteries. Here the weight cost concept is introduced, a simple exponential function is used (Figure.3a). An exponential function was used as this best maps (at a very basic level) the cost of a launching a balloon into space, an improvement for next time would be to use $y = e^{\frac{x}{10}} + 20$, and make the exponential curve increase more dramatically around the centre of the objective value normal bell curve.



(a) Weight Cost Function



(b) Introducing Cost of Additional Weight

Figure 3: Optimising Weight Cost

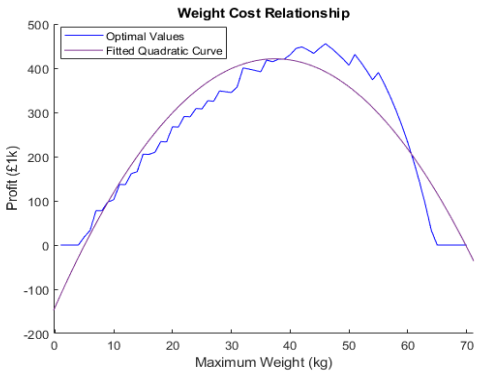
Using this weight cost model, gives an adjusted global maximum of 46kg, $Z = 455.52$. In this solution From Table.3 the solutions are for the same balloon weight, both taking all experiment apart from

Weight	Z	x-string	Batteries
46	755	1 1 1 1 1 1 1 1 1 1 0	10
46	455.52	1 1 1 1 1 1 1 1 1 1 0	10

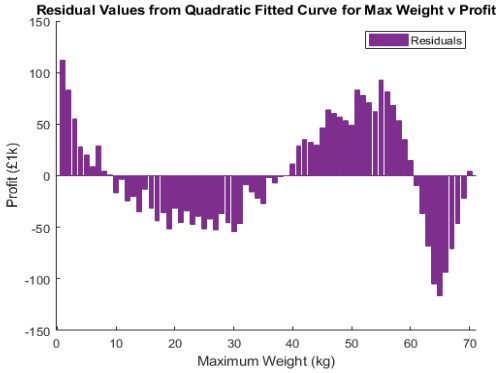
Table 3: Standard & Weight Cost

ECS Exotic Compounds Mass Spectrometer, and also 10 batteries. The difference in z values is from the weight cost of launching the balloon, which is included in Row 2, and is not considered in Row 1. Therefore for $w = 46$, Launch Cost = £299.48k. In Figure.3b the original problem is marked with a red circle at $(x, y) = (14, 165.94)$, this gives a launch cost of £213.06k. If it was not possible to launch the most optimal balloon, and MMU could only achieve a small increase on the current cost it would make financial sense to increase the balloon capacity to (20kg,£267.61k), allowing for an extra £101.67k. with the launch cost of £207.39k. For this weight allowance, an increase of only £6k launch costs allows for £95k increase in profit making it the most logical balloon capacity if not launching a balloon capable of 46kg. Additionally in Figure.4a where the quadratic fitted curve is displayed in purple, where $x = 20$

there is a local maximum $\frac{dy}{dx}$ which can also be seen in Figure.4b. Generally a good weight capacity will see a high $\Delta\frac{dy}{dx}$ for x values below the desired weight, and $\frac{dy}{dx} \leq 0$ for x values above. This represents a position of good cost efficiency for launching the experiments. Other local maxima's are $w = 36$,$w = 32$.



(a) Optimal Maximum Weight v Cost



(b) Optimal Maximum Weight v Cost Quadratic Fit Residuals

Figure 4: Optimising Weight Cost

$w = 32$ gives £400.97k, compared to $w = 31$ giving a objective value of £357.80k. The residual values are more helpful for the weight cost relationship than the standard varying weight problem, this is because the weight cost relationship requires a quadratic curve to accurately fit the data. Therefore this aids identifying potential optimal weights since when launch costs are included simply maximum profit is not always the most important aspect.

