COSC364 Assignment 2

George Drummond(53243258),
Ryan Cox(64656394)

May 27, 2017

Abstract

The following discusses ——- 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 **2**
1.1 Decision and auxiliary variables 2
1.2 objective function 2
1.3 Demand constraints 3
1.4 Additional constraints 3
1.5 Model summary 4

2 Results **4**

3 LP generation source file **5**

4 References **8**

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.

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

$\forall k \in [Y]$

1.2 objective function

The resulting *utilisation* on a link between a given source node S_i and transit node T_k is given by $\frac{u_{ik}}{c_{ik}}$. Similarly, the resulting utilisation on a link between a given transit node T_k and destination node $D_j$is given by $\frac{v_{kj}}{d_{kj}}$.

We therefore formulate our objective function as.

$$r = \max(\frac{u_{ik}}{c_{ik}}, \frac{v_{kj}}{d_{kj}}), \forall i \in [X], k \in [Y], j \in [Z]$$

That is, we wish to minimise the greatest link utilisation. We now note that r is piecewise linear in section 1.4 when considering related constraints.

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 i and destination node j through transit node k, taking the value 1 if path $i- > k- > j$ carries data and 0 otherwise. With our 3 path restriction, we achieve

$$\sum_{k=1}^{Y} w_{ikj} = 3 \tag{3}$$

Now to consider the demand volume between nodes i and j.

$$x_{ikj} = w_{ikj}\frac{h_{ij}}{3} = \frac{(i+j)}{3}w_{ikj} \tag{4}$$

PROOF OF OPERATION HERE

1.4 Additional constraints

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

$$u_{ik} \leq c_{ik} \tag{5}$$
$$v_{kj} \leq d_{kj} \tag{6}$$

Likewise, from out discussion in 1.2, we have the following constraints regarding our objective function,

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

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

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

1.5 Model summary

Equations 1 through to 12 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="")


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):
                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[:-1] + "= "
            mysum2 = mysum2[:-1] + "= "
            mysum3 = mysum3[:-1] + "= "

            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)
            capp += "u{0}{1} - c{0}{1} r <= 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)
            capp += "v{0}{1} - d{0}{1} r <= 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 <= r\n"

    print(nonneg,end="")


def main():
    (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(" r")
    print("Subject to")
    print_aux(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()
```

4 References