COSC364 Assignment 2

George Drummond(53243258),
Ryan Cox(64656394)

May 28, 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

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

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

$$u_{ik} = \sum_{j=1}^{Z} x_{ikj} \tag{1}$$

$$v_{kj} = \sum_{i=1}^{X} x_{ikj} \tag{2}$$

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

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

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

$\forall k \in [Y]$.

1.2 Objective function

The goal of this linear program is to balance the load (total *incoming* traffic) across the transit nodes $T_1, T_2, ..., T_Y$. 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.

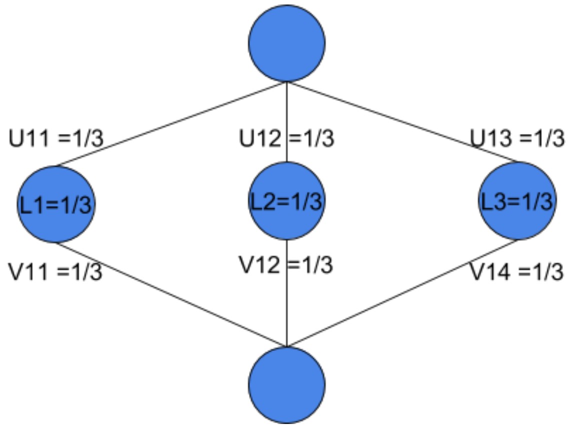

Figure 1: X=1,Y=4,Z=1

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 certain cases such as shown in figure2 below in which all l_k cannot be equal.

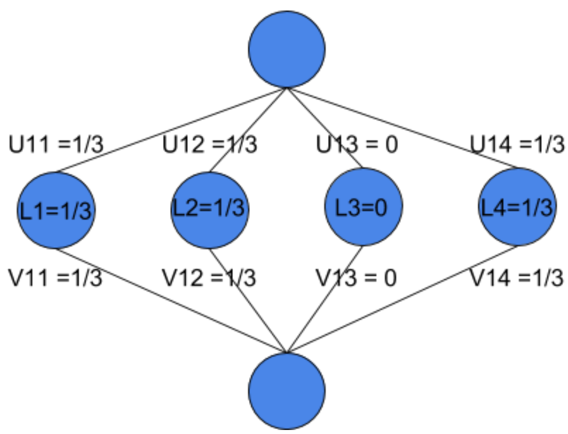

Figure 2: X=1,Y=4,Z=1

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.

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

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.

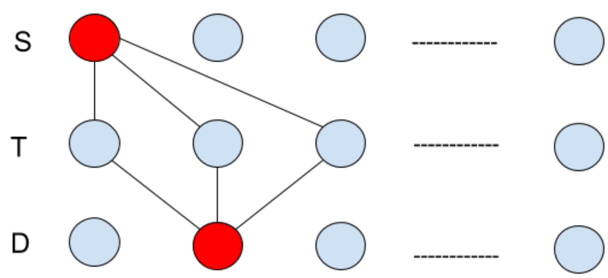

Figure 3: Splitting requirement

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 $S_i -> T_k -> Z_j$ carries data and 0 otherwise. With our 3 path restriction, we achieve

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

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 $T_k, \quad k \in [Y]$.

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

However, we also have that x_{ikj} will be non-zero if and only if there is flow on path $S_i -> T_k -> Z_j$ ($w_{ikj} = 1$). 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.

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

Or explicitly.

