به نام خدا

امتحان پایانترم بهینه‌سازی محدب

کیارش بنی‌هاشم ۹۶۱۰۹۹۶۳ بهار ۱۳۹۹

---

۱. دقت کنید که یک مجموعه نامتناهی است اگر و تنها اگر ماکسیمم قدر مطلق درایه‌هایش نامتناهی باشد. این عملا بدیهی است زیرا اگر تعریف نامتناهی را نامحدود بودن نرم $l2$ بگیریم، چون در فضای nبعدی، نرم ۲ l بزرگتر مساوی ماکسیمم درایه است (طبق فیثاغورث یا حتی نامساوی مثلث) و همچنین از $\sqrt{n}$ برابر نرم l_∞ کمتر است (طبق فیثاغورث)، می‌توان این فرض را کرد.

در نتیجه نامتناهی بودن، معادل با این است که حداقل در یکی از $2n$ جهت مختصات (همه مثبت و هم منفی در نظر گرفته شده‌اند) تصویر مجموعه نامتناهی باشد. معادلا می‌توان برای هر $2n$ حالت از بردار $v = (-1)^k e_i$، سعی کنیم که مقدار را بیشینه کنیم. یعنی می‌توان این $2n$ برنامه را حل کرد و دید که آیا حداقل یکیشان نامتناهی است یا نه

$$maximize v^T x$$

$$s.t. \quad Ax \leq b \wedge Cx = d$$

اما چون گفته شده ۱ برنامه، همه‌ی این‌ها را با هم قاطی می‌کنیم. یعنی به شکل زیر (در زیر، V همان مجموعه‌ی $2n$ برداری است که از در نظر گرفتن خود بردارهای مختصات و منفی‌شان به دست می‌آید.)

$$maximize \sum_{v \in V} v^T x_v$$

$$Ax_v \leq b \wedge Cx_v = d$$

دقت کنید که برای هر کدام از $2n$ حالت مختلف، یک متغیر nتایی جدید گرفته‌ایم. ادعا می‌کنیم مثبت بی‌نهایت بودن جواب این برنامه معادل با نامتناهی بودن polyhedron است. اگر polyhedron نامتناهی باشد، یکی از مختصات‌ها به مثبت بی‌نهایت میل می‌کند. در نتیجه x_v متناظر با همه‌ی دیگر را یک نقطه‌ی ثابت می‌گیریم و در نتیجه در تابع هدف، این جملات ثابت می‌مانند. x_v مربوط به این مختصات اما چون می‌تواند به بی‌نهایت میل کند، lp هم می‌تواند به مثبت بی‌نهایت میل کند.

با توجه به این که این راه حل رو میشه تو cvx زد طوری که کار کنه، سراغ این که دوگانش رو دربیاریم و یه جوری ربط بدیم به این که این نامتناهی اگر و تنها اگر اون feasible باشه و اینا نمی‌رم.

در صورت نیاز، می‌توان برای نوشتن به فرم مناسب، ماتریس A' را در نظر گرفت که ماتریسی است که تعداد سطرها و ستون‌هایش $2n$ برابر A است و روی قطر آن A به تعداد $2n$ بار تکرار شده است. همچنین می‌توان همین کار را برای ماتریس C' انجام داد. بردارهای d', b' هم صرفا d, b هستند که $2n$ بار تکرار شده‌اند. بردار k هم برداری است از ابعاد $2n$ برابر x که در آن e_iها و $-e_i$ها پشت سر هم چیده شده‌اند. در این صورت برنامه به فرم زیر است.

$$maximize \quad k^T x'$$

$$s.t. A'x' \leq b', \quad C'x' = d'$$

ابعاد lp اما خب به وضوح خیلی خوب نیست و اگر n تا lp جدا می‌زدیم فکر کنم بهتر میشد چون عملا من هم همین کار رو کردم و در این حالت صرفا امید به cvx و ایناست که خودشون ببینن که A جدید اسپارسه.

ایده‌هایی واسه سریع‌تر کردن: اول یه توضیح بدم این که این راه حل، ممکنه اشتباه باشه ولی خب فکر کنم درسته (الان ساعت ۱۰ و ۴۰ه که دارم اینو می‌نویسم) و در هر حال هم برای حلش، یه preprocessطور نیازه که توش بیایم چک کنیم که آیا یه بردار پیدا میشه که هم تو نال‌اسپیس C باشه و هم نال‌اسپیس A. این یه جورایی حالت خاص مساله است ولی خب نتونستم با lp حلش کنم. اگه با lp بشه حلش کرد میشه lpها رو ترکیب کرد احتمالا. اما خب اگه نشه هم الگوریتم‌های دیگه‌ای که هستن وجود دارن و در هر کاربرد practical اصولا نباید مهم باشه (با فرم‌های دیگه‌ی convex هم ایده‌ای ذهنم نرسید. در واقع مشکل اصلی اینه که شرط $x \neq 0$ رو نمی‌دونم باید چجوری چپوند تو برنامه‌ریزی خطی). با این توضیحات، باز هم راه حل رو آوردم به این امید که اگه راه حل اصلی نمره‌ی کامل رو نگرفت، این راه حل بخشی از نمره رو جبران کنه.

راه حل:

این رو وقت نکردم خیلی دقیق کنم و در هر حال یه مشکل ریز داره اما خب. فکر کنم بشه گفت که اگه نامتناهی باشه، یکی از ۲ تا حالت می‌تونه پیش بیاد. یکی این که تو جهت یکی از $Ax - b$ها و فکر کنم بشه گفت که تو این حالت، باید یه x'ای باشه که $Ax' \leq 0, Ax' \neq 0, Cx = 0$. این شرط به نظرم کافیه (در کنار feasibillity طبیعتا). این که بگیم این شرط لازمه رو فکر کنم بشه با duality درآورد ولی خب مجددا وقت نکردم دقیق کنم. چک کردن این هم ساده است، کافیه به جای شرط دوم بذاریم

$$1^T Ax' = -1$$

به خاطر این که طبق شرط اول، اگر این شرط برقرار نباشه هم با اسکیل کردن می‌تونیم درستش کنیم.

