1. SDP Relaxation and Heuristics for Two-Way Partitioning Problem

   (a) Q 5.39 textbook

$$\begin{aligned} min\ x^T W x \\ s.t.\ x_i^2 = 1, \forall i \in \{1, .., n\} \end{aligned}$$

i. Show that the two-way partitioning problem can be cast as

$$\begin{aligned} min\ tr(WX) \\ s.t.\ X \succeq 0, rank(X) = 1 \\ X_{ii} = 1, \forall i \in \{1, .., n\} \end{aligned}$$

$x^T W x = tr(x^T W x) = tr(W x x^T)$
$let\ X = x x^T$
$(\forall i) x_i^2 = 1 \iff x_i = \{-1, 1\} \implies x^T I x = n$
$x^T I x = tr(x x^T) = n$
$(\forall i, j) X_{ij} = \{-1, 1\}$
$((\exists i) X_{ii} = -1 \implies tr(X) < n)$
$thus, for\ tr(X) = n : (\forall i) X_{ii} = 1$

$X = x x^T = x \begin{bmatrix} a_1 & a_2 & .. & a_n \end{bmatrix} = \begin{bmatrix} a_1 x & a_2 x & .. & a_n x \end{bmatrix}, a_i \in \mathbb{R}, x \in \mathbb{R}^n$
$(\forall i)(\exists j) \beta_{ij} a_i x = a_j x \implies \beta_{ij} a_i x - a_j x = 0$
$let\ \gamma_{ij} = \beta_{ij} a_i - a_j$
$\gamma_{ij} x = 0$
$x \neq 0 \implies ((\forall i)(\exists j) \gamma_{ij} = 0 \implies linear\ dependence\ between\ all\ column\ vectors\ of\ X)$
$thus,\ rank(X) = 1$

$(\forall w) w^T X w = w^T x x^T w = (x^T w)^T x^T w$
$(\forall i, w)(x^T w)_i (x^T w)_i \geq 0 \implies (\forall w)(x^T w)^T (x^T w) \geq 0 \iff X\ is\ SPD$

Combining all constraints and objective forms the desired result.

ii. SDP relaxation of two-way partitioning problem. Using the formulation in part (a), we can form the relaxation:

$$\begin{aligned} min\ tr(WX) \\ s.t.\ X \succeq 0 \\ X_{ii} = 1, \forall i \in \{1, .., n\} \end{aligned}$$

This problem is an SDP, and therefore can be solved efficiently. Explain why its optimal value gives a lower bound on the optimal value of the two-way partitioning problem (5.113). What can you say if an optimal point X^* for this SDP has rank one?

$$\begin{aligned} &L(X, Z, v) = tr(WX) - tr(XZ) + tr(diag(v)diag(X) - I) \\ &L(X, Z, v) = -tr(diag(v)I) + tr(WX) - tr(XZ) + tr(diag(v)diag(X)) \\ &g(Z, V) = \begin{cases} -1^T v, & W - Z + diag(v) \succeq 0 \\ -\infty, & o/w \end{cases} \\ &dual\ problem: \\ &\max_{Z,v} -1^T v \\ &s.t.\ W - Z + diag(v) \succeq 0 \\ &Z \succeq 0 \end{aligned}$$

If an optimal point X^* for the relaxed problem has rank one:
X^* has minimal possible rank and $X^* \neq 0$, $X^* \succeq 0$.
$X^* \succeq 0$, so primal feasible.
Functions are all differentiable, KKT conditions apply at optimality where there exists a dual solution.
Dual of relaxed problem is feasible: $W + diag(v) \succeq Z, Z \succeq 0$
Using complementary slackness: $X^* \neq 0, -tr(X^* Z^*) = 0 \implies Z^* = 0$.

Dual problem of relaxed problem at optimality:

$$\begin{aligned} &\max_{v,Z} -1^T v = [\max_{v}\ -1^T v]_{Z=Z^*} \\ &s.t.\ W + diag(v) \succeq Z^*, Z^* = 0 \implies \\ &s.t.\ W + diag(v) \succeq 0 \end{aligned}$$

This solution is equivalent to the solution of the dual of the original problem. Thus, if $rank(X^*) = 1$ of the relaxed problem, X^* obtains the same solution as the original problem where $x^* x^{*T} = X^*$ as required.

iii. We now have two SDPs that give a lower bound on the optimal value of the two-way partitioning problem (5.113): the SDP relaxation (5.115) found in part (b), and the Lagrange dual of the two-way partitioning problem, given in (5.114). What is the relation between the two SDPs? What can you say about the lower bounds found by them? Hint: Relate the two SDPs via duality.

$$
\begin{aligned}
(5.115)\ & min\ tr(WX) \\
& s.t.\ X \succeq 0 \\
& X_{ii} = 1, \forall i \in \{1, .., n\}
\end{aligned}
$$

$$
\begin{aligned}
(5.114)\ & maximize\ -1^T v \\
& s.t.\ W + diag(v) \succeq 0
\end{aligned}
$$

Taken from previous section, dual of relaxed problem:

$$
\begin{aligned}
& L(X, Z, v) = tr(WX) - tr(XZ) + tr(diag(v)diag(X) - I) \\
& L(X, Z, v) = -tr(diag(v)I) + tr(WX) - tr(XZ) + tr(diag(v)diag(X)) \\
& g(Z, V) = \begin{cases} -1^T v, & W - Z + diag(v) \succeq 0 \\ -\infty, & o/w \end{cases}
\end{aligned}
$$

Dual problem:

$$
\begin{aligned}
& \max_{Z,v} -1^T v \\
& s.t.\ W - Z + diag(v) \succeq 0 \\
& Z \succeq 0
\end{aligned}
$$

It is evident that solution to the dual of relaxed problem has a tightened generalized inequality constraint due to Z compared to (5.114):

$$
W + diag(v) \succeq Z
$$

This leads to potentially bigger v and hence potentially larger objective value of dual maximization problem (smaller objective value to the primal minimization problem). Thus, solution of relaxed problem provides a lower bound to the the original problem.

