Worcester Polytechnic Institute

Applications of Semidefinite Programming

by

Caroline Johnston

in the
Mathematical Sciences Department
Worcester Polytechnic Institute

April 25, 2019

Abstract

In the field of operations research and combinatorial optimization, many real-world problems can be modeled using semidefinite programming. Semidefinite programming is a type of convex optimization in which we aim to optimize a linear function, in this case, the trace of the product of a matrix and the variable matrix X, while subject to nonlinear constraints. In linear programming, each decision variable x_i is subject to nonnegativity constraints, while in semidefinite programming, the decision matrix X is required to be positive semidefinite.

Semidefinite programming is best known for its contribution to the Max-Cut problem, which is NP-complete. In 1994, Goemans and Williamson's relaxation of this integer quadratic program into a semidefinite program was able to produce an approximate solution within 0.878 of the exact solution, the best known-polynomial time approximation for this problem to date. This breakthrough brought newfound attention to semidefinite programming and its ability to approximate hard, combinatorial optimization problems.

In this report, we examine an interior point method developed by Helmberg, Rendl, Vanderbei, and Wolkowicz. We develop the theory behind it, bringing together concepts from Linear Programming, Mathematical Optimization, Discrete Optimization, and Numerical Methods. We then discuss the algorithm's limitations in practice and expansions for its generalization.

We then explore the application of semidefinite programming to the Quadratic Assignment Problem (QAP), another NP-hard problem. The Quadratic Assignment Problem was first introduced in 1957 by Koopmans and Beckmann as a model for assigning economic activities to a set of locations. The problem is most typically described as assigning n facilities to n locations, minimizing the quadratic objective function that arises from the product of both the distance between these locations and the flow between these facilities. This model has many applications such as optimizing travel time between hospital buildings, the amount of wiring between computer components, or the placement of keys on a typewriter.

Finally, we discuss a relaxation of the QAP formulation into a semidefinite program by Zhao, Karisch, Rendl, and Wolkowicz. We solve these relaxations using the NEOS solver, a web service for numerical optimization problems hosted by the Wisconsin Institute for Discovery and the University of Wisconsin in Madison, which uses an optimized version of the interior point method described above. We provide a comprehensive explanation of the NEOS formulation of these problems and the interpretation of their solutions in order to solve these semidefinite programs and be able to use these results in practice.

Acknowledgements

I would like to thank Professor Martin for his guidance and for the people of the Stratton Hall Math Lounge for their support during this process.

Contents

Chapter 1

Introduction

Semidefinite programming is a very powerful way of modeling convex optimization problems. In semidefinite programming, we aim to optimize a linear function, the trace of the product of a matrix and the variable matrix X, while subject to nonlinear constraints. This is contrary to linear programming in which each decision variable x_i is subject to nonnegativity constraints, while in semidefinite programming, the decision matrix X is required to be positive semidefinite.

Semidefinite programming has been used to approximate solutions to many hard, combinatorial problems. Its most successful contribution in this domain is the solution to the semidefinite relaxation of the Max-Cut problem, which is NP-complete. In 1994, Goemans and Williamson's relaxation of this integer quadratic program into a semidefinite program was able to produce an approximate solution within 0.878 of the exact solution, the best known-polynomial time approximation for this problem to date.

In this report, we examine an interior point method developed by Helmberg, Rendl, Vanderbei, and Wolkowicz. This method draws inspiration from concepts from linear programming, such as the central path, to find the optimal solution to the semidefinite program. We first develop the theory behind this method and then discuss a MATLAB implementation of the algorithm.

We then explore the application of semidefinite programming to the Quadratic Assignment Problem (QAP), another NP-hard problem, typically described as the assignment of n facilities to n locations. The goal of the QAP is to minimize the quadratic objective function that arises from the product of both the distance between these locations and the flow between these facilities. This model has many applications such as optimizing travel time between hospital buildings, the amount of wiring between computer components, or the placement of keys on a typewriter.

Finally, we discuss a relaxation of the QAP formulation into a semidefinite program by Zhao, Karisch, Rendl, and Wolkowicz. We develop the theory behind the relaxation and then solve these relaxations using the NEOS solver, a web service for numerical optimization problems hosted by the Wisconsin Institute for Discovery and the University of Wisconsin in Madison, which uses an optimized version of the interior point method described above. We provide a comprehensive explanation of the NEOS formulation of these problems and the interpretation of their solutions in order to solve these semidefinite programs and be able to use these results in practice.

Chapter 2

Optimization Background

I present the following background information on various optimizations methods/topics that are required for the understanding of this report.

2.1 Properties of Matrices

Recall that a symmetric n x n matrix A is positive semidefinite if $z^T Az \geq 0 \; \forall \; z \in \mathbb{R}^n$. Equivalently, if all eigenvalues of A are nonnegative, the matrix A is positive semidefinite.

A square matrix can uniquely be written as the sum of a symmetric and skew-symmetric matrix. Suppose A is a symmetric matrix. Then $A = \frac{1}{2}(A + A^T) + \frac{1}{2}(A - A^T)$. Note that $\frac{1}{2}(A + A^T)$ is a symmetric matrix and $\frac{1}{2}(A - A^T)$ is a skew-symmetric matrix.

2.2 Semidefinite Programming

The general form of the semidefinite primal problem (SDP) is as follows:

Semidefinite Primal (SDP)

$$\begin{aligned} \text{maximize} \quad & \langle C, X\rangle \\ \text{subject to} \quad & \langle A_i, X\rangle \; = b_i \\ & X \; \succeq 0 \end{aligned}$$

where $\mathcal{M}_n$ is the vector space of symmetric $n \,\mathrm{x}\, n$ matrices,

$$A : \mathcal{M}_n \to \mathbb{R}^k,$$

$$C \in \mathcal{M}_n,\ b \in \mathbb{R}^k,$$

and $\langle W, V\rangle = tr(WV^T)$.

We now show that any linear program (LP) can be formulated as an SDP. Recall that a linear program is written in the form:

Linear Programming Primal (P)

$$\begin{aligned} \text{maximize} \quad & c^T x \\ \text{subject to} \quad & Ax \; = b \\ & x \; \geq 0 \end{aligned}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \text{ x } n}$ (i.e., the set of real matrices),

and $b \in \mathbb{R}^m$.

(2.1)

We can rewrite this LP as an SDP in the following way. Let

$$C = \begin{pmatrix} c_1 & 0 & \dots & 0 \\ 0 & c_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c_n \end{pmatrix}, \qquad X = \begin{pmatrix} x_1 & 0 & \dots & 0 \\ 0 & x_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & x_n \end{pmatrix},$$

$$A_i = \begin{pmatrix} A_{i1} & 0 & \dots & 0 \\ 0 & A_{i2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_{in} \end{pmatrix} \qquad i = 1, ..., m$$

$$\text{and } b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}.$$

Then,

$$CX = \begin{pmatrix} c_1x_1 & 0 & \dots & 0 \\ 0 & c_2x_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c_nx_n \end{pmatrix},$$

$$tr(CX) = c_1x_1 + c_2x_2 + ... + c_nx_n$$

and

$$\langle A_1, X \rangle = \left\langle \begin{pmatrix} A_{11} & 0 & \dots & 0 \\ 0 & A_{12} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_{1n} \end{pmatrix} \begin{pmatrix} x_1 & 0 & \dots & 0 \\ 0 & x_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & x_n \end{pmatrix} \right\rangle$$

$$= tr\left(\begin{pmatrix} A_{11}x_1 & 0 & \dots & 0 \\ 0 & A_{12}x_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_{1n}x_n \end{pmatrix}\right) = A_{11}x_1 + A_{12}x_2 + ... + A_{1n}x_n = b_1$$

Expanding this for $i = 1, ..., m$, the LP becomes the SDP exactly. For example, take the LP

$$\begin{array}{lrl} \text{maximize} & 3x_1 + & x_2 \\ \text{subject to} & x_1 + & x_2 = 5 \\ & 7x_1 + & x_2 = 10 \\ & x_1, & x_2 \geq 0 \end{array} \tag{2.2}$$

$$C = \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} x_1 & 0 \\ 0 & x_2 \end{pmatrix}$$

$$CX = \begin{pmatrix} c_1x_1 & 0 \\ 0 & c_2x_2 \end{pmatrix} = \begin{pmatrix} 3x_1 & 0 \\ 0 & x_2 \end{pmatrix}$$

$$tr(CX) = 3x_1 + x_2$$

$$A_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\ A_2 = \begin{pmatrix} 7 & 0 \\ 0 & 1 \end{pmatrix},\ b = \begin{pmatrix} 5 \\ 10 \end{pmatrix}$$

$$\langle A_1, X\rangle = \left\langle \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} x_1 & 0 \\ 0 & x_2 \end{pmatrix} \right\rangle = tr \begin{pmatrix} x_1 & 0 \\ 0 & x_2 \end{pmatrix} = x_1 + x_2$$

$$\langle A_2, X\rangle = \left\langle \begin{pmatrix} 7 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} x_1 & 0 \\ 0 & x_2 \end{pmatrix} \right\rangle = tr \begin{pmatrix} 7x_1 & 0 \\ 0 & x_2 \end{pmatrix} = 7x_1 + x_2$$

Finally, because of the non-negativity constraints on x in the LP, X must be positive semidefinite. Thus, the LP formulation is equivalent to the form:

$$\begin{array}{ll} \text{maximize} & \langle C, X\rangle \\ \text{subject to} & \langle A_i, X\rangle \ = b_i \\ & X \ \succeq 0 \end{array}$$

$$\text{where } \mathcal{M}_n \text{ is the vector space of symmetric } n \,\text{x}\, n \text{ matrices}$$
$$A : \mathcal{M}_n \to \mathbb{R}^k$$
$$C \in \mathcal{M}_n,\ b \in \mathbb{R}^k$$

Recall that the dual of a linear program takes the form:

Linear Programming Dual (D)

$$\begin{array}{ll} \text{minimize} & b^T y \\ \text{subject to} & A^T y \ \geq c \\ & y \ \geq 0 \end{array}$$

$$\text{where} \quad c \in \mathbb{R}^n, \quad A \in \mathbb{R}^{m \text{ x } n}$$
$$b \in \mathbb{R}^m$$

Thus, the dual of (2.2) is

$$\begin{array}{ll} \text{minimize} & 5y_1 + 10y_2 \\ \text{subject to} & y_1 + 7y_2 \geq 3 \\ & y_1 + y_2 \geq 1 \\ & y_1, \quad y_2 \in \mathbb{R} \end{array}$$

Recall that the duality gap is the difference between the primal and dual solutions, or $c^T x - b^T y$. Note that strong duality (where the duality gap is 0 at optimality, i.e. $c^T x^* = b^T y^*$) does not always hold for SDPs.

2.3 Interior Point/Barrier Methods

The following ideas are adapted from [1]. Interior point methods (also known as barrier methods) are a class of algorithms that are used to solve convex optimization problems. Interior point methods explore the interior of the feasible region as opposed to the simplex method which explores the corner point feasible solutions. The simplex method suffers from the potential of becoming "stuck" in these corners, causing the algorithm to slow down. However, interior point methods do not suffer from this problem and can be used to speed up the convergence to an optimal solution.

A barrier function is a continuous function whose value increases to negative infinity (in the case of a maximization problem) as the function approaches the boundary of the feasible region of a given optimization problem. These functions are used to replace inequality constraints in the formulation of the primal or dual with a term that penalizes in the objective function, keeping the algorithm from reaching the corner points of the feasible region, avoiding the problem of becoming stuck as described above.

The most popular of these barrier functions is the logarithmic function, which can be used to replace the non-negativity constraints on x. Let us consider the form of the primal given in (2.1). Introducing a logarithmic barrier function to the primal would convert the formulation to the following:

$$\begin{array}{ll} \text{maximize} & c^T x + \mu \sum_j \log x_j \\ \text{subject to} & Ax = b \end{array}$$

For example, the objective function for (2.2) would now be

$$\zeta(x, \mu) = 3x_1 + x_2 + \mu \log(x_1) + \mu \log(x_2).$$

Note that as μ becomes small, $\zeta(x, \mu)$ approaches the original objective function of the primal (see Figure (2.1) below). Recall that at each iteration of the simplex method, the algorithm considers a corner point solution in which the non-basic variables have a value of 0. Also recall that the logarithmic function approaches $-\infty$ as x approaches zero. Thus, the logarithmic term in the objective function keeps the current feasible solution away from the corner points of the feasible region.