یه حالت دیگه‌ای هم که ممکنه رخ بده اینه که تو یه جهت عمود بر سطرهای A, C نامتناهی بشه. چک کردن این رو با lp هر کاری کردم نتونستم. یعنی یه شرط از جنس $y \neq 0$ داره که نتونستم کاریش کنم. ولی خب میشه A, C رو گذاشت زیر هم، بعد بیایم الگوریتم‌هایی که واسه پیدا کردن nullspace وجود دارند رو بزنیم. دقت کنید که این همون حالت خاص مساله واسه وقتیه که A نداریم کلا! ولی خب این حالت رو هم با lp نتونستم کاریش کنم و

الگوریتم جدا می‌خواد. **میشه ولی بهش به شکل preprocess قبل lp نگاه کرد.** اما خب اگه بشه این حالت رو با lp حد کرد، ترکیبش با مساله کاری نداره.

یک نکته‌ای هم که کلا هست، اینایی که گفتم چند تا lp میشن (مثلا دو سه تا) ولی خب میشه مثل همون کاری که اول مساله گفتم اوم همشون رو تبدیل کرد به یه مساله. یعنی یه سری متغیر مستقل می‌گیریم و اینا. تنها چیزی که می‌مونه، اینه که اون حرفم رو دقیق کنم. برای اثباتش هم دقت کنید که اگر یه چیزی باشه که $Ax = 0, Cx = 0$، که خب نامتناهیه مجموعه جواب (کلا یه lp جدا واسه چک کردن این که ناتهی باشه مجموعه جواب می‌خوام. دقت کنید که وقتی می‌گم یه چیزی باشه، طبیعتا منظورم یه چیز غیر۰ و واسه همین هم نتونستم با lp حلش کنم.) اگر همچین چیزی نباشه، یعنی این که کل جهت‌های فضا رو بردارهای A, C پوشش می‌دند. در نتیجه برای این که نامتناهی بشه، باید ضرب داخلیش با حداقل یکی از این‌ها به مثبت یا منفی بی‌نهایت میل کنه. ضربداخلیش با سطرای C که خب همیشه ۰اه. پس باید با یکی از سطرای A میل کنه به منفی بی‌نهایت (مثبتش نمیشه چون تو polyhedron نیست). اگه یه x'ای باشه که ضربداخلیش با سطرای A ناصفر باشه که خب همونطور که گفتم اوکیه (با C هم باید ۰ باشه البته طبیعتا). پس برای این که مساله تموم شه باید صرفا بگم که این شرط لازم هم هست. برای این کار فرض کنید که همچین برداری پیدا نشه. یه بردار ثابت توی polyhedron وردارید. بعد یک بردار خیلی خیلی دور هم بیاید وردارید. اختلافشون رو حساب کنید. ضرب داخلی این بردار با سطرای A رو اسمش رو بذارید s. الان s تو یکی از درایه‌هاش یه عدد منفی خیلی خیلی گنده است. تو حداقل یکی از درایه‌هاش هم ولی مثبته چون فرض کردیم که بردار خوبی که ضرب داخلیش با همه‌ی سطرا بشه کمتر مساوی ۰ و ... نداریم. الان بیاید s رو نرمالایز کنید ولی. اون بخشش که مثبته، خیلی کوچیک می‌شه. به طور دقیق‌تر، چون هر چه اون نقطه‌ی دور رو دورتر بگیرم، می‌تونم نرم رو بیشتر کنم. از طرفی دقت کنید که قبل از نرمالایز کردن، کران بالا هست روی این که چقدر هر کدوم از درایه‌های s می‌تونه مثبت باشه به خاطر این که الان اون نقطه‌ی ثابتی که گرفتیم (نزدیکه)، بالاخره یه ضرب داخلی‌ای داره با هر کدوم از سطرا و از اختلاف b_i با اون ضرب داخلی اگه بیشتر بخوایم افزایش بدیم، خب دیگه تو polyhedron نمی‌مونه. پس شما من می‌تونم روی گوی واحد، یه بردار پیدا کنم که $Cx' = 0$ و $Ax' \leq t$ که t رو می‌تونم به دلخواه کوچک کنم (البته طبیعتا مثبته). در نتیجه می‌تونم یه x' پیدا کنم که $Ax' \leq 0$ و $Cx' \leq 0$. دقت کنید که چون یکی از درایه‌های Ax' همینجوریش هم یه عدد منفی اکید بود، مشکلی سر اون شرط دوم هم پیش نمیاد.

پس این راه هم جواب میده و سریعتره اما خب یه preprocess می‌خواد اما در هر حال اصولا هر راه حلی که واسه حالت خاصی که A نداریم رو بدین، اصولا میشه به‌جای preprocess از اون استفاده کرد و در نتیجه اگه اون حالت خاص رو بشه با lp حل کرد، این حالت کلی رو هم میشه.

۲. با استفاده از محاسبه‌ی هسین مساله را حل می‌کنیم. همچنین برای سادگی به‌جای نمادگزاری مساله، از $x^\alpha y^\beta$ استفاده می‌کنیم.

$$\nabla f = \begin{bmatrix} \alpha x^{\alpha-1} y^\beta \\ \beta x^\alpha y^{\beta-1} \end{bmatrix} \implies H = x^{\alpha-2} y^{\beta-2} \begin{bmatrix} \alpha(\alpha-1)y^2 & \alpha\beta xy \\ \alpha\beta xy & \beta(\beta-1)x^2 \end{bmatrix}$$

حال دقت کنید که ضریب پشت ماتریس همواره مثبت است. پس برای بررسی H باید دترمینان آن و نیز ضرایب روی قطر را بررسی کنیم.

