final

July 21, 2020

1 final

1.1 Alireza Darvishi 96109674

1.1.1 3

loading data

[1]:
```
import cvxpy as cp
import json
import numpy as np
with open("Recon3D.json", "r") as file_handle:
    dictionary = json.load(file_handle);
    data = dictionary["reactions"];
    temp_name = [(data[i])["name"] for i in range(len(data))];
    b = [i for i in range(len(data)) if temp_name[i] == "Generic Human Biomass␣
↪Reaction"];
    b = b[0];
    order = [i for j in (range(b), range(b+1, len(data)), range(b,b+1)) for i␣
↪in j];
    name = [(data[i])["name"] for i in order];
    lower_bound = [(data[i])["lower_bound"] for i in order];
    upper_bound = [(data[i])["upper_bound"] for i in order];
    subsystem = [(data[i])["subsystem"] for i in order];
    metabolites = [(data[i])["metabolites"] for i in order];
    id = [((dictionary["metabolites"])[i])["id"] for i in␣
↪range(len(dictionary["metabolites"]))];
    S = np.zeros((len(id), len(metabolites)));
    for i in range(len(metabolites)):
        for j in range(len(id)):
            if id[j] in metabolites[i].keys():
                S[j, i] = metabolites[i][id[j]];
```

a)

[2]:
```
v = cp.Variable(len(upper_bound))
constraints = [upper_bound>=v,lower_bound<=v,S*v==0]
```

```
objective = cp.Maximize(v[-1])
problem = cp.Problem(objective,constraints)
problem.solve()
w = problem.value
print("wild v is:",round(w,2))
```

```
wild v is: 753.34
```

b)

[3]:
```
knockout_indx1 = []
for i in range(len(subsystem)):
    if("Transport, nuclear" in subsystem[i]):
        knockout_indx1.append(i)
knockout_indx1 = np.array(knockout_indx1)

knockout_problem1 = cp.Problem(objective,constraints+[v[knockout_indx1]==0])
knockout_problem1.solve()
print("change after knocking out Transport, nuclear:",(w-knockout_problem1.
 ↪value)/w)
print("diffrence is near 1 so these reactions were very important")
```

```
change after knocking out Transport, nuclear: 0.9999999999997521
diffrence is near 1 so these reactions were very important
```

[4]:
```
knockout_indx2 = []
for i in range(len(subsystem)):
    if("Fatty acid oxidation" in subsystem[i]):
        knockout_indx2.append(i)
knockout_indx2 = np.array(knockout_indx2)

knockout_problem2 = cp.Problem(objective,constraints+[v[knockout_indx2]==0])
knockout_problem2.solve()
print("change after knocking out Fatty acid oxidation:",(w-knockout_problem2.
 ↪value)/w)
print("diffrence is near 0 so these reactions were not important")
```

```
change after knocking out Fatty acid oxidation: -1.8715845223013645e-11
diffrence is near 0 so these reactions were not important
```

c)

[5]:
```
for indx in knockout_indx1:
    knockout_problem = cp.Problem(objective,constraints+[v[indx]==0])
    knockout_problem.solve()
    if((problem.value-knockout_problem.value)/problem.value>=0.02):
        print(name[indx])
```

```
DATP diffusion in nucleus
DGTP diffusion in nucleus
```

1.1.2 5

[6]:
```
w = cp.Variable(pos=True)
l = cp.Variable(pos=True)
objective_fn = 2*w*l+2*np.pi*w+2*l
constraints = [l<=2*w,w<=l,300*cp.inv_pos(l)<=w,10<=w,w<=20,20<=l,l<=30]
problem = cp.Problem(cp.Minimize(objective_fn), constraints)
problem.solve(gp=True)
print("mask is:",round(l.value,2),"in",round(w.value,2))
```

```
mask is: 24.49 in 12.25
```

1.1.3 6

[7]:
```
n=3
gamma = 2
u = 2
d = cp.Variable(n,pos=True)
k = cp.Variable(n,pos=True)
v = cp.Variable(n,pos=True)
landa = cp.Variable(1,pos=True)
G = np.ones((3,3))-np.identity(3)
D = cp.diag(d)
K = cp.diag(k)
A = D+K*G
constraints = [gamma*cp.sum(cp.inv_pos(d))+cp.sum(cp.inv_pos(k))<=u]
constraints += [cp.hstack([d[0],k[1]/2,k[2]/2])*v*cp.inv_pos(landa*v[0])<=1]
constraints += [cp.hstack([k[0]/2,d[1],k[2]/2])*v*cp.inv_pos(landa*v[1])<=1]
constraints += [cp.hstack([k[0]/2,k[2]/2,d[2]])*v*cp.inv_pos(landa*v[2])<=1]
objective = cp.Minimize(landa)
problem = cp.Problem(objective,constraints)
problem.solve(gp=True)
print("D is:\n",D.value)
print("K is:\n",K.value)
print("A is:\n",(D+K*G).value)
```

```
D is:
 [[4.95195719 0.         0.        ]
 [0.         4.99531724 0.        ]
 [0.         0.         5.2830707 ]]
K is:
 [[3.38863977 0.         0.        ]
 [0.         5.37683063 0.        ]
 [0.         0.         2.97539551]]
A is:
 [[4.95195719 3.38863977 3.38863977]
 [5.37683063 4.99531724 5.37683063]
 [2.97539551 2.97539551 5.2830707 ]]
```