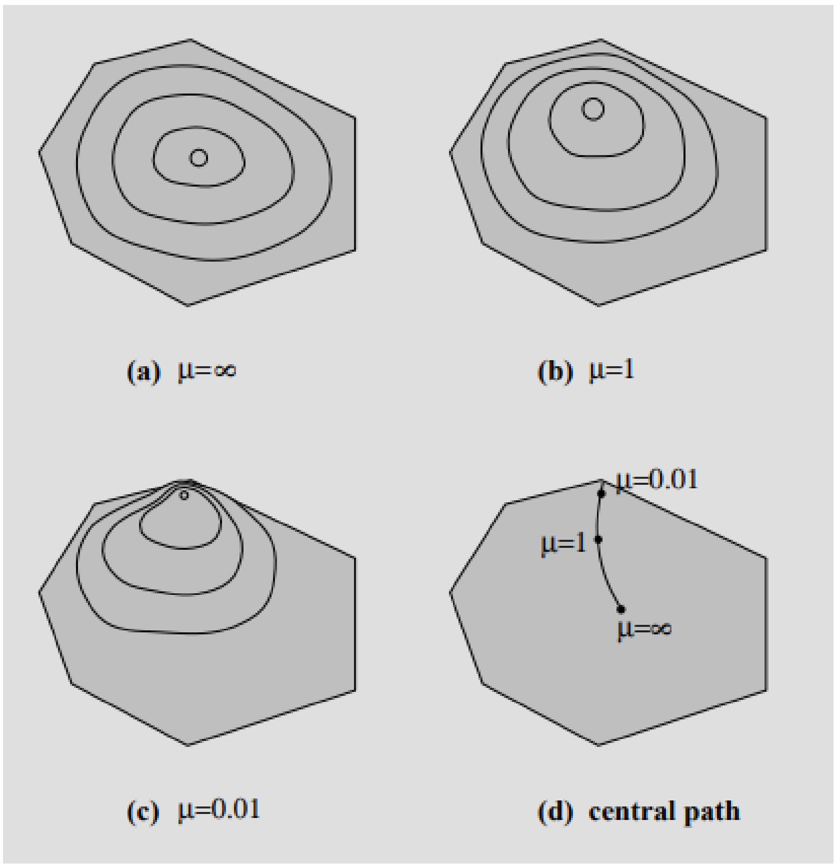

FIGURE 2.1: The level sets of the barrier function for various values of μ. Adapted from [1].

2.4 Lagrangian Multipliers

The following ideas are also adapted from [1]. Let us now consider an optimization problem of the form

$$\begin{aligned} &\text{maximize} \quad f(x) \\ &\text{subject to } g(x) = 0. \end{aligned}$$

Recall that ∇f, the gradient of f, is the direction in which f increases most rapidly. To find the critical points and maximum of f, we set the gradient of f to zero. However, because $g(x) = 0$, we now must consider a gradient in which it is orthogonal to the set of feasible solutions $\{x : g(x) = 0\}$. At each feasible point, $\nabla g(x)$ is a vector that is orthogonal to this feasible set at x. So, in order for x^* to be a critical point, it must be feasible and $\nabla f(x^*)$ must be proportional to $\nabla g(x^*)$. Thus,

$$\begin{aligned} g(x^*) &= 0, \\ \nabla f(x^*) &= \lambda \nabla g(x^*), \end{aligned}$$

where λ is called a “Lagrange multiplier”. The same idea can be extended to an optimization problem with multiple constraints where

$$\begin{aligned} g(x^*) &= 0, \\ \nabla f(x^*) &= \sum_{i=1}^{m} \lambda_i \nabla g(x^*). \end{aligned}$$

We can define the Lagrangian function, $L(x, \lambda) = f(x) - \sum_i \lambda_i g(x)$, which finds critical points for both x and λ. Because the optimization problem is now an unconstrained optimization problem, we can find the critical points by setting the Lagrangian function's first derivative equal to 0:

$$\begin{aligned} \frac{\partial L}{\partial x_j} &= \frac{\partial f}{\partial x_j} - \sum_i y_i \frac{\partial g_i}{\partial x_j} &= 0 \quad j = 1, 2, ..., n \\ \frac{\partial L}{\partial y_i} &= -g_i &= 0 \quad i = 1, 2, ..., m \end{aligned}$$

These equations are referred to as the “first order optimality conditions”.

2.5 Newton's Method

Newton's method is an iterative method used to find roots of a real-valued function. The method states that from the Taylor expansion of $f(x)$ around x, $f(x + \Delta x) \approx f(x) + f'(x)\Delta x$. If this equation is set equal to 0, then $f'(x)\Delta x = -f(x)$. Starting from an initial guess x_0 for the root to a function f, we can then solve for Δx. This brings us to a new point x_1 for which the process described above repeats until we find the desired root of the function.

2.6 Relaxations

Relaxations are a concept used in optimization to approximate solutions to difficult problems. This is achieved by creating a feasible region in the relaxed space that is larger than the feasible region of the original problem and that the objective function's relaxed value is greater than the objective function's original value (in the case of a maximization). Thus, the solution for the relaxed problem creates an upper bound on the original problem that is hopefully “tight” in some sense on the original problem.

The most famous example of a relaxation is most likely the relaxation of an integer linear program into a linear program. This relaxation is achieved by removing the constraint that the decision variables must be integer. Thus, for each $x_i \in \mathbb{Z}^+$, the variable is relaxed so that $x_i \in \mathbb{R}^+$, increasing the size of the feasible region. The problem then becomes a linear program. In this case, this transforms the NP-hard integer linear program into a linear program that can be solved in polynomial time by the simplex method. Similar relaxations can be applied to transform problems into an SDP.

Chapter 3

An Interior-Point Method for Semidefinite Programming

I present a comprehensive study of the algorithm and underlying theory presented in *An Interior-Point Method for Semidefinite Programming* from Helmberg, Rendl, Vanderbei, and Wolkowicz [2].

3.1 Preliminaries

The following are basic definitions and mathematical concepts that are fundamental in understanding the interior-point algorithm:

- $\mathcal{M}_n$ is the vector space of symmetric n x n matrices.
- Let U and $V \in \mathcal{M}_n$. The inner product of U and V is defined as:

  $$\langle U, V\rangle := tr(UV^T).$$

  Note that if $X \succeq 0$ and $Z \succeq 0$, then $\langle X, Z\rangle \succeq 0$ [3].
- Let $A, B \in \mathcal{M}_n$. The Löwner partial order is denoted as $A \preceq B$ (or $A \prec B$), which means that $A - B$ is positive semidefinite (or positive definite, respectively).
- The relation $v \leq w$ means the partial order induced by the cone of nonnegative vectors, i.e. $v \leq w$ if and only if $w - v$ belongs to this cone.
- For two matrices $U = (u_{ij})$ and $V = (v_{ij})$ of the same size, $U \circ V$ denotes the Hadamard (or element-wise) product, $(U \circ V)_{ij} = u_{ij} \cdot v_{ij}$. Note that $\langle U, V\rangle$ is the sum of all entries of $U \circ V : \langle U, V\rangle = e^T(U \circ V)e$, where e is the all ones vector.

- Let A be a linear operator from $\mathcal{M}_n \to \mathbb{R}^k$. Then there exists a unique operator A^T, called the adjoint of A, from $\mathbb{R}^k$ to $\mathcal{M}_n$ such that, for all X in $\mathcal{M}_n$ and all $y \in \mathbb{R}^k$, the following "adjoint relation" holds:

$$\langle A(X), y\rangle = \langle X, A^T(y)\rangle,$$

  where the inner product on the left is the ordinary dot product of vectors and the inner product on the right is the Frobenius inner product on matrices.

- For X in $\mathcal{M}_n$, $diag(X)$ denotes the vector in $\mathbb{R}^n$ that consists of the diagonal elements of X. For a vector x in $\mathbb{R}^n, Diag(x)$ denotes the diagonal matrix in $\mathcal{M}_n$ whose diagonal elements are obtained from x.

For example,

$$A = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}, \quad diag(A) = \begin{pmatrix} 1 \\ 4 \end{pmatrix}, \quad Diag(diag(A)) = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}.$$

Observe that $diag(Diag(x)) = x$ for any vector x, while $Diag(diag(A)) = A \circ I$ for any matrix A.

Note that both the Helmberg et al. paper and algorithm focus on the special case where $A(X) = diag(X)$, or $A^T(y) = Diag(y)$. For example, let $k = 2, A_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, $A_2 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$, and $X = \begin{pmatrix} 5 & 9 \\ 3 & 7 \end{pmatrix}$.Then,

$$A(X) = \begin{bmatrix} \langle A_1, X\rangle \\ \langle A_2, X\rangle \\ \vdots \\ \langle A_k, X\rangle \end{bmatrix} = \begin{bmatrix} \left\langle \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, X \right\rangle \\ \left\langle \begin{pmatrix} 0 & 0 \\ 0 & 1, \end{pmatrix}, X \right\rangle \end{bmatrix} = \begin{bmatrix} tr \begin{pmatrix} 5 & 9 \\ 0 & 0, \end{pmatrix} \\ tr \begin{pmatrix} 0 & 0 \\ 3 & 7, \end{pmatrix} \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \end{bmatrix} = diag(X)$$

$$y = \begin{pmatrix} 5 \\ 7 \end{pmatrix}$$

$$A^T(y) = \textstyle\sum_{i=1}^k y_i A_i = 5 * \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + 7 * \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 5 & 0 \\ 0 & 7 \end{pmatrix} = Diag(y)$$

3.2 The Primal

The authors begin by presenting a semidefinite linear program. It is named such because its objective is to optimize a linear function subject to linear inequality and equality constraints over a positive semidefinite matrix X.

Semidefinite Primal (SDP)

$$\begin{array}{ll} \text{maximize} & tr\,CX \\ \text{subject to} & A(X) \ = a \\ & B(X) \ \leq b \\ & X \ \succeq 0 \end{array}$$

where $\mathcal{M}_n$ is the vector space of symmetric $n \,\mathrm{x}\, n$ matrices

$$A : \mathcal{M}_n \to \mathbb{R}^k, \quad B : \mathcal{M}_n \to \mathbb{R}^m$$

$$C \in \mathcal{M}_n,\ a \in \mathbb{R}^k,\ b \in \mathbb{R}^m$$

$$A(X) = \begin{bmatrix} \langle A_1, X\rangle \\ \langle A_2, X\rangle \\ \vdots \\ \langle A_k, X\rangle \end{bmatrix}, \quad B(X) = \begin{bmatrix} \langle B_1, X\rangle \\ \langle B_2, X\rangle \\ \vdots \\ \langle B_m, X\rangle \end{bmatrix}$$

3.3 The Dual

Helmberg, et al. then derive the dual problem using Lagrangian methods through the following:

Let w^* denote the optimal objective value for the SDP. Let $y \in \mathbb{R}^k$ and $t \in \mathbb{R}^m_+$ be Lagrange multipliers for the equality and inequality constraints, respectively. We can see that

$$w^* = \max_{X \succeq 0} \min_{t \geq 0, y \in \mathbb{R}^k} tr(CX) + y^T(a - A(X)) + t^T(b - B(X)).$$

Proof:

Because X is a feasible solution to SDP, $A(X) = a$. Thus,

$$w^* = \max_{X \succeq 0} \min_{t \geq 0, y \in \mathbb{R}^k} tr(CX) + y^T(a - A(X)) + t^T(b - B(X))$$

$$w^* = \max_{X \succeq 0} \min_{t \geq 0, y \in \mathbb{R}^k} tr(CX) + t^T(b - B(X))$$

Because $t \geq 0$ and $B(X) \leq b$,

$$\min_{t\geq 0, y\in\mathbb{R}^k} t^T(b - B(X)) = 0.$$

Thus,

$$\begin{aligned} w^* &= \max_{X\succeq 0} \min_{t\geq 0, y\in\mathbb{R}^k} tr(CX) + t^T(b - B(X)) \\ w^* &= \max_{X\succeq 0} tr(CX) \end{aligned}$$

□

We can also see that

$$\begin{aligned} w^* &= \max_{X\succeq 0} \min_{t\geq 0, y\in\mathbb{R}^k} tr(CX) + y^T(a - A(X)) + t^T(b - B(X)) \\ &\leq \min_{t\geq 0, y\in\mathbb{R}^k} \max_{X\succeq 0} tr(C - A^T(y) - B^T(t))X + a^T y + b^T t \end{aligned} \tag{3.1}$$

Proof:

Given $X \in \mathcal{M}_n, t \in \mathbb{R}^m, y \in \mathbb{R}^k,$ with $X \succeq 0$ and $t \geq 0,$ define

$$\zeta(X,t,y) = \langle C, X\rangle - \sum_{i=1}^{k} y_i \langle A_i, X\rangle + y^T a + t^T b - \sum_{j=1}^{m} t_j \langle B_j, X\rangle$$

and let

$$w^* = \max_{X\succeq 0} \min_{t\geq 0, y\in\mathbb{R}^k} \zeta(X,t,y).$$

Claim:

$$w^* \leq \min_{t\geq 0, y\in\mathbb{R}^k} \max_{X\succeq 0} \zeta(X,t,y)$$

Now fix t and y and define $Z = \sum_{i=1}^{k} y_i A_i + \sum_{j=1}^{m} t_j B_j - C$.

$$\begin{aligned} \max_{X\succeq 0} \zeta(X,t,y) &= \max_{X\succeq 0} \langle C - \sum_{i=1}^{k} y_i A_i - \sum_{j=1}^{m} t_j B_j, X\rangle + y^T a + t^T b \\ &= \max_{X\succeq 0} \langle -Z, X\rangle + y^T a + t^T b \end{aligned} \tag{3.2}$$

Case 1: $Z \not\succeq 0$

Claim:

$$\max_{X\succeq 0} \zeta(X,t,y) \text{ is unbounded}$$

Because Z is not positive semidefinite, there exists $w \in \mathbb{R}^n$ such that $w^T Z w < 0$. Recall that the trace is invariant under cyclic permutations. Thus,

$$\begin{aligned}
w^T Z w &< 0 \\
tr(w^T Z w) &< 0 \\
tr(Z w w^T) &< 0 \\
\langle Z, w w^T \rangle &< 0 \\
\text{For } X = w w^T, \langle -Z, X \rangle &> 0 \\
\text{For } X = \alpha w w^T, \text{as } \alpha \to \infty, \langle -Z, X \rangle &\to \infty
\end{aligned}$$

Thus, the inner maximization is unbounded when Z has at least one negative eigenvalue.

