

# COSC364 Assignment 2

George Drummond(53243258),  
Ryan Cox(64656394)

May 29, 2017

## Abstract

The following pertains to COSC364-17S1 Assignment 2. This was a joint work by George Drummond and Ryan Cox in accordance with the course requirements of COSC364-17S1 and is a result of **equal contribution** (50%/50%).

## Contents

<b>1</b>	<b>Problem formulation</b>	<b>2</b>
1.1	Decision and auxiliary variables . . . . .	2
1.2	Objective function . . . . .	2
1.3	Demand constraints . . . . .	4
1.4	Additional constraints . . . . .	5
<b>2</b>	<b>Results</b>	<b>5</b>
<b>3</b>	<b>Discussion</b>	<b>5</b>
<b>4</b>	<b>Appendix</b>	<b>6</b>
4.1	LP generation source file . . . . .	6
4.2	LP File for $X = 3$ , $Y = 2$ and $Z = 4$ . . . . .	9

# 1 Problem formulation

We wish to formulate an optimization problem for generic values of  $X$ ,  $Y$  and  $Z$  (with  $Y > 3$ ) such that the load on all transit nodes is balanced. Here we outline the governing mathematical expressions concerning the objective function, the decision variables and all other constraints. The constraints generated by equations 1 through to 12 suffice to fully describe our optimisation problem.

*Notation:*  $[X] = \{1, 2, 3, \dots, X\}$ .

## 1.1 Decision and auxiliary variables

Let  $u_{ik}$  be the total amount of flow on the link between a given source node  $S_i$  and transit node  $T_k$ . Likewise let  $v_{kj}$  be the total amount of flow on the link between a given transit node  $T_k$  and destination node  $D_j$ .

Therefore, letting  $x_{ikj}$  be the part of the demand volume between source node  $S_i$  and destination node  $D_j$  that is routed through transit node  $T_k$ , we achieve

$$u_{ik} = \sum_{j=1}^Z x_{ikj} \quad (1)$$

$$v_{kj} = \sum_{i=1}^X x_{ikj} \quad (2)$$

$\forall i \in [X], k \in [Y], j \in [Z]$ .

Also, the total traffic flow into a transit node is equal to the total traffic flow out of the node, achieving the following balance.

$$\sum_{i=1}^X u_{ik} = \sum_{j=1}^Z v_{kj} \quad (3)$$

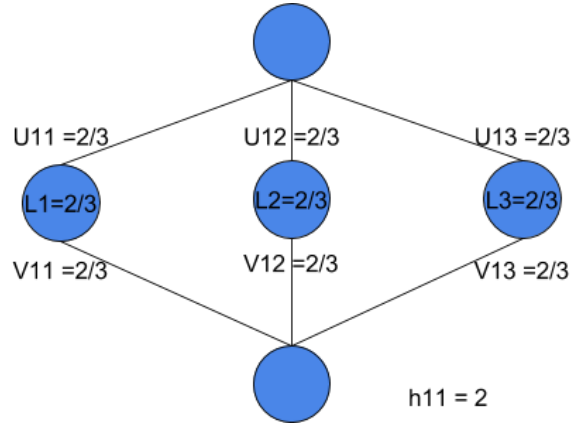
$\forall k \in [Y]$ .

## 1.2 Objective function

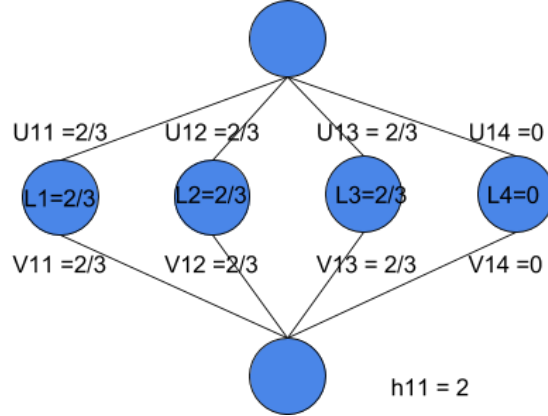
The goal of this linear program is to balance the load (total *incoming* traffic) across the transit nodes  $T_1, T_2, \dots, T_Y$ . As the load on a given transit node  $l_k$  is simply the sum of the flows from all source nodes to  $T_k$ , we achieve

$$l_k = \sum_{i=1}^X u_{ik}, \quad \forall k \in [Y]$$

As demonstrated in the small example of figure 1.

