

۱- کافی است $2n$ مسئله حل کنیم که در هر مسئله $f_i = e_i$ یا $f_i = -e_i$ است. اگر چند دومی می توان

باشد. یکی از $2n$ مسئله، جواب p^* دارد

$$LP_i: \max f^T x$$

$$st: Ax \leq b$$

$$Cx = d$$

$$f_i = \begin{cases} e_i & i = 1, \dots, n \\ -e_i & i = n+1, \dots, 2n \end{cases}$$

حال می توان تمام مسائل را در یک مسئله خلاصه کرد:

~~برای هر x برداری~~

بردار x را برداری تقریبی کنیم. از نسبت سرم قرار دادن x_1 تا x_{2n} درست آمده در x_1 تا

x_{2n} ، برداری n -تایی است حال:

$$\tilde{A} = \begin{pmatrix} A \\ A \\ A \end{pmatrix}$$

$$\tilde{b} = \begin{pmatrix} b \\ b \\ b \end{pmatrix}$$

$$\tilde{C} = \begin{pmatrix} C \\ C \\ C \end{pmatrix}$$

$$\tilde{f} = \begin{pmatrix} f_1 \\ \vdots \\ f_{2n} \end{pmatrix}$$

$$\tilde{d} = \begin{pmatrix} d \\ \vdots \\ d \end{pmatrix}$$

مسئله زیر را حل کنیم.

$$\max \tilde{f}^T x$$

$$st: \tilde{A}x \leq \tilde{b}$$

$$\tilde{C}x = \tilde{d}$$

اگر حداقل یکی از $2n$ مسئله ای که می توان باشد، مسئله ای که شده

هم می توان است.

2- از شرط مشتق دوم استفاده می کنیم.

$$\nabla^2 f_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

$$\frac{\partial^2 f}{\partial x_1^2} = \alpha_1(\alpha_1 - 1) \frac{f}{x_1^2}, \quad \frac{\partial^2 f}{\partial x_2^2} = \alpha_2(\alpha_2 - 1) \frac{f}{x_2^2}, \quad \frac{\partial^2 f}{\partial x_1 \partial x_2} = \alpha_1 \alpha_2 \frac{f}{x_1 x_2}$$

$$\nabla^2 f = \begin{pmatrix} \frac{\alpha_1(\alpha_1-1)f}{x_1^2} & \frac{\alpha_1\alpha_2 f}{x_1 x_2} \\ \frac{\alpha_1\alpha_2 f}{x_1 x_2} & \frac{\alpha_2(\alpha_2-1)f}{x_2^2} \end{pmatrix} = \frac{f}{x_1^2 x_2^2} \begin{pmatrix} \alpha_1(\alpha_1-1)x_2^2 & \alpha_1\alpha_2 x_1 x_2 \\ \alpha_1\alpha_2 x_1 x_2 & \alpha_2(\alpha_2-1)x_1^2 \end{pmatrix}$$

برای محدب بودن: باید ماتریس همبند مثبت معین باشد،
 $\nabla^2 f \succcurlyeq 0$

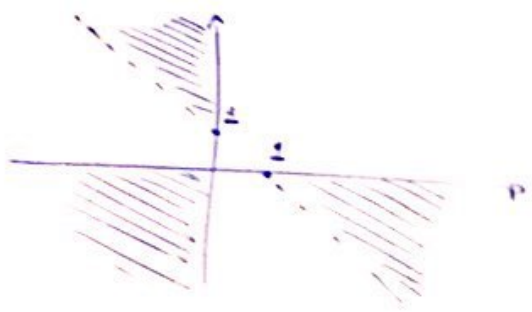
برای مثبت معین بودن ماتریس متقارن، کافی است دترمینان‌های گوشه‌ای اش مثبت معین

باشند پس

$$\begin{cases} \alpha_1(\alpha_1-1) \geq 0 \text{ و } \alpha_2^2 \geq 0 \\ [\alpha_1(\alpha_1-1)\alpha_2(\alpha_2-1) - \alpha_1^2\alpha_2^2] x_1^2 x_2^2 \geq 0 \end{cases}$$