Again, for given vectors t and y, consider now:

Case 2: $Z \succeq 0$
Recall that if $Z \succeq 0$ and $X \succeq 0$, then $\langle Z, X \rangle \succeq 0$. Thus,

$$v^* = \max_{X \succeq 0} \langle -Z, X \rangle = \max_{X \succeq 0} -\langle Z, X \rangle = -\max_{X \succeq 0} \langle Z, X \rangle = 0.$$

and

$$\max_{X \succeq 0} \langle -Z, X \rangle + y^T a + t^T b = y^T a + t^T b.$$

The claim (3.1) with (3.2) then becomes

$$\begin{aligned}
w^* &= \max_{X \succeq 0} \min_{t \geq 0, y \in \mathbb{R}^k} \langle C, X \rangle + \sum_{i=1}^{k} y_i (a_i - \langle A_i, X \rangle) + \sum_{j=1}^{m} t_j (b_j - \langle B_j, x \rangle) \\
&\leq \min_{t \geq 0, y \in \mathbb{R}^k} \max_{X \succeq 0} y^T a + t^T b - \langle Z, X \rangle \\
&= \min \left\{ y^T a + t^T b \,\middle|\, y \in \mathbb{R}^k, 0 \leq t \in \mathbb{R}^m, C \preceq \sum_{i=1}^{k} y_i A_i + \sum_{j=1}^{m} t_j B_j \right\}
\end{aligned}$$

Because X is feasible for SDP,

$$w* = \max \Big\{ \min_{t \geq 0, y \in \mathbb{R}^k} \langle C, X \rangle + \sum_{j=1}^{m} t_j (b_j - \langle B_j, X \rangle) \,\Big|\, \langle A_i, X \rangle = a_i, \langle B_j, X \rangle \leq b_j, X \succeq 0 \Big\}.$$

Note that

$$\begin{aligned}
\langle C, X \rangle + \sum_{j=1}^{m} t_j (b_j - \langle B_j, X \rangle) =& \langle C, X \rangle + \sum_{i=1}^{k} y_i (a_i - \langle A_i, X \rangle) + \sum_{j=1}^{m} t_j (b_j - \langle B, X \rangle) \\
=& \left\langle C - \sum_{i=1}^{k} y_i A_i - \sum_{j=1}^{m} t_j B_j, X \right\rangle + y^T a + t^T b
\end{aligned}$$

Thus,

$$w* = \max\Big\{\min_{t\geq 0, y\in\mathbb{R}^k} \left\langle C - \sum_{i=1}^{k} y_i A_i - \sum_{j=1}^{m} t_j B_j, X \right\rangle + y^T a + t^T b \,\big|\, \langle A_i, X\rangle = a_i, \langle B_j, X\rangle \leq b_j, X \succeq 0\Big\}$$

$$\leq \min \left\{ y^T a + t^T b \;\middle|\; y \in \mathbb{R}^k, 0 \leq t \in \mathbb{R}^m, C \preceq \sum_{i=1}^{k} y_i A_i + \sum_{j=1}^{m} t_j B_j \right\}$$

□

Notice that the original inner maximization of $tr(C - A^T(y) - B^T(t))X + a^T y + b^T t$ over X is bounded from above only if $Z \succeq 0$.

Thus, the weak dual of the problem, denoted as DSDP, is

Semidefinite Dual (DSDP)

$$\begin{array}{rc} \text{minimize} & a^T y + b^T t \\ \text{subject to} & A^T(y) + B^T(t) - C \succeq 0 \\ & y \in \mathbb{R}^k, t \in \mathbb{R}^m_+ \end{array}$$

The weak duality theorem for semidefinite programming states that given an X feasible for the SDP and (y, t) feasible for the DSDP, then $\langle C, X\rangle \leq y^T a + t^T b$.

Proof: Let X be feasible for SDP and let (y, t) be feasible for DSDP. Then, with $Z := A^T(y) + B^T t - C$,

$$\begin{aligned} \langle C, X\rangle &= \langle A^T(y) + B^T(t) - Z, X\rangle \\ &= \left\langle \sum_{i=1}^{k} y_i A_i + \sum_{j=1}^{m} t_j B_j - Z, X \right\rangle \\ &= \sum_{i=1}^{k} y_i \langle A_i, X\rangle + \sum_{j=1}^{m} t_j \langle B_j, X\rangle - \langle Z, X\rangle \end{aligned}$$

Because X is feasible for the SDP and (y, t) is feasible for the DSDP, $\langle A_i, X\rangle = a_i$ for each i and $\langle B_j, X\rangle \leq b_j$ for each j. Likewise, $X \succeq 0$ and $Z \succeq 0$. Thus,

$$\sum_{i=1}^{k} y_i \langle A_i, X\rangle + \sum_{j=1}^{m} t_j \langle B_j, X\rangle - \langle Z, X\rangle \leq \sum_{i=1}^{k} y_i a_i + \sum_{j=1}^{m} t_j b_j - \langle Z, X\rangle \leq y^T a + t^T b,$$

since $\langle Z, X\rangle \geq 0$. □

3.4 Setting Up the Algorithm

Helmberg, et al. introduce a primal-dual interior-point algorithm that solves the problems SDP and the DSDP simultaneously.

We study the non-degenerate case and omit the limit process that extends this to handle degeneracies. Assume that X strictly satisfies the inequalities of the primal problem, i.e. $b - B(X) > 0$ and $X \succ 0$. Also assume that the equality constraints on X are linearly independent; i.e., $\operatorname{rank}(A(\cdot)) = k$. When A and B are applied to nonsymmetric matrices, we will assume the skew-symmetric parts are mapped to zero, which implies that $A(M) = A(M^T)$ and $B(M) = B(M^T)$.

3.4.1 The Dual Barrier Problem

The authors present the barrier problem for DSDP, called the dual barrier problem.

Dual Barrier Problem

$$\begin{array}{rc} \text{minimize} & a^T y + b^T t - \mu(\log\det Z + e^T \log t) \\ \text{subject to} & A^T(y) + B^T(t) - C \;= Z \\ & t \geq 0, Z \succeq 0 \end{array}$$

where $\mu \in \mathbb{R}^+$ is called the barrier parameter, $\log(t)$ is computed entrywise, and e is the all ones vector.

3.4.2 The Lagrangian

For each $\mu > 0$, there exists a corresponding Lagrangian:

$$L_\mu(X, y, t, Z) = a^T y + b^T t - \mu(\log\det Z + e^T \log t) + \langle Z + C - A^T(y) - B^T(t), X\rangle$$

The first-order optimality constraints of the Lagrangian are as follows:

$$\nabla_X L_\mu = Z + C - A^T(y) - B^T(t) = 0 \tag{3.3}$$

$$\nabla_y L_\mu = a - A(X) = 0 \tag{3.4}$$

$$\nabla_t L_\mu = b - B(X) - \mu t^{-1} = 0 \tag{3.5}$$

$$\nabla_Z L_\mu = X - \mu Z^{-1} = 0 \tag{3.6}$$

Proof:

$$\begin{aligned}
\nabla_X L_\mu &= \frac{\partial \langle Z + C - A^T(y) - B^T(t), X\rangle}{\partial X} \\
&= \frac{\partial tr((Z + C - A^T(y) - B^T(t))X)}{\partial X} \\
&= Z + C - A^T(y) - B^T(t) \\
\nabla_y L_\mu &= \frac{\partial(a^T y + \langle Z + C - A^T(y) - B^T(t), X\rangle)}{\partial y} \\
&= \frac{\partial(a^T y + \langle -A^T(y), X\rangle)}{\partial y} \\
&= \frac{\partial(a^T y - \langle y, A(X)\rangle)}{\partial y} \\
&= \frac{\partial(a^T y - tr(y \sum_{i=1}^k A_i X))}{\partial y} \\
&= a - tr(\textstyle\sum_{i=1}^k A_i X) \\
&= a - A(X)
\end{aligned}$$

We use without proof that, $\dfrac{\partial \det W}{\partial W} = (\det W)(W^{-1})$

$$\begin{aligned}
\text{Thus,} \quad \nabla_Z L_\mu &= \frac{\partial(-\mu \log \det Z)}{\partial Z} \\
&= (\det Z)(Z^{-1}) \cdot \frac{1}{\det Z} \cdot -\mu \\
&= \mu Z^{-1}
\end{aligned}$$

when there are no singularities (i.e., assume that Z is invertible).

3.4.3 The Central Trajectory

Because log $\det Z$ and log t_i are strictly concave, there is a unique solution $(X_\mu, y_\mu, t_\mu, Z_\mu)$ for each $0 \le \mu \le \infty$. This set of solutions, parameterized by μ, is called the central trajectory.

Given a point (X, y, t, Z) that is on the central trajectory:

$$\mu = \frac{tr(ZX)}{n} = \frac{t^T(b - B(X))}{m} = \frac{tr(ZX) + t^t(b - B(X))}{n + m} \tag{3.7}$$

Proof:

$$\begin{aligned}
\mu t^{-1} &= b - B(X) \quad \text{(from (3.5))} \\
\mu * m &= t^T(b - B(X)) \\
\frac{m}{n+m}\mu &= \frac{t^T(b - B(X))}{n+m}
\end{aligned}$$

$$\begin{aligned}
\mu Z^{-1} &= X \quad \text{(from (3.6), assuming } Z \text{ invertible)} \\
\mu I &= XZ \\
\mu * tr(I) &= tr(XZ) \\
\mu * n &= tr(XZ) \\
\frac{n}{m+n}\mu &= \frac{tr(ZX)}{m+n}
\end{aligned}$$

$$\mu = \frac{tr(ZX) + t^T(b - B(X))}{n+m}$$

□

Note that $tr(ZX) + t^T(b - B(X))$ is the duality gap.

Proof:

$$\begin{aligned}
Z &= A^T(y) + B^T(t) - C \\
\langle Z, X\rangle &= \sum y_i\langle A_i, X\rangle + \sum t_j\langle B_j, X\rangle - \langle C, X\rangle \\
tr(ZX) + t^Tb - t^TB(X) &= \sum y_i\langle A_i, X\rangle + \sum t_j\langle B_j, X\rangle - \langle C, X\rangle + t^Tb - t^TB(X) \\
tr(ZX) + t^Tb - t^TB(X) &= y^Ta + t^Tb - \langle C, X\rangle
\end{aligned}$$

since $a - A(X) = 0$ gives $y^Ta = \sum y_i\langle A_i, X\rangle$.

□

3.5 The Interior-Point Algorithm

The actual algorithm by Helmberg et al. is as follows:

We start with a quadruple (X, y, t, Z) such that $X \succ 0, Z \succ 0, t > 0,$ and $b - B(X) > 0$. We calculate the parameter μ such that:

$$\mu = \frac{tr(ZX) + t^T(b - B(X))}{2(n+m)}, \tag{3.8}$$

where the division by 2 is a heuristic known to perform well from linear programming [2].

Then we find the directions $\Delta X, \Delta y, \Delta t, \Delta Z$ so that the new point $(X+\Delta X, y+\Delta y, t+\Delta t, Z+\Delta Z)$ lies on the central trajectory at the value μ. However, a linearization is required because (3.5) and (3.6) are nonlinear. For the algorithm, Helmberg et al. use the linearization: $ZX - \mu I = 0$.

Thus we rewrite (3.3) to (3.6) as the function:

$$F_\mu(s) = F_\mu(X,y,t,Z) := \left\{ \begin{array}{l} Z + C - A^T(y) - B^T(t) \\ a - A(X) \\ t \circ (b - B(X)) - \mu e \\ ZX - \mu I \end{array} \right\} . := \left\{ \begin{array}{l} F_d \\ F_p \\ F_{tB} \\ F_{ZX} \end{array} \right\}$$

The solution s^* to $F_\mu(s) = 0$ satisfies the first order optimality conditions (3.3) to (3.6) and thus is the optimal solution to the barrier problem. In order to find the direction $\Delta s = (\Delta X, \Delta y, \Delta t, \Delta Z)$ to s^*, we use Newton's Method.