Figure 1:  $X=1, Y=4, Z=1$ 

At this stage, it may be tempting to define the objective function  $l$  as *equal* to all  $l_k$ ,  $k \in [Y]$  in order to achieve a common minimum load. This however, would actually lead to an infeasible problem for many cases such as shown in figure2 below in which all  $l_k$  cannot be equal.

Figure 2:  $X=1, Y=4, Z=1$ 

To *balance* this load across the transit nodes, we define our objective function  $l$  by

$$l = \max l_k = \max \sum_{i=1}^X u_{ik}, \quad \forall k \in [Y]$$

Proceeding in this fashion acts to minimise the greatest load on a transit node. The piecewise linear interpretation of this is as follows.

$$l \geq \sum_{i=1}^X u_{ik}, \quad \forall k \in [Y] \quad (4)$$

### 1.3 Demand constraints

We now turn our attention to demand constraints and the global requirement that each demand volume  $h_{ij}$  between nodes  $S_i$  and  $D_j$  shall be split over exactly three different paths, such that each path gets an equal share of the demand volume.

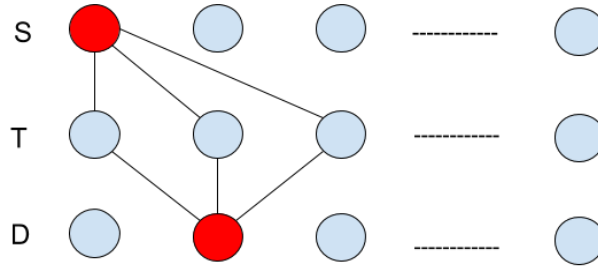


Figure 3: Splitting requirement

let  $w_{ikj}$  be the indicator variable for the path between source node  $S_i$  and destination node  $D_j$  through transit node  $T_k$ , taking the value 1 if path  $S_i -> T_k -> D_j$  carries data and 0 otherwise. With our 3 path restriction, we achieve

$$\sum_{k=1}^Y w_{ikj} = 3, \quad \forall i \in [X], j \in [Z] \quad (5)$$

Now to consider the demand volume between source node  $S_i$  and destination node  $D_j$ . Firstly, we have that this demand volume is simply the sum of the parts of the demand volume through each transit router  $T_k$ ,  $k \in [Y]$ .

$$\sum_{k=1}^Y x_{ikj} = h_{ij} = i + j$$

However, we also have that  $x_{ikj}$  will be non-zero if and only if there is flow on path  $S_i -> T_k -> D_j$  ( $w_{ikj} = 1$ ). As this occurs for precisely 3  $x_{ikj}$  (say  $x_{ik_1j}, x_{ik_2j}, x_{ik_3j}$ ) and the demand  $h_{ij}$  is shared *equally* across the paths that do possess flows, we achieve the following.

$$x_{ikj} = \begin{cases} \frac{(i+j)}{3}, & \text{if } w_{ikj} = 1 \\ 0, & \text{if } w_{ikj} = 0 \end{cases}$$

Or explicitly.

$$x_{ikj} = w_{ikj} \frac{h_{ij}}{3} = \frac{(i+j)}{3} w_{ikj}, \quad \forall i \in [X], k \in [Y], j \in [Z] \quad (6)$$

## 1.4 Additional constraints

Capacity:

$\forall i \in [X], k \in [Y], j \in [Z]$ , we have,

$$u_{ik} \leq c_{ik} \quad (7)$$

$$v_{kj} \leq d_{kj} \quad (8)$$

Non-negativity:

$\forall i \in [X], k \in [Y], j \in [Z]$ , we have,

$$x_{ikj} \geq 0 \quad (9)$$

$$u_{ik} \geq 0 \quad (10)$$

$$v_{kj} \geq 0 \quad (11)$$

$$l \geq 0 \quad (12)$$

## 2 Results

The following are the results for fixing  $X=7$ ,  $Z = 7$  and varying  $Y \in \{3, 4, 5, 6, 7\}$ .

Y	CPLEX run time (s)	Max load (units)	Max capacity (units)
3	0.014	131	25.7
4	0.102	98.0	25.7
5	0.047	78.7	25.7
6	0.082	65.3	23
7	0.173	56.0	16.7

## 3 Discussion

We can see that as we add more transit nodes between the source and destination nodes that generally the time taken to solve the LP file increases. Also, adding more transit nodes rightly reduces the maximum load. The highest capacity link seems to stay the same as we are only changing the amount of transit nodes the link capacity stays the same and only when we add more nodes does the capacity decrease

