دانشگاه علم و صنعت ایران
تئوری بهینه‌سازی (بهمن ۱۳۹۹)

گزارش پروژه پایانی درس بهینه‌سازی

علی اصغر تقی‌زاده
a_taghizadeh@comp.iust.ac.ir
شماره دانشجویی: ۹۸۹۲۲۰۰۴

سوال ۱

همانطور که در متن درس آمده است این مسئله باید به مسئله کسری خطی تبدیل شود. بدین منظور باید تابع هدف را معکوس کنیم و ترتیب min_max به max_min تبدیل شود.

$$\begin{aligned}\frac{1}{\gamma^\star} &= \min_{p_i\in[0,P_i],i=1,\ldots,K} \quad \max_{i=1,\ldots,K} \quad \frac{1}{\gamma_i} \\ &= \min_{p_i\in[0,P_i],i=1,\ldots,K} \quad \max_{i=1,\ldots,K} \quad \frac{\sum_{j=1,j\neq i}^{K} G_{ij}p_j+\sigma_i^2}{G_{ii}p_i},\end{aligned}$$

داده‌های مسئله را به این صورت تعریف می‌کنیم:

```
K = 4
G = 1/100*np.array([[37, 2, 1, 6], [10, 30, 3, 6], [1, 14, 354, 3], [10, 8, 6,
    171]], dtype=float)
sigma = np.ones((K,))
p = cp.Variable((K,), nonneg=True)
```

تعاریف داده‌ها متناظر با تعاریف صورت مسئله است. برای تبدیل این مسئله به فرمت DCP[1] تابع هدف را به صورت زیر تعریف می‌کنیم:

```
۱ numerators = []
۲ for i in range(K):
۳     numerator_i = [G[i, k]*p[k] for k in range(K) if i != k]
۴     numerators.append(cp.sum(cp.hstack(numerator_i)))
۵ numerators = sigma + cp.hstack(numerators)
۶ denominators = cp.multiply(cp.diag(G), p)
۷ isinr = numerators/denominators
```

در کد بالا ابتدا صورت کسرهای مربوط به تابع هدف محاسبه شده است. در خط ۴ استفاده از cp.hstack به این منظور است که لیست پایتونی تبدیل به لیستی شود که برای cvxpy قابل قبول باشد. در خط ۵ مقادیر σ با صورت کسرها جمع شده است. در خط ۶ مخرج کسرها محاسبه شده و در نهایت در خط ۷ صورت‌ها بر مخرج‌ها تقسیم شده‌اند.

[1] Disciplined Convex Programming

همچنین تنها محدودیت مسئله مقدار power است که در کد زیر اعمال شده است.

```
constraints = []
constraints.append(p<=30)
```

در نهایت با حل مسئله به صورت یک مسئله quasiconvex جواب مسئله به دست می‌آید.

```
objective = cp.Minimize(cp.max(isinr))
problem = cp.Problem(objective, constraints)
problem.solve(qcp=True)
print(p.value)
```

مقادیری که برای بردار p به دست می‌آید برابر است با:

$p = [16.11128432, 29.99998682, 4.38789886, 8.52961947]$

سوال ۲

در مورد صورت بندی مسئله برای مجموع محیط‌ها داریم:

$$\begin{aligned} & min \sum_{i=1}^{m} 2\pi r_i \\ s.t. \quad & c_i = c_i^{fix}, \quad r_i = r_i^{fix}, \qquad i = 1, 2, ..., n \\ & r_i + r_j \geq \sqrt{(c_i - c_j)^2}, \quad (i, j) \in S \end{aligned}$$

و همچنین برای مجموع مساحت‌ها داریم:

$$\begin{aligned} & min \sum_{i=1}^{m} \pi r_i^2 \\ s.t. \quad & c_i = c_i^{fix}, \quad r_i = r_i^{fix}, \qquad i = 1, 2, ..., n \\ & r_i + r_j \geq \sqrt{(c_i - c_j)^2}, \quad (i, j) \in S \end{aligned}$$

برای تعریف داده‌های مسئله داریم:

```
# data
m = 14 # circles count
n = 4 # permenant circles count
lim = 10 # used to define one axis of center of the permenant circles
C = np.array([[-lim, 0], [0, -lim], [0, lim], [lim, 0]]) # permenant circles center
R = [2]*n # permenant circles radius

I = np.array([[ 0, 13],[ 1, 7],[ 1, 12],[ 2, 11],[ 3, 12],[ 4, 5],[ 4, 9],[ 5, 8],[
    5, 9],[ 5, 11],[ 6, 10],[ 6, 12],[ 7, 13],[ 8, 13],[10, 11],[10, 12]])
```

که متناظر با تعاریف صورت مسئله است.همچنین برای محدودیت‌های مسئله داریم:

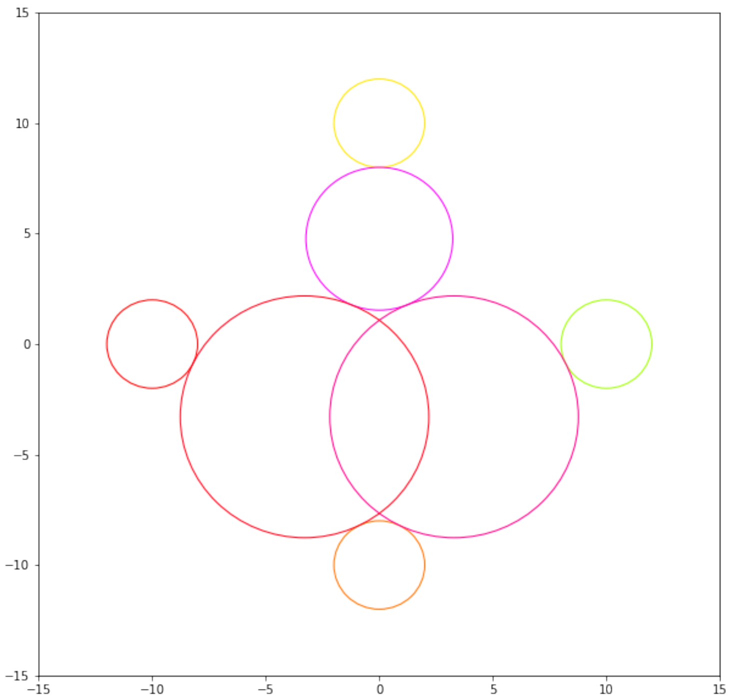
```
centers = cp.Variable((m,2))
radiuses = cp.Variable((m,), nonneg=True)

constraints = []
```

```
۵ for i in range(n):
۶     constraints.append(centers[i] == C[i])
۷     constraints.append(radiuses[i] == R[i])
۸
۹ for el in I:
۱۰     i = el[0]
۱۱     j = el[1]
۱۲     constraints.append(radiuses[i]+radiuses[j] >= cp.norm(centers[i]-centers[j]))
```

خط ۵ الی ۷ محدودیت‌های مربوط به دوایر ثابت را مشخص می‌کند و خط ۹ الی ۱۲ محدودیت‌های مربوط به همپوشانی دایره‌های تعیین شده.این محدودیت به این صورت تعریف شده است که مجموع طول شعاع‌ها بزرگتر از فاصله‌ی مراکز دو دایره‌ی مد نظر باشد.

پس از حل مسئله نمودار دایره‌ها برای دو مسئله در شکل۱ و۲ قابل مشاهده است.



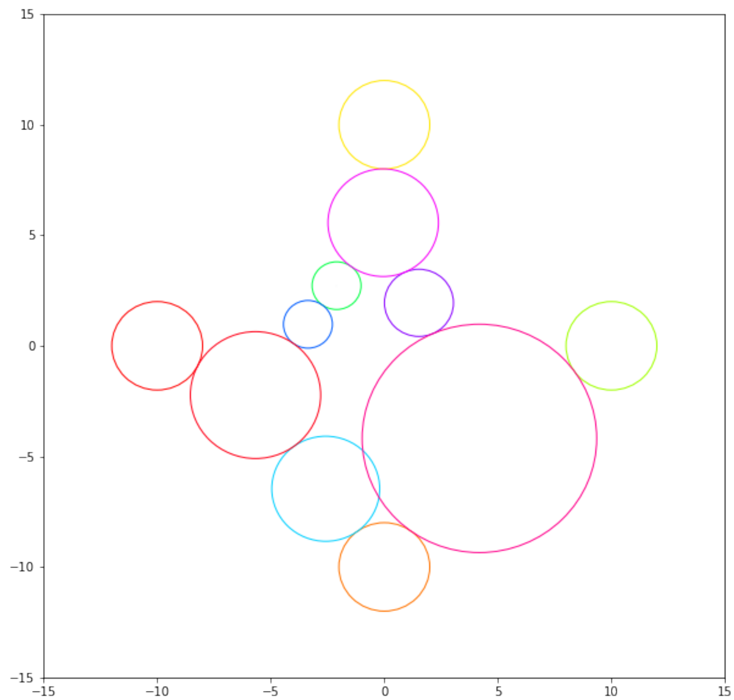
شکل ۱: مسئله با کمینه کردن محیط. برخی از دایره‌ها همپوشانی دارند و در شکل مشخص نیستند.

همچنین مجموع محیط به دست آمده برای بخش اول برابر با 139.3459 و مجموع مساحت برای مسئله دوم 210.7667 به دست آمده است.

سوال ۳

در صورتی که یکی از توابع هدف را بسط دهیم به رابطه‌ی زیر خواهیم رسید:

$$p_1^2 + p_n^2 + 5p_2^2 + 5p_{n-2}^2 + 6\sum_{i=3}^{n-3} p_i^2 - 4p_1p_2 - 4p_{n-1}p_n - 8\sum_{i=2}^{n-2} p_ip_{i+1} + \sum_{i=1}^{n-2} p_ip_{i+2}$$



شکل ۲: مسئله با کمینه کردن مساحت. برخی از دایره‌ها همپوشانی دارند و در شکل مشخص نیستند.