Thus, $F_\mu + \Delta F_\mu(\Delta s) = 0$, where we are given

$$\Delta F_\mu(\Delta s) = \Delta F_\mu(\Delta X, \Delta y, \Delta t, \Delta Z) = \left\{ \begin{array}{l} \Delta Z - A^T(\Delta y) - B^T(\Delta t) \\ -A(\Delta X) \\ \Delta t \circ (b - B(X)) - t \circ B(\Delta X) \\ Z\Delta X + \Delta Z X \end{array} \right\}.$$

The direction Δs is the solution to the system of equations:

$$\Delta Z - A^T(\Delta y) - B^T(\Delta t) = -F_d \tag{3.9}$$

$$-A(\Delta X) = -F_p \tag{3.10}$$

$$\Delta t \circ (b - B(X)) - t \circ B(\Delta X) = -F_{tB} \tag{3.11}$$

$$Z\Delta X + \Delta Z X = -F_{ZX} \tag{3.12}$$

This linear system can be solved for $(\Delta X, \Delta y, \Delta t, \Delta Z)$.

$$\Delta Z = -F_d + A^T(\Delta y) + B^T(\Delta t) \tag{3.13}$$

Note that ΔZ is the sum of symmetric matrices and, thus, is symmetric. Then this is substituted into (3.12) to get:

$$\Delta \tilde{X} = \mu Z^{-1} - X + Z^{-1} F_d X - Z^{-1}(A^T(\Delta y) + B^T(\Delta t))X \tag{3.14}$$

Proof:

$$\begin{aligned}
Z\Delta X + \Delta Z X &= -F_{ZX} \\
Z\Delta X + (-F_d + A^T(\Delta y) + B^T(\Delta t))X &= -F_{ZX} \\
Z\Delta X + (-F_d + A^T(\Delta y) + B^T(\Delta t))X &= -ZX + \mu I \\
Z\Delta X + ZX - \mu I + (-F_d + A^T(\Delta y) + B^T(\Delta t))X &= 0 \\
Z\Delta X + ZX - \mu I - F_d X + (A^T(\Delta y) + B^T(\Delta t))X &= 0 \\
\Delta X + X - \mu Z^{-1} - Z^{-1}F_d X + Z^{-1}(A^T(\Delta y) + B^T(\Delta t))X &= 0 \\
\Delta X = -X + \mu Z^{-1} + Z^{-1}F_d X - Z^{-1}(A^T(\Delta y) + B^T(\Delta t))X &
\end{aligned}$$

□

Let ΔX be $\Delta\tilde{X}$. Note that $\Delta\tilde{X}$ is not generally symmetric. Since we extended $\langle\cdot,\cdot\rangle$ by mapping skew-symmetric matrices to zero, the projection ΔX of $\Delta\tilde{X}$ onto the symmetric subspace takes the same value under operator $A(\cdot)$.

Substituting (3.14) into (3.10), we get the first equation for Δy and Δt:

$$O_{11}(\Delta y) + O_{12}(\Delta t) = v_1 \tag{3.15}$$

where O_{11} and O_{12} are linear operators:

$$\begin{aligned}
O_{11}(\cdot) &:= A(Z^{-1}A^T(\cdot)X) \\
O_{12}(\cdot) &:= A(Z^{-1}B^T(\cdot)X)
\end{aligned}$$

and v_1 is the vector:

$$v_1 := \mu A(Z^{-1}) - a + A(Z^{-1}F_d X)$$

Proof:

$$\begin{aligned}
\Delta X = -X + \mu Z^{-1} + Z^{-1}F_d X - Z^{-1}(A^T(\Delta y) + B^T(\Delta t))X& \\
-A(\Delta X) &= -F_p \\
-A(-X + \mu Z^{-1} + Z^{-1}F_d X - Z^{-1}(A^T(\Delta y) + B^T(\Delta t))X) &= -F_p \\
A(X) - A(\mu Z^{-1}) - A(Z^{-1}F_d X) + A(Z^{-1}(A^T(\Delta y) + B^T(\Delta t))X) &= -F_p \\
A(X) - A(\mu Z^{-1}) - A(Z^{-1}F_d X) + A(Z^{-1}A^T(\Delta y)X) + A(Z^{-1}B^T(\Delta t)X) &= -F_p \\
O_{11}(\Delta y) + O_{12}(\Delta t) &= \mu A Z^{-1} - A(X) + A(Z^{-1}F_d X) - F_p \\
O_{11}(\Delta y) + O_{12}(\Delta t) &= \mu A Z^{-1} - A(X) + A(Z^{-1}F_d X) - a + A(X) \\
O_{11}(\Delta y) + O_{12}(\Delta t) &= \mu A Z^{-1} + A(Z^{-1}F_d X) - a \\
O_{11}(\Delta y) + O_{12}(\Delta t) &= v_1
\end{aligned}$$

□

At last, substituting (3.14) into (3.11), we get:

$$O_{21}(\Delta y) + O_{22}(\Delta t) = v_2 \tag{3.16}$$

where O_{21} and O_{22} are linear operators:

$$\begin{aligned} O_{21}(\cdot) &:= B(Z^{-1}A^T(\cdot)X) \\ O_{22}(\cdot) &:= (b - B(X)) \circ t^{-1} \circ (\cdot) + B(Z^{-1}B^T(\cdot)X) \end{aligned}$$

and v_2 is the vector:

$$v_2 := \mu t^{-1} - b + \mu B(Z^{-1}) + B(Z^{-1}F_d X)$$

Proof:

$$\Delta X = -X + \mu Z^{-1} + Z^{-1}F_d X - Z^{-1}(A^T(\Delta y) + B^T(\Delta t))X$$

$$\Delta t \circ (b - B(X)) - t \circ B(\Delta X) = -F_{tB}$$

$$\Delta t \circ (b - B(X)) - t \circ B(-X + \mu Z^{-1} + Z^{-1}F_d X - Z^{-1}(A^T(\Delta y) + B^T(\Delta t))X) = -F_{tB}$$

$$\Delta t \circ (b - B(X)) + t \circ B(X) - \mu t \circ B(Z^{-1}) - t \circ B(Z^{-1}F_d X) +$$
$$t \circ B(Z^{-1}(A^T(\Delta y)X) + t \circ B(Z^{-1}B^T(\Delta t)X) = -F_{tB}$$

$$t^{-1} \circ \Delta t \circ (b - B(X)) + B(X) - \mu B(Z^{-1}) - B(Z^{-1}F_d X) + B(Z^{-1}(A^T(\Delta y)X) + B(Z^{-1}B^T(\Delta t)X)$$
$$= -t^{-1} \circ F_{tB}$$

$$t^{-1} \circ \Delta t \circ (b - B(X)) + B(X) - \mu B(Z^{-1}) - B(Z^{-1}F_d X) + O_{21}(\Delta y) + B(Z^{-1}B^T(\Delta t)X) = -t^{-1} \circ F_{tB}$$

$$O_{22}(\Delta t) + B(X) - \mu B(Z^{-1}) - B(Z^{-1}F_d X) + O_{21}(\Delta y) = -t^{-1} \circ F_{tB}$$

$$O_{21}(\Delta y) + O_{22}(\Delta t) + B(X) = \mu B(Z^{-1}) + B(Z^{-1}F_d X) - t^{-1} \circ F_{tB}$$

$$O_{21}(\Delta y) + O_{22}(\Delta t) + B(X) = \mu B(Z^{-1}) + B(Z^{-1}F_d X) - (b - B(X)) + \mu t^{-1}$$

$$O_{21}(\Delta y) + O_{22}(\Delta t) + B(X) = v_2 + B(X)$$

$$O_{21}(\Delta y) + O_{22}(\Delta t) = v_2$$

□

Note that (3.15) and (3.16) are linear in the entries of Δy and Δt and do not involve the matrices ΔX and ΔZ. So in conclusion, we solve (3.15) and (3.16) for Δy and Δt. Then we substitute into (3.13) to solve for ΔZ and then substitute into (3.12) to solve for $\Delta \tilde{X}$, for which we only take the symmetric part.

Now that we have $(\Delta X, \Delta y, \Delta t, \Delta Z)$, we step to $(X + \Delta X, y + \Delta y, t + \Delta t, Z + \Delta Z)$, However, this might violate the nonnegativity of t and $b - B(X)$ and the positive definiteness of either X or Z. So then we perform a "line search" to find constants α_p and α_d such that $t + \alpha_d\,\Delta t$ and $b - B(X + \alpha_p\,\Delta X)$ are strictly positive and $X + \alpha_p\,\Delta X$ and $Z + \alpha_d\,\Delta Z$ are positive definite. Given α_p and α_d, we step to the new point:

$$X + \alpha_p \Delta X \tag{3.17}$$

$$y + \alpha_d \Delta y \tag{3.18}$$

$$t + \alpha_d \Delta t \tag{3.19}$$

$$Z + \alpha_d \Delta Z \tag{3.20}$$

We then update μ using (3.8) and then repeat. The algorithm continues until (X, y, t, Z) satisfies primal feasibility and dual feasibility and the duality gap is sufficiently small. Thus concludes the interior-point algorithm.

3.6 MATLAB Code

3.6.1 Overview

Helmberg et al. provide MATLAB code that implements the algorithm described above. However, it solves a SDP with no inequality constraints (thus, the b and $B(X)$ terms disappear in the primal problem and the b^T, $B^T(t)$, and t terms disappear in the dual. It also uses the special case where $A(X) = diag(X)$ and $A^T(y) = Diag(y)$ (see example above). The code is available in Appendix A. The comments have been updated to reflect the numbered equations in this paper here to aid with clarity.

The function $psd\_id(C)$, takes an input of C, a symmetric matrix, and outputs phi, the optimal value of the SDP (i.e. $trace(CX)$; X, the optimal matrix of the DSDP; and y, the optimal dual vector. At each iteration of the algorithm, it outputs the following information:

$$\Big[\text{iteration num} \quad (\Delta y)^T \quad y^T \quad \alpha_p \quad \alpha_d \quad (a^T y - trace(CX)) \quad trace(CX) \quad a^T y\Big]$$

Note that $a^T y - trace(CX)$ is the duality gap, where $a^T y$ is the value of the dual and $trace(CX)$ is the value of the primal.

So for example, $psd\_id(C)$ with

$$C = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix},$$

outputs

$$\begin{bmatrix} 10.0000 & 0.0000 & -0.0000 & 2.0000 & 3.0000 \\ 1.0000 & 1.0000 & 0.0000 & 1.2500 & 1.2500 \end{bmatrix},$$

which signifies an iteration number of 10,

$$(\Delta y)^T = \begin{bmatrix} 0.0000 & -0.0000 \end{bmatrix},$$

$$y^T = \begin{bmatrix} 2.0000 & 3.0000 \end{bmatrix},$$

$\alpha_p = 1.0000$, $\alpha_d = 1.0000$, $(a^T y - trace(CX)) = 0.0000$, $trace(CX) = 1.2500$, and $a^T y = 1.2500$. Note that because the duality gap is 0.0000, the solution is optimal at 10 iterations.

The code for the algorithm is not intuitive, so here I will present an explanation of some of the code and its syntax. I begin with line 30, $dy = (Zi.*X)\ (mu*diag(Zi) - a)$. After a "MATLAB translation", this equation is equivalent to solving the equation $(Z^{-1} \circ X)\Delta y = \mu * diag(Z^{-1}) - a$, which is equivalent to solving (3.15).

Proof:

First, observe the following properties of $diag(M)$ and $Diag(u)$, where M is an n x n matrix and u is an n x 1 vector:

$$diag(Diag(u)) = u \tag{3.21}$$

$$Diag(diag(M)) = M \circ I \tag{3.22}$$

$$diag(M + N) = diag(M) + diag(N) \tag{3.23}$$

$$diag(\alpha M) = \alpha * diag(M) \tag{3.24}$$

$$diag(MN) = (M \circ N^T) * e \tag{3.25}$$

$$diag(MDiag(u)) = diag(M \circ u) \tag{3.26}$$

$$diag(MDiag(u)N) = (M \circ N^T) * u \tag{3.27}$$

From (3.15) and because there are no inequality constraints:

$$\begin{aligned} A(Z^{-1}A^T(\Delta y)X) &= \mu * A(Z^{-1}) - a + A(Z^{-1}F_d X) \\ diag(Z^{-1}Diag(\Delta y)X) &= \mu * diag(Z^{-1}) - a + diag(Z^{-1}(Z + C - A^T(y))X) \\ diag(Z^{-1}Diag(\Delta y)X) &= \mu * diag(Z^{-1}) - a + diag(Z^{-1}(Z + C - Diag(y))X) \end{aligned}$$

Then by (3.27),

$$(Z^{-1} \circ X)\Delta y = \mu * diag(Z^{-1}) - a + diag(Z^{-1}(Z + C - Diag(y))X),$$

since X is symmetric. Recall also that $Z = A^T(y) - C$.

$$\begin{aligned} &Z + C - A^T(y) = 0 \\ &Z + C - Diag(y) = 0 \\ &(Z^{-1} \circ X)\Delta y = \mu * diag(Z^{-1}) - a + diag(Z^{-1}(Z + C - Diag(y))X) \\ &(Z^{-1} \circ X)\Delta y = \mu * diag(Z^{-1}) - a + diag(Z^{-1} * 0 * X) \\ &(Z^{-1} \circ X)\Delta y = \mu * diag(Z^{-1}) - a \end{aligned}$$

□

Another notable part of the algorithm is the line search. In line 35, an α_p value is arbitrarily initialized to be 1. Then a Cholesky factorization is performed on $X + \alpha_p \Delta X$ (3.17), which decomposes a positive definite matrix into the product of a lower triangular matrix and its conjugate transpose. Note that, the actual factorization is not important, but rather the determination of whether the matrix itself is not positive definite if the factorization fails. This is because every positive definite matrix has a Cholesky decomposition [4]. Thus, we determine whether X is still positive definite. If not, the step size was too large and the new point for the next iteration of the algorithm will have left the feasible region. Thus, the step size is scaled down by an arbitrary factor of $\alpha_p * 0.8$ until the new point is feasible. A similar technique is used for the line search on the dual for α_d.