(آ) H مثبت‌نیمه‌معین است اگر و تنها اگر

$$\alpha(\alpha-1) \geq 0 \wedge \beta(\beta-1) \geq 0 \wedge x^2 y^2 \alpha\beta(1-\alpha-\beta) \geq 0$$

در نتیجه اگر $\alpha, \beta > 0$ باشند که ممکن نیست چون که باید هر دو حداقل ۱ باشند و در نتیجه نامساوی سوم درست نخواهد بود. اگر هر دو منفی باشند (منظور نامثبت)، مشکلی نداریم چون که نامساوی‌های برقرارند. پس اگر هر دو نامثبت باشند تابع محدب است. اگر یکی مثبت و دیگری نامثبت باشد، مثلا α مثبت باشد، در این صورت

$$\alpha(\alpha-1) \geq 0 \implies \alpha \geq 1, \quad \beta(1-\alpha-\beta) \geq 0 \implies \beta \geq 1-\alpha$$

از طرفی دقت کنید که اگر $\alpha \geq 1, \beta \geq 1-\alpha$، هر ۳ نامساوی برقرارند. پس یک حالت هم این است که $\alpha\beta \leq 0, \alpha+\beta \geq 1$. شکل مورد نظر در ۱ آمده است.

(ب) برای دو شرط اول باید داشته باشیم

$$\alpha(\alpha-1) \leq 0 \wedge \beta(\beta-1) \leq 0 \implies 1 \geq \alpha, \beta \geq 0$$

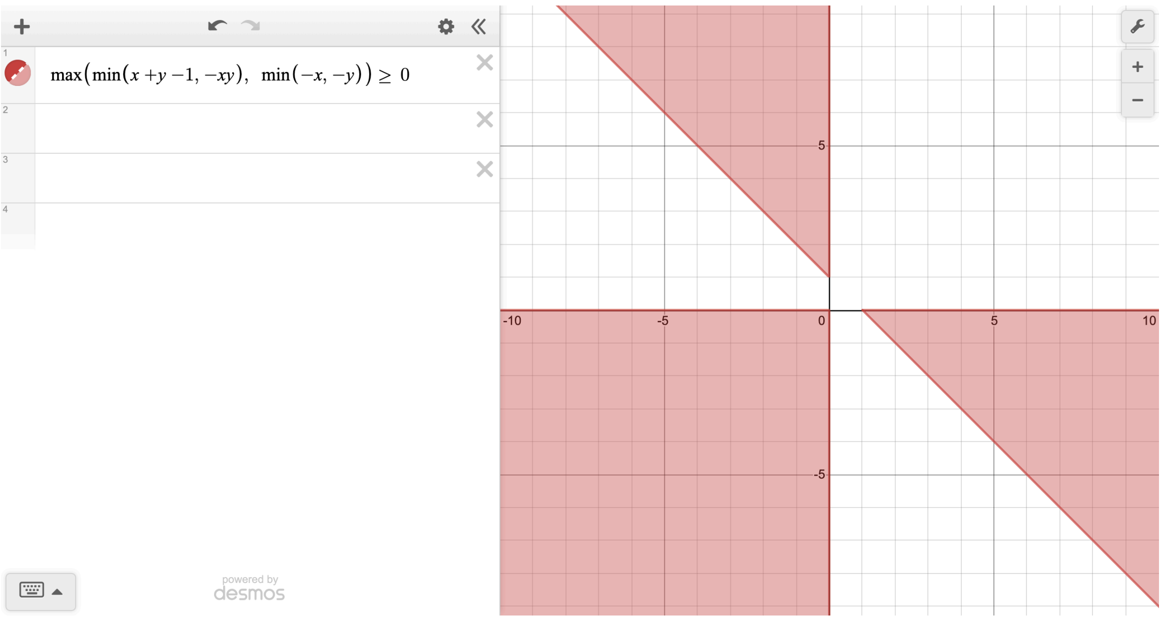
همچنین برای شرط سوم باید داشته باشیم

$$1-\alpha-\beta \geq 0 \implies \alpha+\beta \leq 1$$

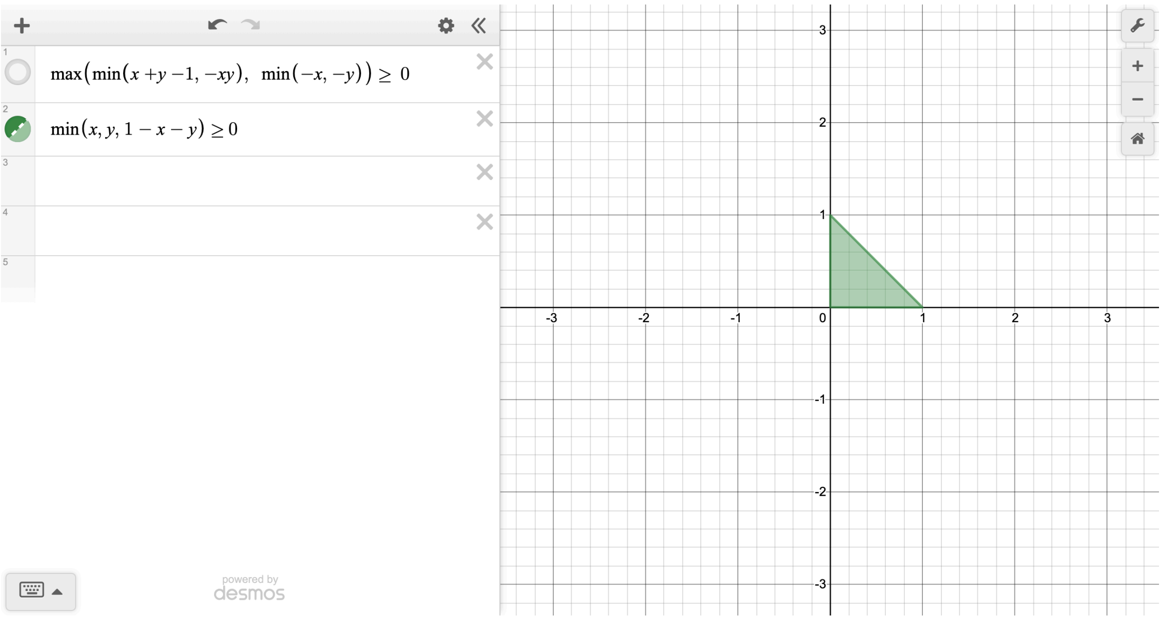
پس مقعر بودن معادل است با

$$\alpha, \beta \geq 0 \wedge \alpha+\beta \leq 1$$

شکل مورد نظر در ۲ آمده است.