## 4 Appendix

### 4.1 LP generation source file

```

1  """
2  #genLP.py
3  #Authors: George Drummond – gmd44
4  #         Ryan Cox – rlc96
5  #Last Edit: 5/27/2017
6  #
7  #Given inputs X, Y, Z generates an LP file aiming to minimise the load on
   transit nodes
8  #
9  """
10
11
12 import sys
13
14 def print_aux(X,Y,Z):
15     """prints constraints pertaining to auxiliary variables """
16     aux = ""
17     for i in range(1,X+1):
18         for k in range(1,Y+1):
19             aux = "u{0}{1}".format(i,k)
20             for j in range(1,Z+1):
21                 aux += " - x{0}{1}{2}".format(i,k,j)
22             aux += " = 0\n"
23             print(aux,end="")
24
25
26     for j in range(1,Z+1):
27         for k in range(1,Y+1):
28             aux = "v{0}{1}".format(k,j)
29             for i in range(1,X+1):
30                 aux += " - x{0}{1}{2}".format(i,k,j)
31             aux += " = 0\n"
32             print(aux,end="")
33
34
35
36     for k in range(1,Y+1):
37         mysum = ""
38         for i in range(1,X+1):
39             mysum += "+ u{0}{1} ".format(i,k)
40         for j in range(1,Z+1):
41             mysum += "- v{0}{1} ".format(k,j)
42
43         mysum += "= 0"
44         print(mysum[2:])
45
46

```

```

47 def print_objective_constraints(X,Y,Z):
48     """prints constraints pertaining to objective function"""
49     for k in range(1,Y+1):
50         con = "l - "
51         for i in range(1,X+1):
52             con += "u{0}{1} - ".format(i,k)
53
54         con = con[:-2] + ">= 0"
55
56         print(con)
57
58 def print_demand(X,Y,Z):
59     """prints demand constraints"""
60     demand = ""
61     for i in range(1,X+1):
62         for j in range(1,Z+1):
63             mysum = ""
64             for k in range(1,Y+1):
65                 demand += "x{0}{1}{2} - {3} w{0}{1}{2} = 0\n".format(i,
66 k,j,(i+j)/3)
67                 mysum += "w{0}{1}{2} + ".format(i,k,j)
68
69             mysum = mysum[:-2] + "= "
70             mysum += "3"
71             demand += mysum + "\n"
72
73
74     print(demand, end="")
75
76
77 def print_capp(X,Y,Z):
78     """prints the capacity constraint of the links"""
79     capp = ""
80     for i in range(1,X+1):
81         for k in range(1,Y+1):
82             capp += "u{0}{1} - c{0}{1} <= 0\n".format(i,k)
83
84
85
86     for k in range(1,Y+1):
87         for j in range(1,Z+1):
88             capp += "v{0}{1} - d{0}{1} <= 0\n".format(k,j)
89
90
91     print(capp, end="")
92
93
94 def print_integer(X,Y,Z):
95     """prints integer declarations"""
96     integer = ""
97     for i in range(1,X+1):

```

```

98         for j in range(1,Z+1):
99             for k in range(1,Y+1):
100                 integer += "w{}{}{}\n".format(i,k,j)
101     print(integer,end="")
102
103 def print_nonneg(X,Y,Z):
104     """prints the constraint of a link being non negative"""
105     nonneg = ""
106     for i in range(1,X+1):
107         for k in range(1,Y+1):
108             nonneg += "0 <= u{}{}\n".format(i,k)
109             for j in range(1,Z+1):
110
111                 nonneg += "0 <= x{}{}{}\n".format(i,k,j)
112
113         for k in range(1,Y+1):
114             for j in range(1,Z+1):
115                 nonneg += "0 <= v{}{}\n".format(k,j)
116
117     nonneg += "0 <= l\n"
118
119     print(nonneg,end="")
120
121
122 def main():
123     """main function that gets inputs from terminal and runs them through
124     the printing functions"""
125     (X,Y,Z) = sys.argv[1:4]
126     X = int(X)
127     Y = int(Y)
128     Z = int(Z)
129
130     print("X = {}, Y = {}, Z = {}".format(X,Y,Z))
131     print("Minimize")
132     print(" l")
133     print("Subject to")
134     print_aux(X,Y,Z)
135     print_objective_constraints(X,Y,Z)
136     print_demand(X,Y,Z)
137     print_capp(X,Y,Z)
138     print("Bounds")
139     print_nonneg(X,Y,Z)
140     print("Integer")
141     print_integer(X,Y,Z)
142     print("End")
143
144
145
146 main()