Something else to note about the algorithm is that, as expected, when the input matrix C is not symmetric, the code enters a seemingly infinite loop.

3.6.2 Limitations

I now present a list of generalizations/assumptions within the MATLAB code which can be improved upon to make a more flexible algorithm:

- The absence of inequality constraints
- $A(X) = diag(X)$ and $A^T(y) = Diag(y)$
- The tolerance is 1.0000e-06
- α_p is initialized to 1
- If α_p creates an infeasible new value for X, $\alpha_p = \alpha_p * 0.8$
- Once a feasible X is found, the α_p value is chosen as $0.95 * \alpha_p$
- If $\alpha_p + \alpha_d > 1.8$, then $\mu = \dfrac{\mu}{2}$

In general, all of the numerical values presented here seem arbitrary. Future development to improve this algorithm could include incorporating inequality constraints or conducting a sensitivity analysis on the parameters mentioned above. The analysis can test the effect of changes to these parameters on the performance and convergence of the algorithm.

Chapter 4

The Quadratic Assignment Problem

The Quadratic Assignment Problem (QAP) was first introduced in 1957 by Koopmans and Beckmann as a model for assigning economic activities to a set of locations [5]. The problem is normally associated with assigning n facilities to n locations, minimizing the quadratic objective that arises from the product of both the distance between these locations and the flow between these facilities.

4.1 QAP Formulation

The trace formulation of the QAP is as follows:

Quadratic Assignment Problem (QAP)

$$\begin{array}{rc} \text{minimize} & tr\,(AXBX^T - 2CX^T) \\ \text{subject to} & X \in \Pi_n \end{array} \tag{4.1}$$

where A and B are symmetric n x n matrices, C is a n x n matrix, and Π_n is the set of n x n permutation matrices. Note that we omit the parenthesis in the objective function for the majority of the rest of the report.

For example,

$$\Pi_2 = \left\{ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \right\}$$

,

$$\Pi_3 = \left\{ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \right\}$$

,

It is well-known that the QAP contains the TSP, so thus it is NP-Hard [5]. Cases of the QAP where $n = 30$ have been shown to be very hard to solve in practice [5].

I now present an example from the Network-Enabled Optimization System (NEOS) [6]. Consider the facilities location problem. Let A be the matrix where a_{ij} represents the flow between facilities i and j, B be the matrix where b_{kl} represents the distance between locations k and l, and C be the matrix where c_{ik} is the cost of placing facility i at location k. Let

$$A = \begin{pmatrix} 0 & 22 & 53 & 53 \\ 22 & 0 & 40 & 62 \\ 53 & 40 & 0 & 55 \\ 53 & 62 & 55 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 4 \\ 2 & 1 & 4 & 0 \end{pmatrix}, \text{ and } C = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

The optimal solution is

$$X = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

The interpretation of this solution is that assigning facility 3 to location A, facility 4 to location B, facility 1 to location C, and facility 2 to location D minimizes the sum of the product of the distance between these facilities and the flow between these facilities. The optimal $trAXBX^T - 2CX^T$ value is 790.

4.2 Applications of the QAP

For the following examples, we use the following formulation of the QAP:

Quadratic Assignment Problem (QAP)

$$\min_{\Pi \in S_n} \qquad \sum_{i=1}^{n} \sum_{j=1}^{n} a_{\Pi(i)\Pi(j)} b_{ij}$$

where S_n is the set of permutation matrices, $a_{\Pi(i)\Pi(j)}$ is the "flow" between facility $\Pi(i)$ and $\Pi(j)$, and b_{ij} is the distance location i and j. In other words, the product $a_{\Pi(i)\Pi(j)}b_{ij}$ is the cost of assigning facility $\Pi(i)$ to location i and facility $\Pi(j)$ to location j.

4.2.1 Campus Planning

Dickey and Hopkins used the QAP to model planning the layout of a college campus [7]. Suppose a college wants to build a set of n buildings, each with its own purpose (i.e. campus center, dining hall, library, dorm, etc.). The campus has n predetermined locations in which these buildings can be built. For each building, $\Pi(i)$ and $\Pi(j)$, there is a number of people, $a_{\Pi(i)\Pi(j)}$, that walk between those buildings per week, otherwise known as the traffic/flow between those buildings. For each location i and j, b_{ij} is the distance between those locations. The authors' objective is to minimize the total weekly walking distance between buildings.

4.2.2 Hospital Layout

Elshafei adapted the QAP to the design and layout of Ahmed Maher Hospital in Cairo, which is composed of 6 departments, each in a separate building [8]. The formulation of this problem is very similar to that of the campus planning problem, except now the objective is to minimize patient flow between buildings per year. Though the mathematical model is the same as the campus planning, this application is noteworthy because of the problem's potential for high societal impact. Elshafei describes how one building, the Out-patient department, had become extremely overcrowded with patients as they must move between the 17 clinics in the building. By applying the QAP to the hospital's layout, he aimed to minimize the total distance traveled by patients, hoping to decrease the amount of medical delays and bottlenecks within buildings.

4.2.3 Minimizing wire length

Steinberg described a "backboard wiring" problem that consisted of placing computer components to minimize the total amount of wiring required to connect them [9]. Let $a_{\Pi(i)\Pi(k)}$ be the number of wires that connects components $\Pi(i)$ and $\Pi(j)$ and let b_{ij} be the "distance" between locations i and j on the backboard. Steinberg considered multiple metrics to measure distance including the 1-norm, the 2-norm, and the squared 2-norm.

4.2.4 Typewriter Keyboard Design

Burkard and Offermann used QAPs to arrange keys on a keyboard such that the time needed to write a text is minimized [10]. Suppose a keyboard has n symbols that need to be placed in n positions. Then $a_{\Pi(i)\Pi(j)}$ denotes the frequency of the appearance of the pair of symbols $\Pi(i)$ and $\Pi(j)$ and b_{ij} denotes the time needed to press the key in position i after pressing the key in position j. The optimal solution for the QAP then minimizes the average time for writing a piece of text.

4.3 SDP Relaxations

I now present Zhao, Karish, Rendl and Wolkowicz's *Semidefinite Programming Relaxations for the Quadratic Assignment Problems.* They present a semidefinite relaxation, which in general are successful in providing tight relaxations for hard combinatorial problems, such as the QAP. [11]

4.3.1 Preliminaries

- The space of t x t symmetric matrices is denoted as $\mathcal{M}_n$.
- e is the vector of ones and e_i is the i-th unit vector. E is the matrix of all ones.
- $\mathcal{E} := \{X : Xe = X^T e = e\}$ is the set of matrices with rows and columns that sum to one. This is referred to as the set of assignment constraints.
- $\mathcal{Z} := \{X : X_{ij} \in \{0, 1\}\}$. This is referred to as the set of (0,1)-matrices.
- $\mathcal{O} := \{X : XX^T = X^T X = I\}$ is the set of orthogonal matrices, where I is the identity matrix.
- The Kronecker product, or tensor product, of two matrices is denoted as $A \otimes B$.

Let A be an m x n matrix and B be a p x q matrix. Then, $A \otimes B$ is the following mp x nq block matrix:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix}$$

From [11], note that

- $(A \otimes B)(U \otimes V) = AU \otimes BV$
- $vec(AYB) = (B^T \otimes A)vec(Y)$, where $vec(Y)$ denotes the vector formed from the columns of the matrix Y
- $(A \otimes B)^T = A^T \otimes B^T$

- The matrix $Y \in \mathcal{M}_{n^2+1}$ is partitioned into the following blocks:

$$\left(\begin{array}{c|ccc} y_{00} & Y^{01} & \dots & Y^{0n} \\ \hline Y^{10} & Y^{11} & \dots & Y^{1n} \\ \vdots & \vdots & \ddots & \vdots \\ Y^{n0} & Y^{n1} & \dots & Y^{nn} \end{array} \right),$$

where 0-based indexing is used for the rows and columns. When considering a block Y^{ik}, $Y_{(i,j),(k,l)}$ is the element (j,l) of block (i,k). $Y_{0,1:n^2}$, indicates the row vector produced by selecting columns 1 through n^2 of the 0-th row of Y.

The authors present a “first” SDP relaxation for the QAP. This comes from “lifting” the problem into a higher-dimensional space of symmetric matrices. The authors note that the QAP is a quadratic problem with a binary (0 or 1) choice in addition to various other constraints that ensure that X is a permutation matrix. Note that these permutation matrices can be represented by binary vectors $vec(X)$. This embedding in $\mathcal{M}_{n^2+1}$ is obtained by

$$\begin{pmatrix} 1 \\ vec(X) \end{pmatrix} \begin{pmatrix} 1, & vec(X)^T \end{pmatrix}.$$

For example, let $n = 3$ and

$$X = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

.

Then,

$$vec(X) = \left(\begin{array}{c} 0 \\ 1 \\ 0 \\ \hline 1 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 1 \end{array}\right).$$

.

$$\begin{pmatrix} 1 \\ vec(X) \end{pmatrix} \begin{pmatrix} 1, & vec(X)^T \end{pmatrix}$$

$$= \left(\begin{array}{c} 1 \\ \hline 0 \\ 1 \\ 0 \\ \hline 1 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 1 \end{array}\right) \left(\begin{array}{c|ccc|ccc|ccc} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

$$= \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

which we note is symmetric, positive semidefinite, and is a 10 x 10, or (3^2+1) x (3^2+1), matrix.

Zhao et al.'s relaxation comes from the dual of the (homogenized) Lagrangian dual. The authors note that the SDP relaxation is equivalent to the Lagrangian relaxation for certain constrained problems.

4.3.2 Converting the QAP

Permutation matrices Π_n can be characterized as the intersection of $(0,1)$ matrices with $\mathcal{E}$ and $\mathcal{O}$, i.e. $\Pi_n = \mathcal{E} \cap \mathcal{Z} = \mathcal{O} \cap \mathcal{Z}$. Recall that a matrix, A, is orthogonal if $AA^T = I$. The QAP can then be rewritten as:

$$\begin{array}{rc} \textbf{QAP}_{\mathcal{E}} & \\ \text{minimize} & tr\, AXBX^T - 2CX^T \\ \text{subject to} & XX^T = X^TX = I \\ & Xe = X^Te = e \\ & X_{ij}^2 - X_{ij} = 0, \quad \forall\, i,j \end{array} \tag{4.2}$$

We now prove that the constraints of (4.2) require that $X \in \Pi_n$ for the case of $n = 2$, making (4.2) an equivalent formulation to (4.1). However, the authors note that these constraints are redundant in $QAP_{\mathcal{E}}$, as having any two of the three constraints demonstrates that $X \in \Pi_n$ as shown below:

Proof: Let $n = 2$. Then

$$X = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \text{ and } e = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

,

Case 1: $XX^T = X^TX = I$ and $Xe = X^Te = e$

$$XX^T = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \begin{pmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \end{pmatrix} = \begin{pmatrix} x_{11}^2 + x_{12}^2 & x_{11}x_{21} + x_{12}x_{22} \\ x_{21}x_{11} + x_{22}x_{12} & x_{21}^2 + x_{22}^2 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{aligned} \implies x_{11}^2 + x_{12}^2 &= 1 \\ x_{11}x_{21} + x_{12}x_{22} &= 0 \\ x_{21}x_{11} + x_{22}x_{12} &= 0 \\ x_{21}^2 + x_{22}^2 &= 1 \end{aligned} \tag{4.3}$$

$$Xe = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} x_{11} + x_{12} \\ x_{21} + x_{22} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

.

$$X^Te = \begin{pmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} x_{11} + x_{21} \\ x_{12} + x_{22} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

.

$$\implies x_{11} + x_{12} = 1 \tag{4.4}$$

$$x_{21} + x_{22} = 1 \tag{4.5}$$

$$x_{11} + x_{21} = 1 \tag{4.6}$$

$$x_{12} + x_{22} = 1 \tag{4.7}$$

$$\implies x_{11} = 1 - x_{12} \tag{4.8}$$

$$\begin{aligned} x_{21} &= 1 - x_{22} \\ x_{21} &= 1 - (1 - x_{12}) \\ \implies x_{21} &= x_{12} \end{aligned}$$

$$x_{22} = 1 - x_{12} \tag{4.9}$$

Now plugging in (4.8) and (4.9) into (4.3), we get:

$$\begin{aligned} x_{11}x_{21} + x_{12}x_{22} &= 0 \\ (1 - x_{12})x_{12} + x_{12}(1 - x_{12}) &= 0 \\ x_{12} - x_{12}^2 + x_{12} - x_{12}^2 &= 0 \\ 2x_{12} - 2x_{12}^2 &= 0 \\ x_{12} - x_{12}^2 &= 0 \\ x_{12} &= x_{12}^2 \\ \implies x_{12} &\in \{0, 1\} \end{aligned}$$

Case 1a: $x_{12} = 0$

From (4.4) - (4.7):

$$\begin{aligned} \implies x_{21} &= 0 \\ x_{11} &= 1 \\ x_{22} &= 1 \end{aligned}$$

Thus,

$$X = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Case 1b: $x_{12} = 1$

From (4.4) - (4.7):

$$\begin{aligned} \implies x_{21} &= 1 \\ x_{11} &= 0 \\ x_{22} &= 0 \end{aligned}$$

Thus,

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Finally, we can conclude $X \in \Pi_2$.