شکل ۱: شکل سوال ۲ الف



شکل ۲: شکل سوال ۲ ب

حال به حل بخش امتیازی می‌پردازیم. به این منظور، دقت کنید که هسین روی قطر برابر است با

$$(\prod x_i^{\alpha_i})\frac{\alpha_i(\alpha_i-1)}{x_i^2}$$

و خارج از قطر برابر است با

$$(\prod x_i^{\alpha_i})\frac{\alpha_i\alpha_j}{x_ix_j}$$

که چون $(\prod x_i^{\alpha_i})$ مثبت است، می‌توان آن را در نظر نگرفت. با این توصیفات، به حل مساله می‌پردازیم. **ابتدا مقعر بودن و سپس محدب بودن را بررسی می‌کنیم**

(آ) اولا دقت کنید که اگر $\alpha_i < 0$، درایه‌ی iام روی قطر مثبت خواهد بود. در نتیجه قطعا مقعر نیست. پس $\alpha \geq 0$. حال برای هر v دلخواه باید داشته باشیم.

$$\forall v \in R^n : \sum v_i^2\frac{\alpha_i^2-\alpha_i}{x_i^2} + 2\sum_{i\neq j} v_iv_j\frac{\alpha_i\alpha_j}{x_ix_j} \leq 0 \iff (\sum v_i\frac{\alpha_i}{x_i})^2 \leq \sum \alpha_i\frac{v_i^2}{x_i^2} \quad (1)$$

اگر که داشته باشیم $\sum \alpha_i \leq 1$،

$$RHS \geq RHS \times \sum \alpha_i = (\sum \alpha_i)(\sum \alpha_i\frac{v_i^2}{x_i^2})$$

که طبق کوشی شوارتز از LHS بیشتر است و در نتیجه تابع مقعر است. از طرفی اگر ۱ برای v دلخواه برقرار باشد، داریم

$$v_i = x_i \implies (\sum \alpha_i)^2 \leq (\sum \alpha_i) \implies \sum \alpha_i \leq 1$$

پس تابع مقعر است اگر و تنها اگر

$$\alpha \geq 0 \wedge \sum \alpha_i \leq 1$$

(ب) فرض کنید که دو تا از α_iها مثبت باشند. در این صورت با در نظر گرفتن تصویر روی همان دو متغیر، باید هنوز تابع محدب بماند. اما با توجه به این که در بخش غیرامتیازی ثابت کردیم که اگر محدب باشد همچین چیزی ممکن نیست، این حالت منتفی است. پس یا همه نامثبت‌اند و یا دقیقا یکی مثبت است. اگر همه نامثبت باشند، باید برای هر v دلخواه داشته باشیم (بسط دادن هسین مشابه قبل است)

$$\forall v \in R^n : \sum v_i^2\frac{\alpha_i^2-\alpha_i}{x_i^2} + 2\sum_{i\neq j} v_iv_j\frac{\alpha_i\alpha_j}{x_ix_j} \geq 0 \iff (\sum v_i\frac{\alpha_i}{x_i})^2 \geq \sum \alpha_i\frac{v_i^2}{x_i^2} \quad (2)$$

که همواره برقرار است زیرا سمت راست نامثبت است و سمت چپ نامنفی. اگر هم دقیقا یکی از α_iها مثبت باشد و بقیه نامثبت، با قرار دادن $v_i = x_i$ نتیجه می‌گیریم که

$$(\sum \alpha_i)^۲ \geq \sum \alpha_i$$

و در نتیجه یا $\sum \alpha_i \leq ۰$، یا $\sum \alpha_i \geq ۱$. در حالت اول، اگر همه‌ی x_i هایی که ضریبشان منفی است را یکی کنیم (یعنی یک متغیر کنیم). این کار ممکن است چون precomposition با آفین)، تابع باید محدب بماند. اما خب محدب نمی‌ماند زیرا الان یک α_i مثبت داریم، یک α_i نامثبت و جمعشان هم نامثبت است که در شرایط بخش غیرامتیازی صدق نمی‌کند چون در حالت یکی مثبت، یکی نامثبت، باید جمع حداقل ۱ می‌شد.
پس در حالت دوم هستیم و در نتیجه

$$\sum \alpha_i \geq ۱$$

ثابت می‌کنیم که همین شرایط کافیست. به این منظور ابتدا حالتی که $\sum \alpha_i = ۱$ است را بررسی می‌کنیم. در این حالت، اگر فرض کنیم که $\alpha_n > ۰$ و بقیه نامثبت‌اند، می‌دانیم $x \in R^{n-۱} \to \prod x_i^{\alpha_i}$ محدب است. در نتیجه طبق قانون پرسپکتیو، تابع زیر محدب است

$$x \in R^{n-۱}, g(t,x) = t \prod((\frac{x_i}{t})^{\alpha_i}) = t^{۱-\sum \alpha_i} \prod x_i^{\alpha_i}$$

