# COSC364 Assignment 2

George Drummond(53243258),
Ryan Cox(64656394)

May 27, 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.

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

## 1.1 Decision and auxiliary variables

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

$$u_{ik} + v_{kj} = x_{ikj} \tag{1}$$

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

Also, the total traffic flow into a transit node is equal to the total traffic flow out of the node, achieving the following balance $\forall k \in [Y]$.

$$\sum_{i=1}^{X} u_{ik} = \sum_{j=1}^{Z} v_{kj} \tag{2}$$

## 1.2 Objective function

The objective 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

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

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

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

As we wish to minimise the greatest load on a transit node.

This will therefore require Y constraint equations of the form.

$$l \geq \sum_{i=1}^{X} u_{ik}, \quad \forall k \in [Y] \tag{3}$$

## 1.3 Demand constraints

We now turn our attention to the global requirement that each demand volume shall be split over exactly three different paths, such that each path gets an equal share of the demand volume.

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

$$\sum_{k=1}^{Y} w_{ikj} = 3, \quad \forall i \in [X], j \in [Z] \tag{4}$$

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 $x_{ikj}$ and this demand is shared *equally* we achieve the following.

$$x_{ikj} = w_{ikj}\frac{h_{ij}}{3} = \frac{(i+j)}{3}w_{ikj}, \quad \forall i \in [X], k \in [Y], j \in [Z] \tag{5}$$

PROOF OF OPERATION HERE

## 1.4 Additional constraints

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

$$u_{ik} \leq c_{ik} \tag{6}$$
$$v_{kj} \leq d_{kj} \tag{7}$$

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

$$x_{ikj} \geq 0 \tag{8}$$
$$u_{ik} \geq 0 \tag{9}$$
$$v_{kj} \geq 0 \tag{10}$$
$$l \geq 0 \tag{11}$$

### 1.5 Model summary

The constraints generated by equations 1 through to 11 suffice to fully describe our optimisation problem.

## 2 Results

# 3 LP generation source file

```
import sys


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

     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):
     for k in range(1,Y+1):
          con = "1 - "
          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):
               mysum1 = mysum2 = mysum3 = ""
               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)
                    mysum1 += "w{}{}{} + ".format(i,k,j)
                    mysum2 += "w{0}{1}{2} x{0}{1}{2} + ".format(i,k,j)
                    mysum3 += "x{}{}{} + ".format(i,k,j)


               mysum1 = mysum1[:-2] + "= "
```

```
                mysum2 = mysum2[:-2] + "= "
                mysum3 = mysum3[:-2] + "= "

                mysum1 += "3"
                mysum2 += str(i + j)
                mysum3 += str(i + j)

                demand += mysum1 + "\n" #+ mysum2 + "\n" + mysum3+ "\n"



    print(demand,end="")


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

def print_nonneg(X,Y,Z):
    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"
```

```
100
101         print(nonneg,end="")
102
103
104 def main():
105         (X,Y,Z) = sys.argv[1:4]
106         X = int(X)
107         Y = int(Y)
108         Z = int(Z)
109
110         print("X = {}, Y = {}, Z = {}".format(X,Y,Z))
111         print("Minimize")
112         print(" l")
113         print("Subject to")
114         print_aux(X,Y,Z)
115         print_objective_constraints(X,Y,Z)
116         print_demand(X,Y,Z)
117         print_capp(X,Y,Z)
118         print("Bounds")
119         print_nonneg(X,Y,Z)
120         print("Integer")
121         print_integer(X,Y,Z)
122         print("End")
123
124
125
126 main()
```

# 4 References