4 Conclusion

If HEARG wish to send a balloon capable of 14kg they should send the following experiments: UVS High Altitude UV Spectrometer, OPP Ozone Partial Pressure Machine, CFC Spectrometry Analyser,HDS Hydrocarbon Detection System, CRIS Cosmic Ray Imaging Spectrometer and GBD Gamma Burst Detector and 2 batteries, to give an optimal grant received value of £370k.

If HEARG are able to modify the balloon to be able to carry the maximum weight of 55kg, taking all experiments and 10 batteries. This is probably infeasible from a engineering perspective, so it would be best to have the capacity at 15kg or 32kg where the returns are £410k and £625k. A balloon with a capacity of 15kg would take OPP Ozone Partial Pressure Machine, CFC Spectrometry Analyser, HDS Hydrocarbon Detection System, CRIS Cosmic Ray Imaging Spectrometer, GBD Gamma Burst Detector and 2 batteries where as a 32kg capacity would take UVS High Altitude UV Spectrometer, OPP Ozone Partial Pressure Machine, CFC Spectrometry Analyser, NOX Nitrogen Oxide Concentration Logger, HDS Hydrocarbon Detection System, CRIS Cosmic Ray Imaging Spectrometer, GBD Gamma Burst Detector, HEP High Energy Particle Calorimeter and 6 batteries.

If HEARG require that costs of launch are included the maximum profit would be attained when launch-

ing a balloon capable of 46kg, taking all experiments apart from ECS Exotic Compounds Mass Spectrometer, and 10 batteries. However if it needs to be a budget launch, 20kg would make sense with very little launch costs and a far larger return than similar weight bands. This solution returns £267.61, and takes OPP Ozone Partial Pressure Machine, CFC Spectrometry Analyser, NOX Nitrogen Oxide Concentration Logger, HDS Hydrocarbon Detection System, CRIS Cosmic Ray Imaging Spectrometer, GBD Gamma Burst Detector and 3 batteries.

References

KY1-Reference (2019). Keith Yates. URL: `https://moodle.mmu.ac.uk/pluginfile.php/3758666/mod_assign/introattachment/0/KY1.pdf` (visited on 07/11/2019).

Taha, Hamdy A. (2017). *Operations Research: An Introduction*. Pearson Education Limited.

A Full Matlab Output

z	x-string
80	0 0 0 0 0 0 0 0 0 0 0 1 0
80	0 0 0 0 0 0 0 0 0 0 0 1 1
80	0 0 0 0 0 0 0 0 0 0 0 1 2
80	0 0 0 0 0 0 0 0 0 0 0 1 3
80	0 0 0 0 0 0 0 0 0 0 0 1 4
80	0 0 0 0 0 0 0 0 0 0 0 1 5
120	0 0 0 0 0 0 0 0 0 0 1 1 0
120	0 0 0 0 0 0 0 0 0 0 1 1 1
120	0 0 0 0 0 0 0 0 0 0 1 1 2
120	0 0 0 0 0 0 0 0 0 0 1 1 3
120	0 0 0 0 0 0 0 0 0 0 1 1 4
145	0 0 0 0 0 0 0 0 0 1 0 1 0
145	0 0 0 0 0 0 0 0 0 1 0 1 1
145	0 0 0 0 0 0 0 0 0 1 0 1 2
145	0 0 0 0 0 0 0 0 0 1 0 1 3
185	0 0 0 0 0 0 0 0 0 1 1 1 1
185	0 0 0 0 0 0 0 0 0 1 1 1 2
190	0 0 0 0 0 0 0 0 1 1 0 1 0
190	0 0 0 0 0 0 0 0 1 1 0 1 1
190	0 0 0 0 0 0 0 0 1 1 0 1 2
190	0 0 0 0 0 0 0 0 1 1 0 1 3
230	0 0 0 0 0 0 0 0 1 1 1 1 2
275	0 0 0 0 0 0 0 1 0 1 0 1 2
280	0 0 0 0 0 0 0 1 1 1 1 0 1
280	0 0 0 0 0 0 1 1 0 0 0 1 3
285	0 0 0 0 0 0 1 1 1 0 1 0 1
285	0 0 0 0 0 0 1 1 1 0 1 0 2
310	0 0 0 0 0 0 1 1 1 1 0 0 1
310	0 0 0 0 0 0 1 1 1 1 0 0 2
325	0 0 1 0 0 1 1 0 1 0 0 1
345	0 1 0 0 0 0 1 1 1 0 0 1
345	0 1 0 0 0 1 1 0 0 1 0 2
350	0 1 0 0 0 1 1 1 0 0 0 2
360	0 1 1 0 0 0 1 0 1 0 0 1
365	0 1 1 0 0 1 1 0 0 0 0 2
370	1 1 1 0 0 1 0 1 0 0 0 2