که همان تابع ماست با این تفاوت که به‌جای x_n از t استفاده کرده‌ایم. پس این حالت حل شد. در حالتی که $\sum \alpha_i > ۱$ هم می‌توانیم تعریف کنیم $g(x) = \prod x_i^{\frac{\alpha_i}{\sum \alpha_i}}$. طبق چیزی که الان ثابت کردیم، g محدب است. پس چون تابع زیر محدب است (چون $\sum \alpha_i \geq ۱$) و همچنین صعودی است

$$x \in R^+, h(x) = x^{\sum \alpha_i}$$

نتیجه می‌گیریم که تابع اصلی هم محدب است. پس برای این بخش هم ۲ حالت وجود دارد. حالت اول این که

$$\alpha \leq ۰$$

و حالت دوم هم این که دقیقا یکی از α_iها مثبت است و همچنین

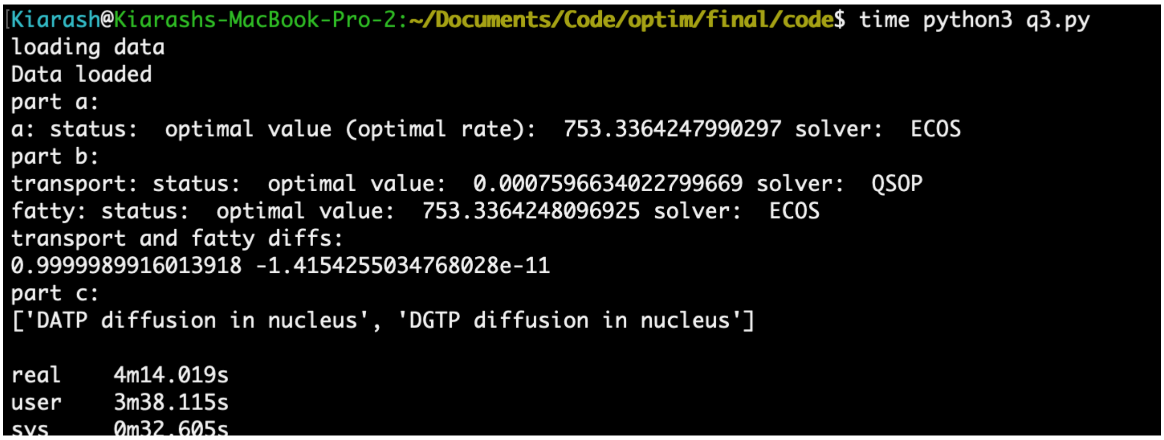
$$\sum \alpha_i \geq ۱$$

۳. کد مورد نظر در زیر آمده‌است. مطابق کد، برای حل مساله از تابع solve_with_removed_index استفاده می‌شود که با حذف تعدادی از واکنش‌ها، مساله‌ی بهینه‌سازی را تشکیل داده و حل می‌کند. خروجی برنامه در شکل ۳ قابل مشاهده است (اعداد در این شکل آورده شده‌اند).
در توضیح بخش ب: مشخص است که یکی از با حذف یک دسته‌از از آزمایش‌ها، نرخ فرق چندانی نمی‌کند اما با حذف یک دسته‌ی دیگری، فرق زیادی می‌کند. علتش این است که همانطور که توضیح داده شده است، هر آزمایش به تعدادی مواد نیاز دارد که دیگری آزمایش‌ها آن را برایش فراهم می‌کنند چون $Sv = ۰$. در نتیجه از نتیجه‌ی حاصل می‌توان دید که یکی از آزمایش‌ها تاثیر زیادی (چه مستقیم و چه غیرمستقیم) در فراهم کردن مواد لازم برای آزمایش biomass داشته و به شدت ضروری بوده است در حالی که آزمایش دیگر این‌گونه نبوده است. به طور دقیق‌تر transport بسیار ضروری بوده است چون با حذف آن نرخ بهینه به شدت کاهش پیدا کرده است.

```
import json
import numpy as np
import cvxpy as cp
print('loading data')
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]];
knockout_names = [
        'Transport, nuclear',
        'Fatty acid oxidation',
        ]
print('Data loaded')

def solve_with_removed_index(removed=None):
    solvers = {'ECOS': cp.ECOS, 'QSOP': cp.OSQP, 'SCS': cp.SCS}
    for solver_name, solver in solvers.items():
        u = np.array(upper_bound.copy())
        l = np.array(lower_bound.copy())
        if removed is not None:
            removed = np.array(removed)
            u[removed] = 0
            l[removed] = 0
        m, n = S.shape
        v = cp.Variable(n)
        constraints = [
```

```
                S @ v == 0,
                v <= u,
                v >= l,
                ]
        obj = cp.Maximize(v[-1])
        problem = cp.Problem(obj, constraints)
        ans = problem.solve(solver=solver)
        if problem.status == 'optimal':
            return problem, v.value, solver_name

def get_knockout_index(knockout):
    return [i for i, sub_name in enumerate(subsystem) if sub_name in knockout]

#returns indexes of all reactions in subsystem 'name'
def get_subystem_indexes(name):
    return get_knockout_index({name})

def part_a():
    #should run read_data before this
    p_a, v_a, solver_a = solve_with_removed_index()
    return p_a, v_a, solver_a

def evaluated_diff(v_w, v_tilde):
    return (v_w[-1] - v_tilde[-1]) / v_w[-1]

def part_b():
    p_transport, v_transport, solver_transport = solve_with_removed_index(
        get_subystem_indexes(knockout_names[0]))
    p_fatty, v_fatty, solver_fatty = solve_with_removed_index(get_subystem_indexes(
        knockout_names[1]))
    return p_transport, v_transport, solver_transport, p_fatty, v_fatty,
        solver_fatty

def part_c(v_a, verbose=True):
    possible_indexes = get_subystem_indexes(knockout_names[0])
    if verbose:
        print('length of list: ', len(possible_indexes))
    li = []
    for i, index in enumerate(possible_indexes):
        if verbose:
            print(f'starting {i}th index {index}, name={name[index]}')
        p_index, v_index, solver_index = solve_with_removed_index([index])
        evaluation = evaluated_diff(v_a, v_index)
        made_it = evaluation > 0.2
        if verbose:
            print(f'status: {p_index.status}, solver: {solver_index}, evaluation: {
                evaluation}, made it: {made_it}')
        if made_it:
            li.append(name[index])
```

```python
    return li

def main():
    p_a, v_a, solver_a = part_a()
    print('part a: ')
    print('a: status: ', p_a.status, 'value (optimal rate): ', p_a.value, 'solver: '
        , solver_a)
    print('part b: ')
    p_transport, v_transport, solver_transport, p_fatty, v_fatty, solver_fatty =
        part_b()
    print('transport: status: ', p_transport.status, 'value: ', p_transport.value, '
        solver: ', solver_transport)
    print('fatty: status: ', p_fatty.status, 'value: ', p_fatty.value, 'solver: ',
        solver_fatty)
    print('transport and fatty diffs: ')
    print(evaluated_diff(v_a, v_transport), evaluated_diff(v_a, v_fatty))
    li_c = part_c(v_a, verbose=False)
    print('part c: ')
    print(li_c)

main()
```



```
[Kiarash@Kiarashs-MacBook-Pro-2:~/Documents/Code/optim/final/code$ time python3 q3.py
loading data
Data loaded
part a:
a: status:  optimal value (optimal rate):  753.3364247990297 solver:  ECOS
part b:
transport: status:  optimal value:  0.0007596634022799669 solver:  QSOP
fatty: status:  optimal value:  753.3364248096925 solver:  ECOS
transport and fatty diffs:
0.9999989916013918 -1.4154255034768028e-11
part c:
['DATP diffusion in nucleus', 'DGTP diffusion in nucleus']

real    4m14.019s
user    3m38.115s
sys     0m32.605s
```