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

1.4 Additional constraints

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

$$u_{ik} \leq c_{ik} \tag{7}$$
$$v_{kj} \leq d_{kj} \tag{8}$$

Non-negativity:
$\forall i \in [X], k \in [Y], j \in [Z]$, we have,

$$x_{ikj} \geq 0 \tag{9}$$
$$u_{ik} \geq 0 \tag{10}$$
$$v_{kj} \geq 0 \tag{11}$$
$$l \geq 0 \tag{12}$$

2 Results

3 Appendix

3.1 LP generation source file

```python
"""
#genLP.py
#Authors: George Drummond - gmd44
#         Ryan Cox - rlc96
#Last Edit: 5/27/2017
#
#Given inputs from terminal generates an LP file with it's
#set constraints that can be run by CPLEX to get a minimised solution
#
"""


import sys

def print_aux(X,Y,Z):
    """prints """
    aux = ""
    for i in range(1,X+1):
        for k in range(1,Y+1):
            aux = "u{}{}".format(i,k)
            for j in range(1,Z+1):
                aux += " - x{0}{1}{2}".format(i,k,j)
            aux += " = 0\n"
            print(aux,end="")


    for j in range(1,Z+1):
        for k in range(1,Y+1):
            aux = "v{}{}".format(k,j)
            for i in range(1,X+1):
                aux += " - x{0}{1}{2}".format(i,k,j)
            aux += " = 0\n"
            print(aux,end="")




    for k in range(1,Y+1):
        mysum = ""
        for i in range(1,X+1):
            mysum += "+ u{}{} ".format(i,k)
        for j in range(1,Z+1):
            mysum += "- v{}{} ".format(k,j)

        mysum += "= 0"
        print(mysum[2:])

```

```
def print_objective_constraints(X,Y,Z):
    """prints """
    for k in range(1,Y+1):
        con = "l - "
        for i in range(1,X+1):
            con += "u{}{} - ".format(i,k)

        con = con[:-2] + ">= 0"

        print(con)

def print_demand(X,Y,Z):
    '''prints demand constraints'''
    demand = ""
    for i in range(1,X+1):
        for j in range(1,Z+1):
            mysum =  ""
            for k in range(1,Y+1):
                demand += "x{0}{1}{2} - {3} w{0}{1}{2} = 0\n".format(i,k,j,(i+j)/3)
                mysum += "w{}{}{} + ".format(i,k,j)



            mysum = mysum[:-2] + "= "


            mysum += "3"


            demand += mysum + "\n"



    print(demand,end="")


def print_capp(X,Y,Z):
    """prints the capacity contraint of the links"""
    capp = ""
    for i in range(1,X+1):
        for k in range(1,Y+1):
            capp += "u{0}{1} - c{0}{1} <= 0\n".format(i,k)



    for k in range(1,Y+1):
        for j in range(1,Z+1):
            capp += "v{0}{1} - d{0}{1} <= 0\n".format(k,j)


```

```
    print(capp,end="")


def print_integer(X,Y,Z):
    """"""
    integer = ""
    for i in range(1,X+1):
        for j in range(1,Z+1):
            for k in range(1,Y+1):
                integer += "w{}{}{}\n".format(i,k,j)
    print(integer,end="")

def print_nonneg(X,Y,Z):
    """prints the constraint of a link being non negitive"""
    nonneg = ""
    for i in range(1,X+1):
        for k in range(1,Y+1):
            nonneg += "0 <= u{}{}\n".format(i,k)
            for j in range(1,Z+1):

                nonneg += "0 <= x{}{}{}\n".format(i,k,j)

    for k in range(1,Y+1):
        for j in range(1,Z+1):
            nonneg += "0 <= v{}{}\n".format(k,j)

    nonneg += "0 <= l\n"

    print(nonneg,end="")


def main():
    """main function that gets inputs from terminal and runs them through
    the printing functions"""
    (X,Y,Z) = sys.argv[1:4]
    X = int(X)
    Y = int(Y)
    Z = int(Z)

    print("X = {}, Y = {}, Z = {}".format(X,Y,Z))
    print("Minimize")
    print(" l")
    print("Subject to")
    print_aux(X,Y,Z)
    print_objective_constraints(X,Y,Z)
    print_demand(X,Y,Z)
    print_capp(X,Y,Z)
    print("Bounds")
    print_nonneg(X,Y,Z)
    print("Integer")
    print_integer(X,Y,Z)
    print("End")
```

```


main()
```