Table 4: Loop for Weight = 14

NB:0 Values omitted

B Matlab Code

B.1 KY1_weight14.m

```
% KY1 - Weight = 14
% Kieran Molloy

% OPTIONS
doPrint = false;

% Define Maximum Weight
maxWeight = 14;

% Solve LP
[val,mat] = getOptimalSolution(maxWeight,'single');

if doPrint
    % Print LP
    fprintf('\nSolution Found!\n')
    fprintf('Z = %7.2f\n',val)
    for i=1:size(mat,2)-1
        fprintf('x%2i | %1i\n',i,mat(i))
    end
    fprintf('Batt| %1i\n',mat(end))
end
```

B.2 KY1_weightvarying.m

```
% KY1 - Varying Max Weight
% Kieran Molloy

% OPTIONS
doPlot = true;
doPrint = false;

% Define Loop Max
loopMax=70;
% Optimal Objective Values
zVal=zeros(1,loopMax)';
% Optimal Decision Variables
dVal=zeros(12,loopMax)';

% Solves LP for 1 -> loopMax
for maxWeight=1:loopMax
    % Solve LP
```

```
    [val,mat] = getOptimalSolution(maxWeight,'single');

    % Add to Solution Matrices
    zVal(maxWeight,1)=val;
    dVal(maxWeight,:)=mat;
end

if doPrint
    % Print LP Optimal Values + Decision Variables
    fprintf('\nSolutions\n')
    fprintf('  Z ||  x1 |  x2 |  x3 |  x4 |  x5 |  x6 |  x7 |  x8 |  x9 | x10 | x11 || Batt |\n')
    fprintf('==========================================================================================\n')
    for i=1:size(zVal)
        fprintf('%3i || %3i | %3i | %3i | %3i | %3i | %3i | %3i | %3i | %3i | %3i || %4i |\n',zVal(i),dVal(i,1),dVal(i,2),dVal(i,3),dVal(i,4),dVal(i,5),dVal(i,6),dVal(i,7),dVal(i,8),dVal(i,9),dVal(i,10),dVal(i,11),dVal(i,12))
    end
end


if doPlot
    hold on
    % Plot Point of Max Weight = 14
    plot(14,zVal(14),'ro-')
    % Plot LP's Optimal Values
    plot(zVal(:),'b')
    % Labels & Annotations
    xlabel('Maximum Weight (kg)')
    ylabel('Grant Recieved (1k)')
    title('Increasing Max Weight Increases Grant Recieved')
    hold off
end
```