```



## 4.2 LP File for $X = 3$ , $Y = 2$ and $Z = 4$

```

1 X = 3, Y = 2, Z = 4
2 Minimize
3   l
4 Subject to
5 u11 - x111 - x112 - x113 - x114 = 0
6 u12 - x121 - x122 - x123 - x124 = 0
7 u21 - x211 - x212 - x213 - x214 = 0
8 u22 - x221 - x222 - x223 - x224 = 0
9 u31 - x311 - x312 - x313 - x314 = 0
10 u32 - x321 - x322 - x323 - x324 = 0
11 v11 - x111 - x211 - x311 = 0
12 v21 - x121 - x221 - x321 = 0
13 v12 - x112 - x212 - x312 = 0
14 v22 - x122 - x222 - x322 = 0
15 v13 - x113 - x213 - x313 = 0
16 v23 - x123 - x223 - x323 = 0
17 v14 - x114 - x214 - x314 = 0
18 v24 - x124 - x224 - x324 = 0
19 u11 + u21 + u31 - v11 - v12 - v13 - v14 = 0
20 u12 + u22 + u32 - v21 - v22 - v23 - v24 = 0
21 l - u11 - u21 - u31 >= 0
22 l - u12 - u22 - u32 >= 0
23 x111 - 0.6666666666666666 w111 = 0
24 x121 - 0.6666666666666666 w121 = 0
25 w111 + w121 = 3
26 x112 - 1.0 w112 = 0
27 x122 - 1.0 w122 = 0
28 w112 + w122 = 3
29 x113 - 1.3333333333333333 w113 = 0
30 x123 - 1.3333333333333333 w123 = 0
31 w113 + w123 = 3
32 x114 - 1.6666666666666667 w114 = 0
33 x124 - 1.6666666666666667 w124 = 0
34 w114 + w124 = 3
35 x211 - 1.0 w211 = 0
36 x221 - 1.0 w221 = 0
37 w211 + w221 = 3
38 x212 - 1.3333333333333333 w212 = 0
39 x222 - 1.3333333333333333 w222 = 0
40 w212 + w222 = 3
41 x213 - 1.6666666666666667 w213 = 0
42 x223 - 1.6666666666666667 w223 = 0
43 w213 + w223 = 3
44 x214 - 2.0 w214 = 0
45 x224 - 2.0 w224 = 0
46 w214 + w224 = 3
47 x311 - 1.3333333333333333 w311 = 0
48 x321 - 1.3333333333333333 w321 = 0
49 w311 + w321 = 3
50 x312 - 1.6666666666666667 w312 = 0

```

```

51 x322 - 1.6666666666666667 w322 = 0
52 w312 + w322 = 3
53 x313 - 2.0 w313 = 0
54 x323 - 2.0 w323 = 0
55 w313 + w323 = 3
56 x314 - 2.3333333333333335 w314 = 0
57 x324 - 2.3333333333333335 w324 = 0
58 w314 + w324 = 3
59 u11 - c11 <= 0
60 u12 - c12 <= 0
61 u21 - c21 <= 0
62 u22 - c22 <= 0
63 u31 - c31 <= 0
64 u32 - c32 <= 0
65 v11 - d11 <= 0
66 v12 - d12 <= 0
67 v13 - d13 <= 0
68 v14 - d14 <= 0
69 v21 - d21 <= 0
70 v22 - d22 <= 0
71 v23 - d23 <= 0
72 v24 - d24 <= 0
73 Bounds
74 0 <= u11
75 0 <= x111
76 0 <= x112
77 0 <= x113
78 0 <= x114
79 0 <= u12
80 0 <= x121
81 0 <= x122
82 0 <= x123
83 0 <= x124
84 0 <= u21
85 0 <= x211
86 0 <= x212
87 0 <= x213
88 0 <= x214
89 0 <= u22
90 0 <= x221
91 0 <= x222
92 0 <= x223
93 0 <= x224
94 0 <= u31
95 0 <= x311
96 0 <= x312
97 0 <= x313
98 0 <= x314
99 0 <= u32
100 0 <= x321
101 0 <= x322
102 0 <= x323

```

```
103 0 <= x324
104 0 <= v11
105 0 <= v12
106 0 <= v13
107 0 <= v14
108 0 <= v21
109 0 <= v22
110 0 <= v23
111 0 <= v24
112 0 <= 1
113 Integer
114 w111
115 w121
116 w112
117 w122
118 w113
119 w123
120 w114
121 w124
122 w211
123 w221
124 w212
125 w222
126 w213
127 w223
128 w214
129 w224
130 w311
131 w321
132 w312
133 w322
134 w313
135 w323
136 w314
137 w324
138 End
```