3.2 LP File for X = 3, Y = 2 and Z = 4

```
X = 3, Y = 2, Z = 4
Minimize
 l
Subject to
u11 - x111 - x112 - x113 - x114 = 0
u12 - x121 - x122 - x123 - x124 = 0
u21 - x211 - x212 - x213 - x214 = 0
u22 - x221 - x222 - x223 - x224 = 0
u31 - x311 - x312 - x313 - x314 = 0
u32 - x321 - x322 - x323 - x324 = 0
v11 - x111 - x211 - x311 = 0
v21 - x121 - x221 - x321 = 0
v12 - x112 - x212 - x312 = 0
v22 - x122 - x222 - x322 = 0
v13 - x113 - x213 - x313 = 0
v23 - x123 - x223 - x323 = 0
v14 - x114 - x214 - x314 = 0
v24 - x124 - x224 - x324 = 0
u11 + u21 + u31 - v11 - v12 - v13 - v14 = 0
u12 + u22 + u32 - v21 - v22 - v23 - v24 = 0
l - u11 - u21 - u31 >= 0
l - u12 - u22 - u32 >= 0
x111 - 0.6666666666666666 w111 = 0
x121 - 0.6666666666666666 w121 = 0
w111 + w121 = 3
x112 - 1.0 w112 = 0
x122 - 1.0 w122 = 0
w112 + w122 = 3
x113 - 1.3333333333333333 w113 = 0
x123 - 1.3333333333333333 w123 = 0
w113 + w123 = 3
x114 - 1.6666666666666667 w114 = 0
x124 - 1.6666666666666667 w124 = 0
w114 + w124 = 3
x211 - 1.0 w211 = 0
x221 - 1.0 w221 = 0
w211 + w221 = 3
x212 - 1.3333333333333333 w212 = 0
x222 - 1.3333333333333333 w222 = 0
w212 + w222 = 3
x213 - 1.6666666666666667 w213 = 0
x223 - 1.6666666666666667 w223 = 0
w213 + w223 = 3
x214 - 2.0 w214 = 0
x224 - 2.0 w224 = 0
w214 + w224 = 3
x311 - 1.3333333333333333 w311 = 0
x321 - 1.3333333333333333 w321 = 0
w311 + w321 = 3
x312 - 1.6666666666666667 w312 = 0
```

```
x322 - 1.6666666666666667 w322 = 0
w312 + w322 = 3
x313 - 2.0 w313 = 0
x323 - 2.0 w323 = 0
w313 + w323 = 3
x314 - 2.3333333333333335 w314 = 0
x324 - 2.3333333333333335 w324 = 0
w314 + w324 = 3
u11 - c11 <= 0
u12 - c12 <= 0
u21 - c21 <= 0
u22 - c22 <= 0
u31 - c31 <= 0
u32 - c32 <= 0
v11 - d11 <= 0
v12 - d12 <= 0
v13 - d13 <= 0
v14 - d14 <= 0
v21 - d21 <= 0
v22 - d22 <= 0
v23 - d23 <= 0
v24 - d24 <= 0
Bounds
0 <= u11
0 <= x111
0 <= x112
0 <= x113
0 <= x114
0 <= u12
0 <= x121
0 <= x122
0 <= x123
0 <= x124
0 <= u21
0 <= x211
0 <= x212
0 <= x213
0 <= x214
0 <= u22
0 <= x221
0 <= x222
0 <= x223
0 <= x224
0 <= u31
0 <= x311
0 <= x312
0 <= x313
0 <= x314
0 <= u32
0 <= x321
0 <= x322
0 <= x323
```

```
0 <= x324
0 <= v11
0 <= v12
0 <= v13
0 <= v14
0 <= v21
0 <= v22
0 <= v23
0 <= v24
0 <= l
Integer
w111
w121
w112
w122
w113
w123
w114
w124
w211
w221
w212
w222
w213
w223
w214
w224
w311
w321
w312
w322
w313
w323
w314
w324
End
```