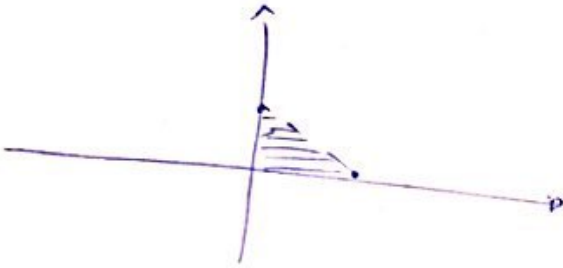
$$\alpha_1(\alpha_1-1) \geq 0, \quad \alpha_1\alpha_2[1-\alpha_1-\alpha_2] \geq 0$$

$$\Rightarrow \begin{cases} \alpha_1 \geq 1 \text{ و } \alpha_2 \leq 0, \quad \alpha_2 \geq 1-\alpha_1 \\ \alpha_1 \leq 0 \rightarrow \begin{cases} \alpha_2 \leq 0 \\ \alpha_2 \geq 1-\alpha_1 \end{cases} \end{cases}$$



$$\nabla^2 f \leq 0 \Rightarrow \alpha_1(\alpha_1 - 1) \leq 0, \alpha_1 \alpha_2 [1 - \alpha_1 - \alpha_2] \leq 0 \quad \text{شرط مقعر بودن:}$$

$$\alpha_1(\alpha_1 - 1) \leq 0 \Rightarrow 0 \leq \alpha_1 \leq 1 \Rightarrow 0 \leq \alpha_2 \leq 1 - \alpha_1$$



$$f_x = \prod_i x_i^{\alpha_i}$$

(e)

$$\frac{\partial f}{\partial x_i x_j} = \alpha_i \alpha_j \frac{f}{x_i x_j}, \quad \frac{\partial^2 f}{\partial x_i^2} = \alpha_i (\alpha_i - 1) \frac{f}{x_i^2}$$

$$\Rightarrow H = \nabla^2 f = \begin{pmatrix} \alpha_1(\alpha_1-1) \frac{1}{x_1^2} & \frac{\alpha_1 \alpha_2}{x_1 x_2} & \frac{\alpha_1 \alpha_3}{x_1 x_3} & \dots & \frac{\alpha_1 \alpha_n}{x_1 x_n} \\ \vdots & & & & \vdots \\ \frac{\alpha_1 \alpha_n}{x_1 x_n} & - & - & - & \frac{\alpha_n(\alpha_n-1)}{x_n^2} \end{pmatrix}$$

$$\nabla^2 f > 0 \Rightarrow \alpha_1(\alpha_1-1) > 0, \dots, \begin{vmatrix} \alpha_1(\alpha_1-1) & \alpha_1 \alpha_2 \\ \alpha_1 \alpha_2 & \alpha_2(\alpha_2-1) \end{vmatrix} > 0$$

$$\cdot \begin{vmatrix} \alpha_1(\alpha_1-1) & \alpha_1 \alpha_2 & \alpha_1 \alpha_3 \\ \alpha_1 \alpha_2 & \alpha_2(\alpha_2-1) & \alpha_2 \alpha_3 \\ \alpha_1 \alpha_3 & \alpha_2 \alpha_3 & \alpha_3(\alpha_3-1) \end{vmatrix} > 0, \dots$$

این مسئله، Strong alternative هم هستند. یعنی دگان مسئله feasibility ادل،

مسئله feasibility دم است. و با توجه به معرفت بودن قیودها و قضیه

$$SV=C \quad \left\{ \begin{array}{l} v \geq 0 \\ S^TH \leq 0 \end{array} \right\} \rightarrow \left\{ \begin{array}{l} S^TH \leq 0 \\ H^TC \leq 0 \end{array} \right.$$

دگانی قوی، گمراه صحیح است.

