UNIVERSITÉ PARIS 6, MASTER ANDROIDE

Projet MOGPL

Olivier Bachollet, Arthur Ramolet

30 novembre 2014

TABLE DES MATIÈRES

1 Première modélisation du problème **3**
1.1 Modélisation du problème 3
1.2 Essais numériques 4

2 Approche égalitariste **5**
2.1 L'approche flot max 5
2.2 L'approche programme linéaire 6
2.3 Essais numériques 6
2.4 Approche un peu moins égalitariste 6

3 Approche égalitariste en regrets **9**
3.1 Regrets en programme linéaire 9
3.2 Modélisation du problème sous forme de flot max 9

4 Extension á l'affectation multiple **10**

1 Première modélisation du problème

1.1 Modélisation du problème

Listing 1 – Programme linéaire P0 avec pygurobi

```
# -*- coding: utf-8 -*-
"""
Created on Wed Nov 19 15:19:03 2014

@author: arthur
"""

from gurobipy import *
import numpy as np

def genUtils(M,N):
    u = np.ones((M,N))

    for i in range(N):
        for j in range(M):
            u[i][j]=np.round(np.random.triangular(0,M/2,M))

    return u

#N = np.random.random_integers(5,15)
N = 3
M = N

u = genUtils(M,N)
x = []
u = [[2,0,0],[0,1,0],[0,3,1]]
print u


m = Model("prjMogpl")

for i in range(N):
    tmp = []
    for j in range(M):
        name = "x"+str(i)+","+str(j)
        tmp.append(m.addVar(vtype=GRB.BINARY, name=name))
    x.append(tmp)

m.update()

obj = LinExpr()
obj = 0

for i in range(N):
```

```
    for j in range(M):
        obj += u[i][j]*x[i][j]

print "obj : ",obj

m.setObjective(obj,GRB.MAXIMIZE)

for i in range(N):
    m.addConstr(quicksum(x[i][j] for j in range(M))==1,
                                    "contrainte%d" % i)

for j in range(M):
    m.addConstr(quicksum(x[i][j] for i in range(N))==1,
                                    "contrainte%d" % (N+j))

m.optimize()

print ""
print "Liste des objets :"
print u
print 'Solution optimale : '
for i in range(N):
    for j in range(M):
        print 'x'+str(i)+str(j), '=', x[i][j].x
print ""
print 'Valeur de la fonction objectif :', m.objVal
```

1.2 Essais numériques

2 Approche égalitariste

2.1 L'approche flot max

Listing 2 – Résolution problème flot max

```
# -*- coding: utf-8 -*-
"""
Created on Wed Nov 26 20:08:37 2014

@author: arthur
"""

from pygraph.classes.graph import graph
from pygraph.classes.digraph import digraph

from pygraph.algorithms.minmax import maximum_flow

import numpy as np

def genUtils(M,N):
    u = np.ones((M,N))

    for i in range(N):
        for j in range(M):
            u[i][j]=np.round(np.random.triangular(0,M/2,M))

    return u

#N = np.random.random_integers(5,15)
N = 3
M = N
u = genUtils(M,N)
x = []
#u = [[2,0,0],[0,3,1],[0,1,0]]
lmb = 0
sortie = 1
#print u

# Graph creation
while(sortie):
    gr = digraph()

    gr.add_nodes([0])
    gr.add_nodes([N+M+1])

    for i in range(N):
        gr.add_nodes([i+1])
        gr.add_edge((0,i+1), wt=1)
```

```
    for i in range(M):
        gr.add_nodes([N+i+1])
        gr.add_edge((N+i+1,N+M+1), wt=1)

    for i in range(N):
        for j in range(M):
            if(u[i][j]>=lmb):
                gr.add_edge((i+1,N+j+1), wt=1)
            else:
                gr.add_edge((i+1,N+j+1), wt=0)

    flows, cuts = maximum_flow(gr, 0, N+M+1)



    for i in range(N):
        k=0
        for j in range(M):
            if(flows[(i+1,N+j+1)]==0):
                k+=1
        if(k==N):
            sortie = 0

    if(sortie == 1):
        oldflow = flows
    lmb += 1

print u
print oldflow
```

2.2 L'APPROCHE PROGRAMME LINÉAIRE

2.3 ESSAIS NUMÉRIQUES

2.4 APPROCHE UN PEU MOINS ÉGALITARISTE

Listing 3 – Programme linéaire P1 avec pygurobi

```
# -*- coding: utf-8 -*-
"""
Created on Wed Nov 19 15:19:03 2014

@author: arthur
"""

from gurobipy import *
import numpy as np

def genUtils(M,N):
```

```
    u = np.ones((M,N))

    for i in range(N):
        for j in range(M):
            u[i][j]=np.round(np.random.triangular(0,M/2,M))

    return u

#N = np.random.random_integers(5,15)
N = 3
M = N
e=0.00001
y = 16
u = genUtils(M,N)
x = []
u = [[2,0,0],[0,1,0],[0,3,1]]
print u

n = Model("prjMogpl2")

for i in range(N):
    tmp = []
    for j in range(M):
        name = "x"+str(i)+","+str(j)
        tmp.append(n.addVar(vtype=GRB.BINARY, name=name))
    x.append(tmp)

y = n.addVar(vtype=GRB.CONTINUOUS, name="y")

n.update()

obj = LinExpr()
obj = 0

obj += y
for i in range(N):
    for j in range(M):
        obj +=e*u[i][j]*x[i][j]

print "obj : ",y

n.setObjective(obj,GRB.MAXIMIZE)

for i in range(N):
    n.addConstr(quicksum(x[i][j]*u[i][j] for j in
                range(M))-y>=0,"contrainte%d" % i)

for j in range(M):
    n.addConstr(quicksum(x[i][j] for i in
```

```
                range(N))==1,"contrainte%d" % (N+j))

n.optimize()

print ""
print "Liste des objets :"
print u
print 'Solution optimale : '
for i in range(N):
    for j in range(M):
        print 'x'+str(i)+str(j), '=', x[i][j].x
print ""
print 'Valeur de la fonction objectif :', n.objVal
```

3 Approche égalitariste en regrets

3.1 Regrets en programme linéaire

3.2 Modélisation du problème sous forme de flot max

4 Extension á l'affectation multiple

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

$$\begin{aligned}(x+y)^3 &= (x+y)^2(x+y)\\ &= (x^2+2xy+y^2)(x+y)\\ &= (x^3+2x^2y+xy^2)+(x^2y+2xy^2+y^3)\\ &= x^3+3x^2y+3xy^2+y^3\end{aligned} \tag{4.1}$$

Phasellus viverra nulla ut metus varius laoreet. Quisque rutrum. Aenean imperdiet. Etiam ultricies nisi vel augue. Curabitur ullamcorper ultricies