Case 2: $XX^T = X^T X = I$ and $X_{ij}^2 - X_{ij} = 0$

$$XX^T = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \begin{pmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \end{pmatrix} = \begin{pmatrix} x_{11}^2 + x_{12}^2 & x_{11}x_{21} + x_{12}x_{22} \\ x_{21}x_{11} + x_{22}x_{12} & x_{21}^2 + x_{22}^2 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\implies x_{11}^2 + x_{12}^2 = 1 \tag{4.10}$$

$$x_{11}x_{21} + x_{12}x_{22} = 0 \tag{4.11}$$

$$x_{21}x_{11} + x_{22}x_{12} = 0 \tag{4.12}$$

$$x_{21}^2 + x_{22}^2 = 1 \tag{4.13}$$

$$\begin{aligned} X_{ij}^2 - X_{ij} &= 0 \\ X_{ij}^2 &= X_{ij} \\ \implies X_{ij} &= \{0, 1\} \end{aligned}$$

Case 2a: $x_{11} = 1$

From (4.10) - (4.13):

$$\begin{aligned} x_{12}^2 = 0 \implies x_{12} &= 0 \\ x_{21} &= 0 \\ x_{22}^2 = 1 \implies x_{22} &= 1 \end{aligned}$$

Thus,

$$X = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Case 2b: $x_{11} = 0$

From (4.10) - (4.13):

$$\begin{aligned} x_{12}^2 = 1 &\implies x_{12} = 1 \\ & \quad\; x_{22} = 0 \\ x_{21}^2 = 1 &\implies x_{21} = 1 \end{aligned}$$

Thus,

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Finally, $X \in \Pi_2$.

Case 3: $Xe = X^T e = e$ and $X_{ij}^2 - X_{ij} = 0$

$$Xe = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} x_{11} + x_{12} \\ x_{21} + x_{22} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

.

$$X^T e = \begin{pmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} x_{11} + x_{21} \\ x_{12} + x_{22} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

.

$$\begin{aligned} \implies x_{11} + x_{12} &= 1 \\ x_{21} + x_{22} &= 1 \\ x_{11} + x_{21} &= 1 \\ x_{12} + x_{22} &= 1 \end{aligned}$$

$$\begin{aligned} X_{ij}^2 - X_{ij} &= 0 \\ X_{ij}^2 &= X_{ij} \\ \implies X_{ij} &= \{0, 1\} \end{aligned}$$

Similar to the proofs above, $X \in \Pi_2$. Thus we can conclude for X with $n = 2$, $X \in \Pi_2$. It is clear to see that we can expand this proof to conclude that for X of size n, $X \in \Pi_n$. $\square$

Though these constraints are redundant, the authors note that they are not redundant in the SDP relaxation as shown below.

4.3.3 The Relaxation

The authors then present the SDP relaxation directly obtained from the QAP. This happens from lifting the vectors $x = vec(X)$ into the matrix space $\mathcal{M}_{n^2+1}$.

Let $X \in \Pi_n$ be a permutation matrix and let $x = vec(X)$ and $c = vec(C)$. The objective function for QAP then becomes

$$\begin{aligned} q(X) &= tr(AXBX^T - 2CX^T) \\ &= x^T(B \otimes A)x - 2c^T x \\ &= tr(xx^T B \otimes A - 2c^T x) \\ &= tr(L_Q Y_X), \end{aligned}$$

where L_Q and Y_X are (n^2+1) x (n^2+1) matrices defined as the following:

$$L_Q := \begin{pmatrix} 0 & -vec(C)^T \\ -vec(C) & B \otimes A \end{pmatrix},$$

and

$$Y_X := \begin{pmatrix} x_0 & x^T \\ x & xx^T \end{pmatrix}.$$

Recall that $\langle U, V\rangle = tr(UV^T)$ is the sum of all entries of $U \circ V : \langle U, V\rangle = e^T(U \circ V)e$. Note also that now the quadratic objective function of the QAP is a linear function in the SDP relaxation.

We now relax the combinatorial restriction of $Y_X, X \in \Pi_n$ with more manageable constraints. There are three constraints on the matrix Y: it is positive semidefinite, the top-left component $y_{00} = 1$, and it is rank-one. In order to guarantee that the matrix Y is built from a permutation X, the authors add additional constraints. For example, the $(0, 1)$ constraints $X_{ij}^2 - x_0 X_{ij} = 0$ are equivalent to the restriction that the diagonal of Y is equal to its first row (or column).

The following is equivalent to QAP:

$$\begin{array}{ll} \textbf{QAP}_{\mathcal{O}} & \\ \text{minimize} & tr(AXBX^T - 2CX^T) \\ \text{subject to} & XX^T = I \\ & X^T X = I \\ & ||Xe - e||^2 + ||X^T e - e||^2 = 0 \\ & X_{ij}^2 - X_{ij} = 0 \quad \forall\, i, j \end{array}$$

The authors then use Lagrangian duality to arrive at the SDP relaxation. They note that there is no duality gap between the Lagrangian relaxation and the dual, so solving the SDP relaxation is equivalent to solving the Lagrangian relaxation. This means we can find the optimal solution of the dual of the SDP in the Lagrangian relaxation of $(QAP_{\mathcal{E}})$ and then find the optimal matrix X where the Lagrangian reaches its minimum. This X is then a good approximation for the original QAP.

The authors first arrive at the following equation by adding the $(0,1)$ and row-column constraints to the objective function using Lagrange multipliers W_{ij} and u_0.

$$\mu_{\mathcal{O}} = \min_{XX^T=X^TX=I} \max_{W,u_0} tr(AXBX^T - 2CX^T) + \sum_{i,j} W_{ij}(X_{ij}^2 - X_{ij}) + u_0(||Xe-e||^2 + ||X^Te-e||^2))$$

Then the authors interchange the max and min to get

$$\begin{aligned}\mu_{\mathcal{O}} \geq \mu_{\mathcal{L}} := \max_{W,u_0} \min_{XX^T=X^TX=I} & \; tr(AXBX^T - 2CX^T) + \sum_{i,j} W_{ij}(X_{ij}^2 - X_{ij}) + \\ & u_0(||Xe-e||^2 + ||X^Te-e||^2).\end{aligned}$$

The authors then homogenize the objective function by multiplying by a constrained scalar x_0 and increasing the dimension of the problem by 1. This homogenization helps to simplify the transition to a SDP problem.

$$\begin{gathered}\mu_{\mathcal{O}} \geq \mu_{\mathcal{L}} = \max_{W} \min_{XX^T=X^TX=I, x_0^2=1} tr[AXBX^T + W(X \circ X)^T \\ +u_0(||Xe||^2 + ||X^Te||^2) - x_0(2C+W)X^T] - 2x_0u_0e^T(X+X^T)e + 2nu_0\end{gathered}$$

They then introduce a Lagrange multiplier w_0 for the constraint on x_0 and Lagrange multipliers S_b for $XX^T = I$ and S_o for $X^TX = I$ to the lower bound μ_R :

$$\begin{aligned}\mu_{\mathcal{O}} \geq \mu_{\mathcal{L}} \geq \mu_R := \max_{W,S_b,S_o,u_0,w_0} \min_{X,x_0} & \; tr[AXBX^T + u_0(||Xe||^2 + ||X^Te||^2) + W(X \circ X)^T \\ & +w_0x_0^2 + S_bXX^T + S_oX^TX] - tr(x_0(2C+W)X^T) \\ & -2x_0u_0e^T(X+X^T)e - w_0 - tr(S_b) - tr(S_o) + 2nu_0\end{aligned}$$

The authors note that there can be duality gaps in each of the Lagrangian relaxations.

Now define $x := vec(X), y^T := (x_0, x^T)$ and $w^T := (w_0, vec(W)^T)$.

Thus,

$$\begin{aligned}\mu_R := \max_{W,S_b,S_o,u_0} \min_{y} & \; y^T[L_Q + Arrow(w) + B^0Diag(S_b) \\ & +O^0Diag(S_o) + u_0D]y - w_0 - tr(S_b) - tr(S_o), \qquad (4.14)\end{aligned}$$

where L_Q is defined as above and the linear operators

$$Arrow(w) := \begin{pmatrix} w_0 & -\frac{1}{2}w_{1:n^2}^T \\ -\frac{1}{2}w_{1:n^2} & Diag(w_{1:n^2}) \end{pmatrix},$$

$$B^0Diag(S) := \begin{pmatrix} 0 & 0 \\ 0 & I \otimes S_b \end{pmatrix},$$

$$O^0Diag(S) := \begin{pmatrix} 0 & 0 \\ 0 & S_o \otimes I \end{pmatrix},$$

$$D := \begin{pmatrix} n & -e^T \otimes e^T \\ -e \otimes e & I \otimes E \end{pmatrix} + \begin{pmatrix} n & -e^T \otimes e^T \\ -e \otimes e & E \otimes I \end{pmatrix}.$$

.

We note without proof that there is a hidden semidefinite constraint in (4.14), i.e. the inner minimization problem is bounded below only if the Hessian of the quadratic form is positive semidefinite. In this case, the quadratic form has minimum value 0. This produces the following SDP:

$$\begin{array}{ll} \mathbf{D}_{\mathcal{O}} & \\ \text{maximize} & -w_0 - tr(S_b) - tr(S_o) \\ \text{subject to } L_Q + Arrow(w) + B^0Diag(S_b) + O^0Diag(S_o) + u_0 D \succeq 0 & \end{array}$$

The authors then arrive at an SDP relaxation of (QAP_O) as the Lagrangian dual of $D_{\mathcal{O}}$. They introduce a dual matrix variable $Y \succeq 0$:

$$\begin{array}{rl} \mathbf{SDP}_{\mathcal{O}} & \\ \text{minimize} & tr\, L_Q Y \\ \text{subject to} & b^0diag(Y) = I \\ & o^0diag(Y) = I \\ & arrow(Y) = e_0 \\ & tr(DY) = 0 \\ & Y \succeq 0 \end{array} \tag{4.15}$$

where the arrow operator, acting on the $(n^2 + 1)$ x $(n^2 + 1)$ matrix Y is the adjoint operator to Arrow$(\cdot)$ and is defined by:

$$arrow(Y) := diag(Y) - (0, Y_{0,1:n^2})^T,$$

i.e. the arrow constraint guarantees that the diagonal and 0-th row (or column) are identical. For example, let

$$X = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix},\ x = \begin{pmatrix} x_{11} \\ x_{12} \\ x_{21} \\ x_{22} \end{pmatrix},$$

,

$$Y_X := \begin{pmatrix} x_0 & x^T \\ x & xx^T \end{pmatrix} = \left(\begin{array}{c|cc|cc} 1 & x_{11} & x_{21} & x_{12} & x_{22} \\ \hline x_{11} & x_{11}^2 & x_{21}x_{11} & x_{12}x_{11} & x_{22}x_{11} \\ x_{21} & x_{11}x_{21} & x_{21}^2 & x_{12}x_{21} & x_{22}x_{21} \\ \hline x_{12} & x_{11}x_{12} & x_{21}x_{12} & x_{12}^2 & x_{22}x_{12} \\ x_{22} & x_{11}x_{22} & x_{21}x_{22} & x_{12}x_{22} & x_{22}^2 \end{array}\right)$$

$$arrow(Y) := diag(Y) - (0, Y_{0,1:n^2})^T$$

$$= \left(\begin{array}{c} 1 \\ \hline x_{11}^2 \\ x_{21}^2 \\ x_{12}^2 \\ x_{22}^2 \end{array}\right) - \left(\begin{array}{c} 0 \\ \hline x_{11} \\ x_{21} \\ x_{12} \\ x_{22} \end{array}\right)$$

$$= \left(\begin{array}{c} 1 \\ \hline x_{11}^2 - x_{11} \\ x_{21}^2 - x_{21} \\ x_{12}^2 - x_{12} \\ x_{22}^2 - x_{22} \end{array}\right).$$

The block-0 diagonal operator and off-0 diagonal operator acting on Y are defined by

$$b^0diag(Y) := \sum_{k=1}^{n} Y_{(k,\cdot),(k,\cdot)}$$

and

$$o^0diag(Y) := \sum_{k=1}^{n} Y_{(\cdot,k),(\cdot,k)},$$

where $Y_{(k,\cdot),(k,\cdot)}$ indicates selecting the diagonal blocks of Y and $Y_{(\cdot,k),(\cdot,k)}$ indicates selecting the (k,k)th entry of each block of Y. Let us consider the example of Y_X given above. Then

$$b^0diag(Y) := \sum_{k=1}^{n} Y_{(k,\cdot),(k,\cdot)} = Y_{(1,\cdot),(1,\cdot)} + Y_{(2,\cdot),(2,\cdot)}$$
$$= \begin{pmatrix} x_{11}^2 & x_{21}x_{11} \\ x_{11}x_{21} & x_{21}^2 \end{pmatrix} \begin{pmatrix} x_{12}^2 & x_{22}x_{12} \\ x_{12}x_{22} & x_{22}^2 \end{pmatrix}$$

$$o^0diag(Y) := \sum_{k=1}^{n} Y_{(\cdot,k),(\cdot,k)} = Y_{(\cdot,1),(\cdot,1)} + Y_{(\cdot,2),(\cdot,2)}$$
$$= \begin{pmatrix} x_{11}^2 + x_{12}x_{11} + x_{11}x_{12} + x_{12}^2 & x_{21}x_{11} + x_{22}x_{11} + x_{21}x_{12} + x_{22}x_{12} \\ x_{11}x_{21} + x_{12}x_{21} + x_{11}x_{22} + x_{12}x_{22} & x_{21}^2 + x_{22}x_{21} + x_{21}x_{22} + x_{22}^2 \end{pmatrix}$$

These are the adjoint operators of $B^0Diag(\cdot)$ and $O^0Diag(\cdot)$, respectively. The block-0-diagonal operator guarantees that the sum of the diagonal blocks equals the identity. The off-0-diagonal operator guarantees that the trace of each diagonal block is 1, while the trace of the off-diagonal blocks is 0. These come from the orthogonality constraints, $XX^T = I$ and $X^TX = I$, respectively.