اما یک طرف راه را حتی می توان به طریق دیگری نشان داد:

$$\left. \begin{array}{l} SV=C \\ v \geq 0 \\ S^TH \leq 0 \end{array} \right\} \Rightarrow \left. \begin{array}{l} H^TSV = H^TC \\ S^TH \leq 0 \\ v \geq 0 \end{array} \right\} \Rightarrow H^TC \leq 0$$

ب) طبق قدرت سؤال اگر $SV > 0 \Rightarrow SV = 0$

حال تعریف می کنیم $b_i = \epsilon \epsilon_i$ و $\epsilon > 0$

در این صورت طبق فرض سؤال $SV > b_i$ جواب ندارد.

حال طبق هم فارکاش یکی از صورت جابین، $Ax \leq b$ جواب ندارد.

و تنها اگر $A=0$ و $b < 0$ باشد.

حال: $m_i^T(-S) = 0$ و $m_i^T(-b_i) < 0$ و $m_i = 0$ و $m_i^T(-S) = 0$ و $m_i^T(-b_i) < 0$ جواب ندارد.

پس: $m_i^TS = 0$ و متالغی از m_i الیه ثبت است.

حال n بار برای $i=1$ تا n این کار را انجام می دهیم و تعریف می کنیم $m = \sum m_i$

در این صورت m در گمراه های سؤال صدق می کند.

5. شرایطی سوال.

$$Lw' \leq 2$$

$$1 \leq Lw'$$

$$300 \leq L \times w$$

$$1 \leq w$$

$$w \leq 20$$

$$20 \leq L$$

$$L \leq 30$$

$$f_0 = 2L + \pi w + 2Lw$$

تابع هدف:

تمام شرایطی سوال را می توان به شکل $monomial$ تغییر داد. تابع هدف هم $Polynomial$

است. پس باید تغییر متغیر داریم: $y_1 = \log(L)$, $y_2 = \log(w)$

$$\left\{ \begin{array}{l} y_1 - y_2 - \log 2 \leq 0 \\ y_2 - y_1 \leq 0 \\ -y_1 - y_2 + \log(300) \leq 0 \\ -y_2 + \log(1) \leq 0 \\ y_2 - \log(20) \leq 0 \\ -y_1 + \log(20) \leq 0 \\ y_1 - \log(30) \leq 0 \end{array} \right. \rightarrow st$$

$$\min \log \left(e^{2y_1 + \log 2} + e^{2y_2 + \log \pi} + e^{2y_1 + y_2 + \log 2} \right)$$

6. الف)

متغیرهای d, k را به عنوان قطر D ، k تعریف کنیم. حال

$$A = D + kG$$

درایمان A به ازای d یا k است. حال هدف این است که $\lambda_{\min}(A)$ را کمینه کنیم. برای این کار از GP استفاده می‌کنیم.

یک پوزیتمینال از k و d است: A_{ij}

طبق جزوه و کتاب:

$$\min \lambda$$

$$\text{st } \sum_{j=1}^n (A_{ij} v_j) / (\lambda v_i) \leq 1 \quad \text{for } i=1, \dots, n$$

$$\lambda \left(\sum \frac{1}{d_i} \right) + \sum \frac{1}{k_i} \leq u$$

حالا برنامه ریزی GP برای حل سؤال است.

برای معادب شدن سؤال کافی است از (d, k) برای هر تابع استفاده کنیم و

از جایگذاری (n) و m برای هر متغیر استفاده کنیم.

6. - متغیرهای d, k را تقریبی کنیم. حال:

$$D = \text{diag}(d), K = \text{diag}(k) \quad G = \frac{1}{2} \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \Rightarrow A = D + K * G = \begin{pmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \end{pmatrix}$$

$$A = \begin{pmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} 0 & k_2 & k_3 \\ k_1 & 0 & k_3 \\ k_1 & k_2 & 0 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 2d_1 & k_2 & k_3 \\ k_1 & 2d_2 & k_3 \\ k_1 & k_2 & 2d_3 \end{pmatrix}$$