شکل ۳: شکل سوال ۳

۴. (آ) صورت حکم چیزی به فرم فلان اگر و فقط اگر بهمان است. من ثابت می‌کنم که بهمان اگر و تنها اگر فلان. دقت کنید که همان مطلب ثابت شده است و صرفا چون تشکیل دوگان از این سمت برایم راحت‌تر بود این کار را می‌کنم.
مساله‌ی زیر را در نظر بگیرید

$$minmize_{\mu} \quad -c^T\mu$$
$$s.t. \quad S^T\mu \leq 0$$

این مساله به وضوح محدب است (اصلا آفین است). همچنین شرط slater برقرار است زیرا اگر قرار دهیم $\mu = 0$، در نامساوی‌ها (که همه خطی‌اند) صدق می‌کند. در نتیجه strong dualtiy برایش برقرار است. از طرفی دقت کنید که جواب این مساله بیشتر مساوی ۰ است، اگر و تنها اگر بخش بهمان سوال برقرار باشد یعنی برای هر $S^T\mu \leq 0$ بتوان گفت که $c^T\mu \leq 0$. همچنین همواره کمترمساوی ۰ است زیرا می‌توان قرار داد $\mu = 0$. پس جواب این مساله ۰ است اگر و تنها اگر بخش بهمان برقرار باشد. از طرفی با تشکیل دوگان داریم

$$\mathcal{L} = -c^T\mu + v^TS^T\mu = (-c + Sv)^T\mu$$

که با اینفیمم‌گیری روی μ، منفی بی‌نهایت است اگر ضریب ناصفر باشد و در غیر این صورت ۰ است. پس مساله‌ی دوگان به فرم زیر است

$$maximize \quad 0$$
$$s.t. \quad Sv - c = 0 \wedge v \geq 0$$

از طرفی جواب این مساله برابر با ۰ است، اگر و تنها اگر feasible باشد که معادل است با این که بخش فلان مساله درست باشد یعنی c یک بردار thermodinamically feasible باشد. پس حکم ثابت شد چون ثابت کردیم که فلان معادل است با ۰ بودن جواب مساله‌ی دوگان که معادل است با ۰ بودن مساله‌ی اصلی که معادل است با بهمان.

(ب) فرض کنید که حکم غلط باشد. مساله‌ی زیر را در نظر بگیرید

$$minimize \quad -t$$
$$s.t. \quad S^Tm = 0 \wedge m \geq t$$

در این صورت جواب این مساله، قطعا مقداری منفی‌ای نخواهد بود زیرا اگر منفی باشد، t مثبت است و در نتیجه $m > 0$ و همچنین $m^TS = 0$ که حکم است و در نتیجه با فرض خلف در تناقض است. پس جواب مساله بزرگتر مساوی ۰ است. از طرفی اگر قرار دهیم $m = 0 \wedge t = 0$، به وضوح به ۰ می‌رسیم. پس جواب این مساله ۰ است.
مساله به وضوح محدب است (مجددا همه‌چی آفین است) و شرط slater برقرار است چون می‌توانیم قرار دهیم $t = -1, m = 0$ (البته اگر $t = 0$ قرار می‌دادیم هم اوکی بود). در نتیجه strong dualtiy داریم. مساله‌ی دوگان را تشکیل می دهیم.

$$\mathcal{L} = -t + u^T(S^Tm) + \lambda^T(t1 - m) = t(\lambda^T1 - 1) + (Su - \lambda)^Tm$$

که با اینفیمم‌گیری روی t, m، برابر با منفی بی‌نهایت است اگر یکی از ضرایب ناصفر باشد و در غیر این صورت ۰ است. پس مساله‌ی دوگان به فرم زیر است.

$$maximize \quad 0$$
$$s.t. \quad \lambda \geq 0 \wedge Su = \lambda \wedge \sum 1^T\lambda = 1$$

چون جواب مساله‌ی اصلی ۰ بود، جواب این مساله هم ۰ است و در نتیجه feasible است. پس چون داریم

$$Su = \lambda \geq 0$$

طبق فرض

$$Su = 0 \implies \lambda = 0 \implies 1 = 1^T\lambda = 0$$

که به وضوح تناقض است.