The authors then prove that the optimal solution to $(SDP_{\mathcal{O}})$ provides the permutation matrix $X = Mat(x)$ that solves the QAP, where $Mat(X)$ denotes the matrix formed from the vector x.

Chapter 5

NEOS Solver

5.1 NEOS Background

The NEOS solver is a web service used for solving numerical optimization problems. It is hosted by the Wisconsin Institute for Discovery and the University of Wisconsin in Madison and makes use of distributed high-performance machines. Though it solves many types of optimization problems we will be focusing on its Semidefinite Programming optimization solver "csdp", which uses a computationally optimized version of the interior point method described above.

5.2 NEOS Example

We present the following linear programming example.

$$\begin{array}{rlcll}
\text{maximize} & x_1 & + & 2x_2 & \\
\text{subject to} & x_1 & & & \leq 4 \\
 & x_1 & + & x_2 & = 7 \\
 & x_1, & & x_2 & \geq 0
\end{array}$$

Recall the formulation of a SDP from (3.2). Then

$$a_1 = 7 \qquad a = \begin{bmatrix} 7 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad A(x) = \begin{bmatrix} \langle A_1, X \rangle \end{bmatrix}$$

$$A^T(y) = y_1 A_1$$

$$b_1 = 4 \qquad b = \begin{bmatrix} 4 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \qquad B(x) = \begin{bmatrix} \langle B_1, X \rangle \end{bmatrix}$$

$$B^T(t) = t_1 B_1$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \qquad X = \begin{bmatrix} x_1 & 0 \\ 0 & x_2 \end{bmatrix}$$

Thus, the primal problem becomes:

$$\begin{aligned} \text{maximize} \quad & tr(\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 & 0 \\ 0 & x_2 \end{bmatrix}) \\ \text{subject to} \quad & \begin{bmatrix} 7 \end{bmatrix} - \begin{bmatrix} \langle \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} x_1 & 0 \\ 0 & x_2 \end{bmatrix} \rangle \end{bmatrix} = 0 \\ & \begin{bmatrix} 4 \end{bmatrix} - \begin{bmatrix} \langle \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} x_1 & 0 \\ 0 & x_2 \end{bmatrix} \rangle \end{bmatrix} \geq 0 \\ & \begin{bmatrix} x_1 & 0 \\ 0 & x_2 \end{bmatrix} \succeq 0 \end{aligned}$$

We can easily verify through matrix algebra that these formulations are equivalent.

The dual problem then becomes:

$$\begin{aligned} \text{minimize} \quad & \begin{bmatrix} 7 \end{bmatrix} \begin{bmatrix} y_1 \end{bmatrix} + \begin{bmatrix} 4 \end{bmatrix} \begin{bmatrix} t_1 \end{bmatrix} \\ \text{subject to} \quad & y_1 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + t_1 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \succeq 0 \\ & y_1 \in \mathbb{R}, t_1 \in \mathbb{R}_+ \end{aligned}$$

Again, through matrix algebra, we can see that this formulation is equivalent to the dual formulation:

$$\begin{aligned} \text{minimize} \quad & 7y_1 + 4t_1 \\ \text{subject to} \quad & y_1 + t_1 \geq 1 \\ & y_1 \geq 2 \\ & y_1 \in \mathbb{R}, \quad t_1 \geq 0 \end{aligned}$$

However, the NEOS Solver cannot handle inequality constraints in the primal problem, so we present a new LP with only equality constraints:

Primal

$$\begin{aligned} \text{maximize} \quad & x_1 + 2x_2 \\ \text{subject to} \quad & 5x_1 + x_3 = 4 \\ & x_1 + 3x_2 = 7 \\ & x_1, \quad x_2, \quad x_3 \geq 0 \end{aligned}$$

Dual

$$\begin{aligned} \text{minimize} \quad & 4y_1 + 7y_2 \\ \text{subject to} \quad & 5y_1 + y_2 \geq 1 \\ & 3y_2 \geq 2 \\ & y_1 \geq 0 \\ & y_1, \quad y_2 \in \mathbb{R} \end{aligned}$$

$$a_1 = 4 \qquad\qquad a_2 = 7$$

$$a = \begin{bmatrix} 4 \\ 7 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$A(x) = \begin{bmatrix} \langle A_1, X \rangle \\ \langle A_2, X \rangle \end{bmatrix} \quad A^T(y) = y_1 A_1 + y_2 A_2$$

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad X = \begin{bmatrix} x_1 & 0 & 0 \\ 0 & x_2 & 0 \\ 0 & 0 & x_3 \end{bmatrix}$$

$$Z = y_1 A_1 - y_2 A_2 - C = \begin{bmatrix} 5y_1 + y_2 - 1 & 0 & 0 \\ 0 & 3y_2 - 2 & 0 \\ 0 & 0 & y_1 \end{bmatrix}$$

Thus, the primal problem becomes:

$$\text{maximize} \qquad tr\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 & 0 & 0 \\ 0 & x_2 & 0 \\ 0 & 0 & x_3 \end{bmatrix} \right)$$

$$\text{subject to} \quad a = \begin{bmatrix} 4 \\ 7 \end{bmatrix} - \begin{bmatrix} \left\langle \begin{bmatrix} 5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} x_1 & 0 & 0 \\ 0 & x_2 & 0 \\ 0 & 0 & x_3 \end{bmatrix} \right\rangle \\ \\ \left\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} x_1 & 0 & 0 \\ 0 & x_2 & 0 \\ 0 & 0 & x_3 \end{bmatrix} \right\rangle \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The dual problem becomes:

$$\begin{array}{ll} \text{minimize} & \begin{bmatrix} 4 \\ 7 \end{bmatrix}^T \begin{bmatrix} y_1 & y_2 \end{bmatrix} \\ \text{subject to} & y_1 \begin{bmatrix} 5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} + y_2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \succeq 0 \\ & y \in \mathbb{R} \end{array} \tag{5.1}$$

5.3 Creating the NEOS File

The following is the SDPA Sparse File Format as described in [12]. The format was designed for SDP problems in which the matrices $F_i, i = 0, ...m$ are block diagonal with sparse blocks. The file format consists of 6 sections:

1. Comments. The file can begin with any line of comments. Each line of comments must begin with ''' or '*'.
2. The first line after the comments contains m, the number of constraint matrices. Additional text on this line after m is ignored.
3. The second line after the comments contains $nblocks$, the number of blocks in the block diagonal structure of the matrices. Additional text on this line after $nblocks$ is ignored.
4. The third line after the comments contains a vector of numbers that gives the sizes of the individual blocks. The special characters ',', '(', ')', '{', and '}' can be used as punctuation and are ignored. Negative numbers may be used to indicate that a block is actually a diagonal submatrix. Thus a block a size of "-5" indicates a 5 by 5 block in which the only diagonal elements are nonzero.
5. The fourth line after the comments contains the objective function vector c.
6. The remaining lines of the file contain entries in the constraint matrices, with one entry per line. The format for each line is $< matno >< blkno >< i >< j >$ $< entry >$. Here, $< matno >$ is the number of the matrix to which this entry belongs, $< blkno >$ specifies the block within this matrix, $< i >$ and $< j >$ specify a location within the block, and $< entry >$ gives the value of the entry in the matrix. Note that since all matrices are assumed to be symmetric, only entries in the upper triangle of a matrix are given.

Let us recall the example (5.1). Let the constraint matrix in (5.1) be rewritten as:

$$y_1F_1 + y_2F_2 - F_0 \succeq 0$$

where

$$F_1 = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, F_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \text{ and } F_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

We can decompose F_0 into the following block matrix:

$$F_0 = \left[\begin{array}{c|c|c} 1 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \end{array}\right] = \left[\begin{array}{c|c|c} \text{Block 1} & 0 & 0 \\ \hline 0 & \text{Block 2} & 0 \\ \hline 0 & 0 & \text{Block 3} \end{array}\right] \tag{5.2}$$

Following a similar structure for F_1 and F_2, the NEOS file format takes the following form:

(a) Comments. None

(b) m, the number of constraint matrices = 2

(c) $nblocks$, the number of blocks in the block diagonal structure of the matrices = 3 (as determined by (5.2))

(d) The sizes of the individual blocks = 1, 1, 1

(e) Objective function vector c = 4, 7

(f) Entries in the constraint matrices, with one entry per line in the format: $< matno >< blkno >< i >< j >< entry >$. $< matno >$ is the number of the matrix to which this entry belongs, $< blkno >$ specifies the block within this matrix, $< i >$ and $< j >$ specify a location within the block, and $< entry >$ gives the value of the entry in the matrix:

Thus for (5.2), each corresponding row in $< matno > < blkno > < i > < j > < entry >$ form is:

```
0  1  1  1  1.0
0  2  1  1  2.0
0  3  1  1  0.0
```

The corresponding Sparse File Format for the entire formulation is then:

```
2
3
1    1   1
4.0  7.0
0    1   1  1  1.0
0    2   1  1  2.0
0    3   1  1  0.0
1    1   1  1  5.0
1    2   1  1  0.0
1    3   1  1  1.0
2    1   1  1  1.0
2    2   1  1  3.0
2    3   1  1  0.0
```

We then upload this to the NEOS solver in a MATLAB *.m* file.

The solution returned by the solver is as follows:

```
***CSDP***
CSDP 6.2.0
Iter: 0 Ap: 0.00e+00 Pobj: 3.0000000e+01 Ad: 0.00e+00 Dobj: 0.0000000e+00
Iter: 1 Ap: 9.00e-01 Pobj: 7.0047891e+00 Ad: 1.00e+00 Dobj: 2.7935874e+01
Iter: 2 Ap: 1.00e+00 Pobj: 4.7772865e+00 Ad: 1.00e+00 Dobj: 1.4079995e+01
Iter: 3 Ap: 1.00e+00 Pobj: 4.7834505e+00 Ad: 9.00e-01 Dobj: 5.7130994e+00
Iter: 4 Ap: 1.00e+00 Pobj: 4.8575350e+00 Ad: 1.00e+00 Dobj: 5.3223575e+00
Iter: 5 Ap: 1.00e+00 Pobj: 4.8921266e+00 Ad: 1.00e+00 Dobj: 5.1245360e+00
Iter: 6 Ap: 1.00e+00 Pobj: 4.9083073e+00 Ad: 1.00e+00 Dobj: 5.0245101e+00
Iter: 7 Ap: 1.00e+00 Pobj: 4.9183341e+00 Ad: 1.00e+00 Dobj: 4.9764336e+00
Iter: 8 Ap: 1.00e+00 Pobj: 4.9248623e+00 Ad: 1.00e+00 Dobj: 4.9539101e+00
Iter: 9 Ap: 1.00e+00 Pobj: 4.9333327e+00 Ad: 9.00e-01 Dobj: 4.9353871e+00
Iter: 10 Ap: 1.00e+00 Pobj: 4.9329916e+00 Ad: 1.00e+00 Dobj: 4.9340169e+00
Iter: 11 Ap: 1.00e+00 Pobj: 4.9333105e+00 Ad: 1.00e+00 Dobj: 4.9333755e+00
Iter: 12 Ap: 1.00e+00 Pobj: 4.9333318e+00 Ad: 1.00e+00 Dobj: 4.9333345e+00
Iter: 13 Ap: 1.00e+00 Pobj: 4.9333332e+00 Ad: 1.00e+00 Dobj: 4.9333335e+00
Iter: 14 Ap: 1.00e+00 Pobj: 4.9333333e+00 Ad: 9.00e-01 Dobj: 4.9333334e+00
Success: SDP solved
Primal objective value: 4.9333333e+00
```

```
Dual objective value: 4.9333334e+00
Relative primal infeasibility: 4.90e-17
Relative dual infeasibility: 6.93e-10
Real Relative Gap: 1.54e-09
XZ Relative Gap: 1.88e-09
DIMACS error measures: 5.55e-17 0.00e+00 1.29e-09 0.00e+00 1.54e-09 1.88e-09
Elements time: 0.000073
Factor time: 0.000010
Other time: 0.088326
Total time: 0.088409


Solution:
6.666666871535291417e-02 6.666666788805729385e-01
1 1 1 1 1.275172612507736432e-08
1 2 1 1 4.936107289582052057e-09
1 3 1 1 6.666667000974137169e-02
2 1 1 1 7.999999999998541611e-01
2 2 1 1 2.066666666666715280e+00
2 3 1 1 7.289393479915185848e-13
```

Thus, the optimal value to the dual occurs at $y^* = \begin{bmatrix} 0.0667 & 0.667 \end{bmatrix}$.