B.3 KY1_weightcost.m

```
% KY1 - Varying Max Weight with Weight Cost
% Kieran Molloy

% OPTIONS
doPlot = true;
doPrint = false;

% Define Loop Max
```

```
loopMax=70;
% Optimal Objective Values
zVal=zeros(1,loopMax)';
% Optimal Decision Variables
dVal=zeros(12,loopMax)';

% Solves LP for 1 -> loopMax
for maxWeight=1:loopMax
    % Solve LP
    [val,mat] = getWeightSolution(maxWeight,'single');

    % Add to Solution Matrices
    zVal(maxWeight,1)=val;
    dVal(maxWeight,:)=mat;
end

if doPrint
    % Print LP Optimal Values + Decision Variables
    fprintf('\nSolutions\n')
    fprintf('  Z ||  x1 |  x2 |  x3 |  x4 |  x5 |  x6 |  x7 |  x8 |  x9 | x10 | x11 ||
    Batt |\n')
    fprintf('
    ==========================================================================\n')
    for i=1:size(zVal)
        fprintf('%3i || %3i | %3i | %3i | %3i | %3i | %3i | %3i | %3i | %3i | %3i
    || %4i |\n',zVal(i),dVal(i,1),dVal(i,2),dVal(i,3),dVal(i,4),dVal(i,5),dVal(i,6),dVal
    (i,7),dVal(i,8),dVal(i,9),dVal(i,10),dVal(i,11),dVal(i,12))
    end
end


if doPlot
    hold on
    % Plot Point of Max Weight = 14
    plot(14,zVal(14),'ro-')
    % Plot LP's Optimal Values
    plot(zVal(:),'b')
    % Labels & Annotations
    xlabel('Maximum Weight (kg)')
    ylabel('Grant Recieved (1k)')
    title('Increasing Max Weight Increases Grant Recieved')
    hold off
end
```

C Solving Functions

C.1 getOptimalSolution.m

```
% KY1 - Function : Get Optimal Plan
% Kieran Molloy

%  INPUT:       w - Max Weight of Ship (double)
%          output - Output Method (string: {'single','path'})
% OUTPUT: solVal - Optimal Objective Variables (1xn Array)
%          solMat - Decision Variables relating to Optimal Obj (12xn Array)

function [solVal,solMat]=getOptimalSolution(w,output)

    % Initialisations from KY1 Specification
    % Maximum Weight of Ship
    maxW=w;
    % Maximum Current of Ship
    maxA = 45;
    % Weight of Additional Battery
    battW = 2;
    % Current of Additional Battery
    battA = 10;
    % Weight of Experiments
    weight=[3.3 2.1 1.8 4.2 2.4 1.5 3.6 1.3 3.6 1.9 2.8];
    % Current of Experiments
    current=[17 15 8 25 16 12 24 5 8 14 30];
    % Grant of Experiments
    grant=[90 105 60 90 60 70 130 45 65 40 80];

    % Solution Objective Value
    solVal=[];
    % Solution Decision Matrix
    solMat=[];

    % Objective Value Tracker
    MAXZ = 0;

    for x1=0:1
     for x2=0:1
      for x3=0:1
       for x4=0:1
        for x5=0:1
         for x6=0:1
          for x7=0:1
```

```
42          for x8=0:1
43           for x9=0:1
44            for x10=0:1
45             for x11=0:1
46              for y=0:20
47               if (addup(x1*weight(1),x2*weight(2),x3*weight(3),x4*weight(4),x5*weight
    (5),x6*weight(6),x7*weight(7),x8*weight(8),x9*weight(9),x10*weight(10),x11*weight
    (11))+y*battW) <= maxW
48                if (addup(x1*current(1),x2*current(2),x3*current(3),x4*current(4),x5*
    current(5),x6*current(6),x7*current(7),x8*current(8),x9*current(9),x10*current(10),
    x11*current(11))) <= (maxA + y*battA)
49                 z=addup(x1*grant(1),x2*grant(2),x3*grant(3),x4*grant(4),x5*grant(5),
    x6*grant(6),x7*grant(7),x8*grant(8),x9*grant(9),x10*grant(10),x11*grant(11));
50                 if z > MAXZ
51                  % Update highest Z
52                  MAXZ=z;
53
54                  if strcmp(output,'single')
55                      % Return Single (& FIRST) Optimal Solution
56                      solVal = z;
57                      solMat = [x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 y];
58                  elseif strcmp(output,'path')
59                      % Return Each Increment Optimal Solution Found
60                      solVal = [solVal ; z];
61                      solMat = [solMat ; x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 y];
62                  else
63                      %default is single solution
64                      solVal = z;
65                      solMat = [x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 y];
66                  end
67
68                 end
69               end
70              end
71             end
72            end
73           end
74          end
75         end
76        end
77       end
78      end
79     end
80    end
```

```
     end
    end

    % No optimal solution found
    if MAXZ==0
        solVal = 0;
        solMat = [0 0 0 0 0 0 0 0 0 0 0 0];
    end

end

% Helper Functions

function y=addup(f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11)
    % Adds up 11 values and returns solution
    y=f1+f2+f3+f4+f5+f6+f7+f8+f9+f10+f11;
end
```