که می‌توان آن را به فرم ماتریسی $p^T Ap$ نوشت. با توجه به مقدار بسط داده شده ماتریس A برابر خواهد بود با:

$$A = \begin{bmatrix} 1 & -4 & 2 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 5 & -8 & 2 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 6 & -8 & 2 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & -8 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 6 & -8 & 2 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 5 & -4 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix}$$

البته این فرم تقارن ندارد و بعد از ساخت ماتریس متقارن مشخص می‌شود که این ماتریس مثبت معین است و در نتیجه مسئله را می‌توان به صورت برنامه‌نویسی مربعی نوشت. بنابراین مسئله به فرم زیر خواهد شد:

$$\begin{gathered} min \sum_{i=1}^{3} p^{(i)T} Ap^{(i)} \\ s.t. \; -1 \leq p_j^{(i)} \leq 1 \quad i = 1, 2, 3 \;\; j = 1, 2, ..., M \\ p_j^{(1)} = p_j^{(2)} = p_j^{(3)} \quad j = 1, 2, ..., F \\ p_G^{(1)} \geq -0.5 \\ p_G^{(2)} \leq -0.5 \end{gathered}$$

در قطعه کد زیر محدودیت‌های مسئله مشخص هستند:

```
A = np.zeros((M, M), dtype=float)
A[0,0] = 1
A[1,1] = 5
A[M-1, M-1] = 1
A[M-2, M-2] = 5
```

```
A[0,1] = -4
A[M-2, M-1] = -4

for i in range(2, M-2):
    A[i,i] = 6

for i in range(1, M-2):
    A[i,i+1] = -8

for i in range(0, M-2):
    A[i,i+2] = 2

A = (A + A.T)/2


v1 = cp.Variable(M, name='p1')
v2 = cp.Variable(M, name='p2')
v3 = cp.Variable(M, name='p3')

constraints = []
constraints.append(v1[0]==p1)
constraints.append(v1[1]==p2)
constraints.append(v1[M-2]==pN_1)
constraints.append(v1[M-1]==pN)

constraints.append(v2[0]==p1)
constraints.append(v2[1]==p2)
constraints.append(v2[M-2]==pN_1)
constraints.append(v2[M-1]==pN)

constraints.append(v3[0]==p1)
constraints.append(v3[1]==p2)
constraints.append(v3[M-2]==pN_1)
constraints.append(v3[M-1]==pN)

constraints.append(v1<=1)
constraints.append(v2<=1)
constraints.append(v3<=1)

constraints.append(-1<=v1)
constraints.append(-1<=v2)
constraints.append(-1<=v3)

for i in range(F):
    constraints.append(v1[i]==v2[i])
    constraints.append(v2[i]==v3[i])

constraints.append(v1[G] <= (5.0-
constraints.append(v2[G] >= (5.0
```

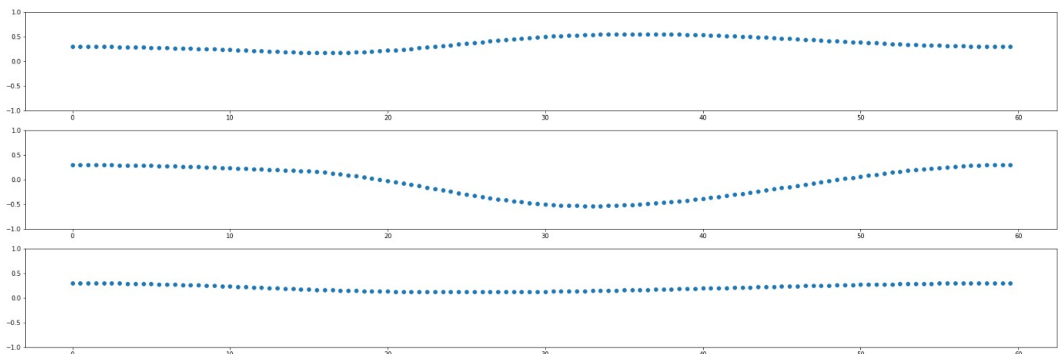
همچنین برای تابع هدف داریم:

```
objectives = [cp.quad_form(v1, A), cp.quad_form(v2, A), cp.quad_form(v3, A)]

objective = cp.Minimize(cp.sum(cp.hstack(objectives)))
problem = cp.Problem(objective, constraints)
problem.solve(solver=cp.ECOS)
```

در نهایت مسیری که به دست می‌آید در شکل ۳ قابل مشاهده است.

انهنای کوچکی که در مسیر p_3 به وجود آمده به علت این است که اعداد به دست آمده خیلی دقیق نیستند و معمولا خطا دارند.



شکل ۳: مسیرهای به دست آمده برای مسئله سوم. به ترتیب از بالا به پایین برای p_1و p_2 و p_3