۵. (آ) با توجه به قیدها، مشخص است که $l, w > ۰$. با این حساب، تعریف کنید

$$l' = ln(l), w' = ln(w)$$

قیدهای روی $l, w, \frac{l}{w}$ به فرم زیر درمی‌آیند.

$$۰ \leq l' - w' \leq ln(۲), \quad ln(۱۰) \leq w' \leq ln(۲۰), \quad ln(۲۰) \leq l' \leq ln(۳۰)$$

همچنین قید روی مساحت به شکل زیر است

$$l' + w' \geq ln(۳۰۰)$$

برای محاسبه‌ی هزینه هم داریم

$$cost = ۲lw + ۲l + ۲\pi\frac{w}{۲} = ۲e^{l'+w'} + ۲e^{l'} + \pi e^{w'}$$

که به وضوح محدب است.

(ب) کد مورد نظر در زیر آمده است (همچنین در صورت نیاز در فایل $q۵۲$)

```python
import numpy as np
import cvxpy as cp
lp = cp.Variable()
wp = cp.Variable()
constraints = [
        0 <= lp - wp,
        lp - wp <= np.log(2),
        np.log(20) <= lp,
        lp <= np.log(30),
        np.log(10) <= wp,
        wp <= np.log(20),
        lp + wp >= np.log(300),
        ]
cost = 2 * cp.exp(lp + wp) + 2 * cp.exp(lp) + np.pi * cp.exp(wp)
obj = cp.Minimize(cost)
problem = cp.Problem(obj, constraints)
problem.solve(solver=cp.SCS)
print('status: ', problem.status)
print('total cost: ', problem.value)
l = np.exp(lp.value)
w = np.exp(wp.value)
print('l: ', l)
print('w: ', w)
s = l * w
print('filter size: ', s)
```

با اجرای کد می‌توان دید که سایز فیلتر ۳۰۰ است.

در صورتی که استدلال بدون اجرای کد نیاز باشد: اگر شرط روی مساحت را در نظر نگیریم، بهترین l, w همان کمترین l, w هستند یعنی مساحت ۲۰۰ که خب مطلوب نیست. اگر شرط مساحت را اضافه کنیم، قطعا باید شرط tight باشد زیرا اگر یک شرط tight نباشد، ضریب دوگانش ۰ است و در نتیجه با حذف آن، طبق شرایط KKT می‌توان دید که این شرط نیاز نبوده است اما خب گفتیم که این شرط نیاز است چون بدون آن جواب فرق می‌کند (دقیق‌تر از این دلیلی نمی‌بینم بنویسم چون در کلاس کلا این مطلب توضیح داده شده بود).

۶. (آ) اگر D و K را با بردار جایگزین کنیم (با توجه به این که قطری هستند)، داریم $A_{i,:} = d_i + k_i G_{i,:}$. همچنین

$$\gamma tr(D^{-۱}) + tr(K^{-۱}) = \sum \frac{\gamma}{d_i} + \sum \frac{۱}{k_i}$$

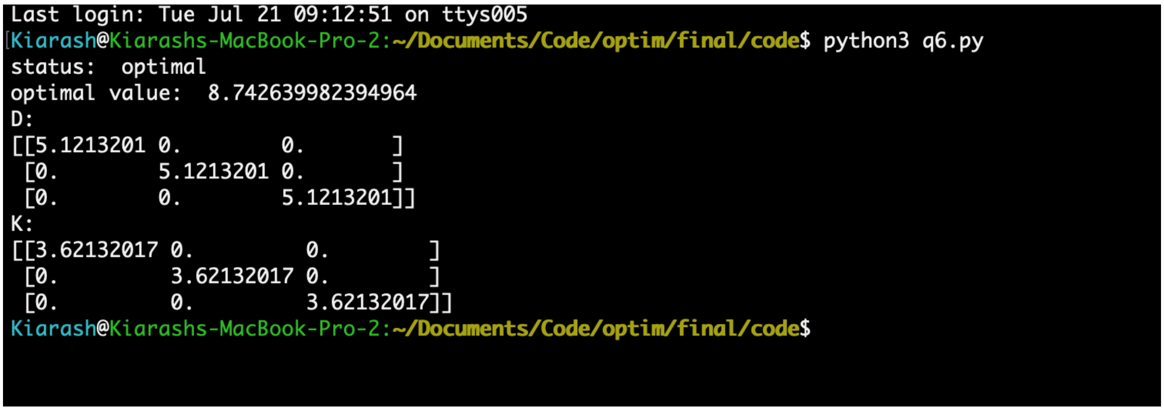
با این توصیفات مساله به فرم زیر در می‌آید.

$$\begin{gathered} minimize \quad \lambda_{max}(A) \\ s.t. \quad d \geq ۰ \wedge k \geq ۰ \\ A_{i,:} = d_i + k_i G_{i,:} \\ \sum \frac{\gamma}{d_i} + \sum \frac{۱}{k_i} \leq u \end{gathered}$$

که با توجه به محدب بودن بزرگترین مقدار ویژه و نیز $\frac{1}{x}$، مساله به فرم مناسب درآمد.

(ب) کد مورد نظر در زیر (و مجددا در صورتی که نیاز باشد در فایل $q۶$ آمده است) خروجی هم در شکل ۴ قابل مشاهده است.

```
import numpy as np
import cvxpy as cp

n = 3
u = 2
gamma = 2
G = (np.ones((n, n)) - np.identity(n)) / 2
d = cp.Variable(n)
k = cp.Variable(n)
A = cp.Variable((n, n))
constraints = [
        d >= 0,
        k >= 0,
        A == cp.diag(d) + cp.diag(k) @ G,
        gamma * cp.sum(cp.inv_pos(d)) + cp.sum(cp.inv_pos(k)) <= u,
        ]
obj = cp.Minimize(cp.norm(A))
problem = cp.Problem(obj, constraints)
problem.solve()
print('status: ', problem.status)
print('optimal value: ', problem.value)
print("D: ")
print(np.diag(d.value))
print("K: ")
print(np.diag(k.value))
```



```
Last login: Tue Jul 21 09:12:51 on ttys005
Kiarash@Kiarashs-MacBook-Pro-2:~/Documents/Code/optim/final/code$ python3 q6.py
status:  optimal
optimal value:  8.742639982394964
D:
[[5.1213201 0.        0.       ]
 [0.        5.1213201 0.       ]
 [0.        0.        5.1213201]]
K:
[[3.62132017 0.         0.        ]
 [0.         3.62132017 0.        ]
 [0.         0.         3.62132017]]
Kiarash@Kiarashs-MacBook-Pro-2:~/Documents/Code/optim/final/code$
```