C.2 getWeightSolution.m

```
% KY1 - Function : Get Optimal Plan
% Kieran Molloy

%  INPUT:       w - Max Weight of Ship (double)
%          output - Output Method (string: {'single','path'})
% OUTPUT: solVal - Optimal Objective Variables (1xn Array)
%          solMat - Decision Variables relating to Optimal Obj (12xn Array)

function [solVal,solMat]=getWeightSolution(w,output)

    % Initialisations from KY1 Specification
    % Maximum Weight of Ship
    maxW=w;
    % Maximum Current of Ship
    maxA = 45;
    % Weight of Additional Battery
    battW = 2;
    % Current of Additional Battery
    battA = 10;
    % Weight of Experiments
    weight=[3.3 2.1 1.8 4.2 2.4 1.5 3.6 1.3 3.6 1.9 2.8];
    % Current of Experiments
    current=[17 15 8 25 16 12 24 5 8 14 30];
    % Grant of Experiments
    grant=[90 105 60 90 60 70 130 45 65 40 80];
```

```
26
27  % Solution Objective Value
28  solVal=[];
29  % Solution Decision Matrix
30  solMat=[];
31
32  % Objective Value Tracker
33  MAXZ = 0;
34
35  for x1=0:1
36   for x2=0:1
37    for x3=0:1
38     for x4=0:1
39      for x5=0:1
40       for x6=0:1
41        for x7=0:1
42         for x8=0:1
43          for x9=0:1
44           for x10=0:1
45            for x11=0:1
46             for y=0:20
47              if (addup(x1*weight(1),x2*weight(2),x3*weight(3),x4*weight(4),x5*weight
    (5),x6*weight(6),x7*weight(7),x8*weight(8),x9*weight(9),x10*weight(10),x11*weight
    (11))+y*battW) <= maxW
48               if (addup(x1*current(1),x2*current(2),x3*current(3),x4*current(4),x5*
    current(5),x6*current(6),x7*current(7),x8*current(8),x9*current(9),x10*current(10),
    x11*current(11))) <= (maxA + y*battA)
49                z=addup(x1*grant(1),x2*grant(2),x3*grant(3),x4*grant(4),x5*grant(5),
    x6*grant(6),x7*grant(7),x8*grant(8),x9*grant(9),x10*grant(10),x11*grant(11))-
    costOfWeight(maxW);
50                 if z > MAXZ
51                  % Update highest Z
52                  MAXZ=z;
53
54                  if strcmp(output,'single')
55                      % Return Single (& FIRST) Optimal Solution
56                      solVal = z;
57                      solMat = [x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 y];
58                  elseif strcmp(output,'path')
59                      % Return Each Increment Optimal Solution Found
60                      solVal = [solVal ; z];
61                      solMat = [solMat ; x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 y];
62                  else
63                      %default is single solution
```

```
                        solVal = z;
                        solMat = [x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 y];
                    end

                end
            end
        end
    end
  end
 end
end

% No optimal solution found
if MAXZ==0
    solVal = 0;
    solMat = [0 0 0 0 0 0 0 0 0 0 0 0];
end

end

% Helper Functions

function y=addup(f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11)
    % Adds up 11 values and returns solution
    y=f1+f2+f3+f4+f5+f6+f7+f8+f9+f10+f11;
end


function y=costOfWeight(maxW)
    y=exp(maxW/10);
end
```