حال برای کسب کردن مسئله LP از A و λ می‌گیریم:



$$\min \quad \lambda$$

$$\text{st:} \quad \sum_{j=1}^n A_{ij} v_j / (\lambda v_i) \leq 1 \quad \text{for } i=1, \dots, n$$

$$\forall \left(\frac{1}{d_1} + \frac{1}{d_2} + \frac{1}{d_3} \right) + \left(\frac{1}{k_1} + \frac{1}{k_2} + \frac{1}{k_3} \right) \leq u$$

تمام قیدهای مسئله را تابع هدف پوزی نوشتار هستند پس با تغییر متغیرهای (d, k) می‌توان

و $y_i = \lambda(x_i)$ می‌توان مسئله را ساده کرد.

الف، به وضع کردن بالای برای تعداد دانشجویان وجود دارد. یعنی به در هر دانشجو به مربعی با طول $\frac{w}{2}$ باید خالی باشد:  حال مساحت کلاس تقسیم بر $4 \times \frac{w^2}{4}$ برابر با حداکثر دانشجویان ممکن است.  برای $n = \frac{wL}{2}$ تست می کنیم آیا امکان دارد که n دانشجو در کلاس باشد یا خیر اگر نشد، از n یکی کم می کنیم و باز هم این کار را تکرار می کنیم تا یک n فیزیکی بیابیم. یا اینکه binary سرچ می کنیم و از $\frac{n}{2}$ شروع می کنیم و اگر نشد، سرانجام $\frac{3n}{4}$ می دهیم و اگر نشد

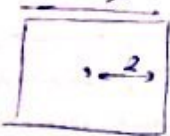
سرانجام $\frac{n}{4}$ را می آید

۱. برای چک کردن feasibility n :

به تعداد $\binom{n}{2}$ شرط داریم: $|x_i - x_j| + |y_i - y_j| \geq 2$
 ۱. شرطها معدوم نیستند. برای اینکه معدوم شوند، $4 \times \binom{n}{2}$ مسئلهای معدوم حل می کنیم که در هر کدام فرض می کنیم $x_i > x_j$ و $y_i > y_j$ و بعد از برداشتن قدر مطلق حاصل مسئلهای خطی می رسمیم. حال بعد از حل مسئله بررسی می کنیم که فرضهای ثانویه برقرار هستند یا خیر. پس $\frac{n}{2}$ کران بالا برای تعداد مسائل:

$$m \leq \log_2 n \times 4 \times \binom{n}{2}$$

\downarrow $n = \frac{wL}{2}$
 \downarrow $\frac{n}{2}$ کران هر مسئله
 \downarrow باینری سرچ

7- ب) باز هم کران بالایی برای تعداد دانشجویان هست. در هر سطح با طول l فقط یک دانشجو حضور دارد. پس l  ، $l=4$. پس حداکثر $n = \frac{w \cdot l}{16}$ قابل قبول است.

بخش بابیتری سرچشگه داریم مثل • بخش قبل است. اما برای چید کردن غیر یکنواختی n داریم:

به تعداد $\binom{n}{2}$ شرط داریم که: $2 \geq |x_i - y_i|$ یا $2 \geq |x_i - x_j|$ که محذب نیست. باز هم $\binom{n}{2}$ حالت در نظر می گیریم. در هر حالت یکی از 4 شرط: $2 \geq (x_i - x_j)$ ، $2 \leq x_j - x_i$ ، $2 \leq z_i - z_j$ ، $2 \leq z_j - z_i$ را می توانیم در شرط ها می آوریم. اگر یکی از این مسائل غیر یکنواخت باشد ، n غیر یکنواخت است.

پس ~~حداکثر~~ کران بالایی تعداد مسائل محذب $\binom{n}{2}$ $n \leq 4 \times 2^n$ است.

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("difference is near 1 so these reactions were very important")

```

change after knocking out Transport, nuclear: 0.9999999999997521

difference 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("difference is near 0 so these reactions were not important")

```

change after knocking out Fatty acid oxidation: -1.8715845223013645e-11

difference 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<=1,300*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 ]]
```