شکل ۴: شکل سوال ۶

در صورتی که استدلال فوق به علت فرض تقارن نمره‌ی خاصی نمی‌گیرد: نتونستم مساله رو حل کنم اما خب تنها چیزی که ذهنم رسید رو می‌نویسم. استاد گفتند که A ماتریس دلخواه نیست. به این منظور کافیه دقت کنیم که اگه معادله‌ی مقدارویژه و بردار ویژه رو در نظر بگیریم، میشه به این شکل بهش نگاه کرد (میشه فرم بلوکی هم نوشت اینو که خب فرقی نمی‌کنه صرفا همین معادله‌هست)

$$Gv = y, \quad Dv + Ky = \lambda v$$

۷. فرض کنید که می‌خواهیم بررسی کنیم که آیا n نفر جا می‌شوند یا نه. باید مساله‌ی زیر را بررسی کنیم که feasible است یا نه.

$$۰ \leq x_i \leq l, \quad ۰ \leq y_i \leq w$$

$$||x_i - x_j, y_i - y_j||_p \geq ۲ \quad i < j$$

حال دقت کنید که

$$||a, b||_۱ \geq k \iff a + b \geq k \vee a - b \geq k \vee -a + b \geq k \vee -a - b \geq k$$

و به طور مشابه

$$||a, b||_\infty \geq k \iff a \geq k \vee b \geq k \vee -a \geq k \vee -b \geq k$$

نکته‌ی مهم این است که اگر علامت a و b را بدانیم، می‌توانیم قید بالا را برای $l۱$ به فرم آفین بنویسیم زیرا برای این که بررسی کنیم که آیا حداقل یکی از عبارات بزرگتر یا مساوی k است، کافیست آن عبارتی که در آن علامت‌ها مثبتند را بررسی کنیم چون بیشترین است. به طور دقیق‌تر اگر بدانیم که $a \geq ۰, b \geq ۰$، می‌دانیم که $||a, b||_۱ = a + b$. برای l_∞ هم، اگر هم بدانیم که علامت‌ها چه هستند و هم بدانیم که کدام‌یک از $|a|, |b|$ بزرگتر است، می‌توان قید را به فرم آفین نوشت. با این توصیفات، برای حل مساله‌ی feasiblity برای n نفر در نرم $l۱$، فرض می‌کنیم که بر اساس x مرتب‌اند به طوری که $x_i \leq x_{i+۱}$. برای بررسی ترتیب yها هم، همه‌ی $n!$ حالت ممکن را در نظر می‌گیریم. یعنی روی همه‌ی این حالات فور می‌زنیم. در هر کدام از این حالات، چون علامت را می‌دانیم، می‌توانیم برنامه‌ی محدب را تشکیل دهیم و بررسی کنیم که آیا n نفر جا می‌شوند یا خیر. حال اگر جواب نهایی T باشد، می‌توانیم با شروع از ۱ و امتحان همه‌ی اعداد تا زمانی که به T برسیم فور بزنیم (به طور دقیق‌تر از ۱ شروع می‌کنیم و برنامه را حل می‌کنیم، اگر feasible بود یعنی کمترمساوی T هستیم. اولین باری feasible نبود یعنی این که به $T + ۱$ رسیده‌ایم.)[۱] پس در کل

$$\sum_{i=۱}^{T+۱} i!$$

تا باید برنامه حل کنیم. عبارت فوق از نظر مرتبه عملا $T!$ است زیرا

$$\sum_{i=۱}^{T+۱} i! = \sum_{i=۱}^{T} i! + (T+۱)! \leq T \times (T)! + (T+۱)! \leq ۲(T+۱)!$$

برای نرم l_∞ هم دقیقا همین کار را می‌کنیم با این تفاوت که این که $|x_i - x_j|$ بزرگتر است یا $|y_i - y_j|$ را هم حدس می‌زنیم. در نتیجه برای چک کردن مساله برای n نفر، $۲^{\binom{n}{۲}} n!$ تا باید مساله حل کنیم. در نتیجه کرن بالا برای l_∞ برابر است با

$$\sum_{n=۱}^{T+۱} = ۲^{\binom{n}{۲}} n! \leq ۲^{\binom{T}{۲}+۱} T!$$

در صورت نیاز، می‌توان کران‌هایی بر حسب l, w هم داد. به طور دقیق‌تر، در حالتی که $p = ۱$، اگر حول هر نفر و به مرکزیت آن یک دایره‌ی l_∞ به شعاع ۱ بزنیم (در واقع همان مربع خودمان هستند)، این دوایر همدیگر را قطع نمی‌کنند زیرا اگر قطع کنند یعنی دو نقطه نزدیک‌تر از ۲ فاصله دارند. همه‌ی دوایر در مستطیل به طول و عرض $l + ۲, w + ۲$ جا می‌شوند و همچنین مساحت هرکدام حداقل ۴ است. در نتیجه

$$p = \infty \implies ۴T \leq (l+۲)(w+۲) \implies T \leq \frac{(l+۲)(w+۲)}{۴}$$

برای نرم $l۱$ هم به طور مشابه همه‌چیز در همان مستطیل جا می‌شوند اما مربع‌ها مساحتشان نصف شده پس کران بالا به شکل

$$T \leq \frac{(l+۲)(w+۲)}{۲}$$

در می‌آید.

[۱] چون حل مساله برای T به مراتب سخت‌تر از حل آن برای $T - ۱$ است، باینری سرچ زدن کار اشتباهی است.