(b) Q 11.23(b-d) textbook

	SDP bound (11.66)	Optimum	b (11.67)	c
small	-5.33445288482		-12.30891878	-5.33440509588
medium	42.2266162135		13.05483539	-39.220002609
large	-66.0855132055	x	-2135.80923688	-660.557123191

	d-a	d-b	d-c
small			
medium			
large			

- b) heuristic partitioning

```
import cvxpy as cp
import numpy as np
from scipy.io import loadmat
from os.path import dirname, join as pjoin
import numpy.linalg as linalg

def solve(W):
    print("problem size:", W.shape[0])

    #dual of original:
    print("dual of original:")
    dim = W.shape[0]
    v = cp.Variable((dim,1))
    constraints = [W + cp.diag(v) >> 0]
    prob = cp.Problem(cp.Maximize( -cp.sum(v) ),
                      constraints)
    prob.solve()
    print("prob.status:", prob.status)

    lower_bound = 0
    if prob.status not in ["infeasible", "unbounded"]:
        # Otherwise, problem.value is inf or -inf, respectively.
        print("Optimal value: %s" % prob.value)
        lower_bound = prob.value

    lower_bound = -lower_bound
    print("lower_bound:", lower_bound)

    #dual of relaxed:
    print("dual of relaxed:")
    X = cp.Variable((dim,dim))
    constraints = [X >> 0, cp.diag(X) == np.ones((dim,))]
    prob = cp.Problem(cp.Minimize( cp.trace(cp.matmul(W,X)) ),
                      constraints)
    prob.solve()
    print("prob.status:", prob.status)

    if prob.status not in ["infeasible", "unbounded"]:
        # Otherwise, problem.value is inf or -inf, respectively.
        print("Optimal value: %s" % prob.value)

    ret = prob.variables()[0].value
    eigenValues, eigenVectors = linalg.eig(ret)
```

```
    idx = eigenValues.argsort()[::-1]
    eigenValues = eigenValues[idx]
    eigenVectors = eigenVectors[:,idx]
    x_approx = np.sign(eigenVectors[0])[:,np.newaxis]
    p_heuristic = (x_approx.T).dot(W).dot(x_approx)
    print("heuristic objective: ", p_heuristic)
    print("heuristic objective - lower_bound: ", p_heuristic - lower_bound)
    print("-----")

m = loadmat('../data/hw4data.mat')
w5 = np.array(m['W5'])
w10 = np.array(m['W10'])
w50 = np.array(m['W50'])

solve(w5)
solve(w10)
solve(w50)
```

- c) randomized method

```
import cvxpy as cp
import numpy as np
from scipy.io import loadmat
from os.path import dirname, join as pjoin
import numpy.linalg as linalg
import math

def solve(W):
    print("problem size:", W.shape[0])

    #dual of original:
    print("dual of original:")
    dim = W.shape[0]
    v = cp.Variable((dim,1))
    constraints = [W + cp.diag(v) >> 0]
    prob = cp.Problem(cp.Maximize( -cp.sum(v) ),
                      constraints)
    prob.solve()

    print("prob.status:", prob.status)

    lower_bound = 0

    if prob.status not in ["infeasible", "unbounded"]:
        # Otherwise, problem.value is inf or -inf, respectively.
        print("Optimal value: %s" % prob.value)
        lower_bound = prob.value

    lower_bound = -lower_bound
    print("lower_bound: ", lower_bound)

    #dual of relaxed:
    print("dual of relaxed:")

    #restrict to PSD for randomized sampling
    #on proper covariance matrix later
    X = cp.Variable((dim,dim), PSD=True)

    constraints = [X >> 0, cp.diag(X) == np.ones((dim,))]
    prob = cp.Problem(cp.Minimize( cp.trace(cp.matmul(W,X)) ),
                      constraints)
    prob.solve(solver=cp.SCS, max_iters=4000, eps=1e-11, warm_start=True)
```

```
    print("prob.status:", prob.status)

    if prob.status not in ["infeasible", "unbounded"]:
        # Otherwise, problem.value is inf or -inf, respectively.
        print("Optimal value: %s" % prob.value)

    ret = prob.variables()[0].value
    eigenValues, eigenVectors = linalg.eig(ret)

    K = 100 #number of samples
    xs_approx = np.random.multivariate_normal(np.zeros((dim,)),
                                              ret, size=(K))
    xs_approx = np.sign(xs_approx)
    ps = xs_approx.dot(W).dot(xs_approx.T)
    p_best = np.amin(ps)
    print("best objective (randomized): ", p_best, "size: ", K)
    print("objective delta: best objective - lower_bound: ",
          p_best - lower_bound)
    print("-----")

m = loadmat('../data/hw4data.mat')
w5 = np.array(m['W5'])
w10 = np.array(m['W10'])
w50 = np.array(m['W50'])

solve(w5)
solve(w10)
solve(w50)
```

- d) greedy heuristic refinement

2. Interior Point Method