The matrices returned by the results are:

$$Z = \begin{bmatrix} 5y_1 + y_2 - 1 & 0 & 0 \\ 0 & 3y_2 - 2 & 0 \\ 0 & 0 & y_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.0667 \end{bmatrix}$$

$$X = \begin{bmatrix} x_1 & 0 & 0 \\ 0 & x_2 & 0 \\ 0 & 0 & x_3 \end{bmatrix} = \begin{bmatrix} 0.79 & 0 & 0 \\ 0 & 2.06 & 0 \\ 0 & 0 & 0 \end{bmatrix} \implies x^* = \begin{bmatrix} 0.79 & 2.06 & 0 \end{bmatrix}.$$

5.4 Solving the QAP with NEOS

Using the SDP relaxation above, we can use this solver to solve QAPs. Let us consider an example of QAP where $n = 2$.

Let

$$A = \begin{bmatrix} 0 & 10 \\ 10 & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} 0 & 5 \\ 5 & 0 \end{bmatrix}.$$

.

Because $n = 2$ (i.e. we are only considering 2 facilities and 2 locations) and A and B are symmetric, this example does not provide a very interesting QAP. However, as we will see, this problem size is small in order to be manageable to present here in this report. To make the optimization problem nontrivial, we also introduce the cost matrix C, where

$$C = \begin{bmatrix} 3 & 1 \\ 1 & 10 \end{bmatrix}.$$

Thus, we expect that the optimal answer is

$$X^* = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

because of the high costs of placing facility 1 at location 1 and facility 2 at location 2.

Recall the SDP relaxation of the QAP from (4.15). We now apply this to our example:

$$X = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \quad x = \begin{pmatrix} x_{11} \\ x_{12} \\ x_{21} \\ x_{22} \end{pmatrix}$$

,

$$-vec(C) = \begin{bmatrix} -3 \\ -1 \\ -1 \\ -10 \end{bmatrix}$$

$$B \otimes A = \left[\begin{array}{c|c} b_{11}A & b_{21}A \\ \hline b_{21}A & b_{22}A \end{array}\right] = \left[\begin{array}{cc|cc} 0 & 0 & 0 & 50 \\ 0 & 0 & 50 & 0 \\ \hline 0 & 50 & 0 & 0 \\ 50 & 0 & 0 & 0 \end{array}\right]$$

$$L_Q := \begin{pmatrix} 0 & -vec(C)^T \\ -vec(C) & B \otimes A \end{pmatrix} = \left(\begin{array}{c|cc|cc} 0 & -3 & -1 & -1 & -10 \\ \hline -3 & 0 & 0 & 0 & 50 \\ -1 & 0 & 0 & 50 & 0 \\ \hline -1 & 0 & 50 & 0 & 0 \\ -10 & 50 & 0 & 0 & 0 \end{array}\right),$$

$$Y_X := \begin{pmatrix} x_0 & x^T \\ x & xx^T \end{pmatrix} = \left(\begin{array}{c|cc|cc} 1 & x_{11} & x_{21} & x_{12} & x_{22} \\ \hline x_{11} & x_{11}^2 & x_{21}x_{11} & x_{12}x_{11} & x_{22}x_{11} \\ x_{21} & x_{11}x_{21} & x_{21}^2 & x_{12}x_{21} & x_{22}x_{21} \\ \hline x_{12} & x_{11}x_{12} & x_{21}x_{12} & x_{12}^2 & x_{22}x_{12} \\ x_{22} & x_{11}x_{22} & x_{21}x_{22} & x_{12}x_{22} & x_{22}^2 \end{array}\right)$$

In order for NEOS to be able to solve this QAP, we need to reformulate the constraints in (4.15) such that they are all in the form $tr(W\,Y)$, where $W \in \mathcal{M}_{n+1}$.

Recall that the block-0-diagonal operator, $b^0diag(Y)$, guarantees that the sum of the diagonal blocks of Y equals the identity.

Thus,

$$\begin{pmatrix} x_{11}^2 & x_{21}x_{11} \\ x_{11}x_{21} & x_{21}^2 \end{pmatrix} + \begin{pmatrix} x_{12}^2 & x_{22}x_{12} \\ x_{12}x_{22} & x_{22}^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\implies x_{11}^2 + x_{12}^2 = 1 \tag{5.3}$$

$$x_{21}x_{11} + x_{22}x_{12} = 0 \tag{5.4}$$

$$x_{11}x_{21} + x_{12}x_{22} = 0 \tag{5.5}$$

$$x_{21}^2 + x_{22}^2 = 1 \tag{5.6}$$

Note that (5.5) is redundant. Thus the constraints (5.3), (5.4), and (5.6) can be converted into the following trace formulations using the following symmetric matrices:

$$tr\left(\left(\begin{array}{c|cc|cc} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) Y_X\right) = 1$$

$$tr\left(\left(\begin{array}{c|cc|cc} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0 \end{array}\right) Y_X\right) = 0$$

$$tr\left(\left(\begin{array}{c|cc|cc} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right) Y_X\right) = 1$$

Recall that the off-0-diagonal operator, $o^0 diag(Y)$ guarantees that the trace of each diagonal block is 1, while the trace of the off-diagonal blocks is 0.

$$\implies x_{11}^2 + x_{21}^2 = 1 \tag{5.7}$$

$$x_{12}^2 + x_{22}^2 = 1 \tag{5.8}$$

$$x_{11}x_{12} + x_{21}x_{22} = 0 \tag{5.9}$$

$$x_{12}x_{11} + x_{22}x_{21} = 0 \tag{5.10}$$

Similarly, equations (5.6) - (5.7) can be rewritten as

$$tr\left(\left(\begin{array}{c|cc|cc} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) Y_X\right) = 1$$

$$tr\left(\left(\begin{array}{c|cc|cc} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right) Y_X\right) = 1$$

$$tr\left(\left(\begin{array}{c|cc|cc} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0 \end{array}\right) Y_X\right) = 0$$

$$tr\left(\left(\begin{array}{c|cc|cc} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0 \end{array}\right) Y_X\right) = 0$$

Recall that the arrow constraint guarantees that the diagonal and 0-th row (or column) are identical.

$$arrow(Y) := diag(Y) - (0, Y_{0,1:n^2})^T = e_0$$

$$diag(Y) - (0, Y_{0,1:n^2})^T = \left(\begin{array}{c} 1 \\ \hline x_{11}^2 \\ x_{21}^2 \\ x_{12}^2 \\ x_{22}^2 \end{array}\right) - \left(\begin{array}{c} 0 \\ \hline x_{11} \\ x_{21} \\ x_{12} \\ x_{22} \end{array}\right)$$

$$= \left(\begin{array}{c} 1 \\ \hline x_{11}^2 - x_{11} \\ x_{21}^2 - x_{21} \\ x_{12}^2 - x_{12} \\ x_{22}^2 - x_{22} \end{array}\right) = \left(\begin{array}{c} 1 \\ \hline 0 \\ 0 \\ 0 \\ 0 \end{array}\right) = e_0$$

$$\implies x_{11}^2 - x_{11} = 0 \tag{5.11}$$

$$x_{21}^2 - x_{21} = 0 \tag{5.12}$$

$$x_{12}^2 - x_{12} = 0 \tag{5.13}$$

$$x_{22}^2 - x_{22} = 0 \tag{5.14}$$

Similarly, we can rewrite (5.11) - (5.14) as

$$tr\left(\left(\begin{array}{c|cc|cc} 0 & -0.5 & 0 & 0 & 0 \\ \hline -0.5 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \cline{2-5} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) Y_X\right) = 0$$

$$tr\left(\left(\begin{array}{c|cc|cc} 0 & 0 & -0.5 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ -0.5 & 0 & 1 & 0 & 0 \\ \cline{2-5} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) Y_X\right) = 0$$

$$tr\left(\left(\begin{array}{c|cc|cc} 0 & 0 & -0.5 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ -0.5 & 0 & 1 & 0 & 0 \\ \cline{2-5} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) Y_X\right) = 0$$

$$tr\left(\left(\begin{array}{c|cc|cc} 0 & 0 & 0 & 0 & -0.5 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \cline{2-5} 0 & 0 & 0 & 0 & 0 \\ -0.5 & 0 & 0 & 0 & 1 \end{array}\right) Y_X\right) = 0$$

Finally,

$$D := \begin{pmatrix} n & -e^T \otimes e^T \\ -e \otimes e & I \otimes E \end{pmatrix} + \begin{pmatrix} n & -e^T \otimes e^T \\ -e \otimes e & E \otimes I \end{pmatrix}.$$

.

$$-e^T \otimes e^T = \begin{bmatrix} -1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 & -1 \end{bmatrix}$$

$$-e \otimes e = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

$$I \otimes E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \left[\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{array}\right]$$

$$E \otimes I = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \left[\begin{array}{cc|cc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}\right]$$

Thus,

$$D := \left(\begin{array}{c|cc|cc} 2 & -1 & -1 & -1 & -1 \\ \hline -1 & 1 & 1 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 \\ \cline{2-5} -1 & 0 & 0 & 1 & 1 \\ -1 & 0 & 0 & 1 & 1 \end{array}\right) + \left(\begin{array}{c|cc|cc} 2 & -1 & -1 & -1 & -1 \\ \hline -1 & 1 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & 1 \\ \cline{2-5} -1 & 1 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & 1 \end{array}\right)$$

.

$$= \left(\begin{array}{c|cc|cc} 4 & -2 & -2 & -2 & -2 \\ \hline -2 & 2 & 1 & 1 & 0 \\ -2 & 1 & 2 & 0 & 1 \\ \cline{2-5} -2 & 1 & 0 & 2 & 1 \\ -2 & 0 & 1 & 1 & 2 \end{array}\right)$$

$$tr\left(\left(\begin{array}{c|cc|cc} 4 & -2 & -2 & -2 & -2 \\ \hline -2 & 2 & 1 & 1 & 0 \\ -2 & 1 & 2 & 0 & 1 \\ \cline{2-5} -2 & 1 & 0 & 2 & 1 \\ -2 & 0 & 1 & 1 & 2 \end{array}\right) Y\right) = 0$$

Now that we have written the SDP in the appropriate trace formulation, we can write this in the following NEOS format:

```
12
1
5
1 0 1 1 1 0 0 0 0 0 0 0 0
0 1 1 2 -3
0 1 1 3 -1
0 1 1 4 -1
0 1 1 5 -10
0 1 2 5 50
0 1 3 4 50
1 1 2 2 1
1 1 3 3 1
2 1 2 3 0.5
2 1 4 5 0.5
3 1 3 3 1
3 1 5 5 1
4 1 2 2 1
4 1 3 3 1
5 1 4 4 1
5 1 5 5 1
6 1 2 3 0.5
6 1 4 5 0.5
7 1 2 3 0.5
7 1 4 5 0.5
8 1 2 1 -0.5
8 1 2 2 1
9 1 3 1 -0.5
9 1 3 3 1
10 1 3 1 -0.5
10 1 3 3 1
11 1 5 1 -0.5
11 1 5 5 1
12 1 1 1 4
12 1 1 2 -2
12 1 1 3 -2
12 1 1 4 -2
12 1 1 5 -2
12 1 2 2 2
12 1 2 3 1
```

```
12 1 2 4 1
12 1 2 5 0
12 1 3 3 2
12 1 3 4 0
12 1 3 5 1
12 1 4 4 2
12 1 4 5 1
12 1 5 5 2
```

Running this through NEOS, we get a result of

```
***CSDP***
CSDP 6.2.0 Iter: 0 Ap: 0.00e+00 Pobj: 0.0000000e+00 Ad: 0.00e+00 Dobj:
0.0000000e+00 Iter: 1 Ap: 8.10e-01 Pobj: 7.1573064e+01 Ad: 1.00e+00 Dobj:
3.6137687e+02 Iter: 2 Ap: 9.00e-01 Pobj: 5.6097143e+01 Ad: 1.00e+00 Dobj:
2.4056889e+02 Iter: 3 Ap: 9.00e-01 Pobj: 8.1840637e+01 Ad: 9.00e-01 Dobj:
1.0822547e+02 Iter: 4 Ap: 9.00e-01 Pobj: 9.3477282e+01 Ad: 9.00e-01 Dobj:
9.6114477e+01 Iter: 5 Ap: 9.00e-01 Pobj: 9.5748843e+01 Ad: 1.00e+00 Dobj:
9.6001781e+01 Iter: 6 Ap: 9.00e-01 Pobj: 9.5975021e+01 Ad: 1.00e+00 Dobj:
9.6000371e+01 Iter: 7 Ap: 9.00e-01 Pobj: 9.5997640e+01 Ad: 1.00e+00 Dobj:
9.6000324e+01 Iter: 8 Ap: 7.29e-01 Pobj: 9.5999266e+01 Ad: 1.00e+00 Dobj:
9.6000334e+01 Iter: 9 Ap: 9.00e-01 Pobj: 9.5999880e+01 Ad: 1.00e+00 Dobj:
9.5999972e+01 Iter: 10 Ap: 9.00e-01 Pobj: 9.5999988e+01 Ad: 1.00e+00 Dobj:
9.6000000e+01 Iter: 11 Ap: 9.00e-01 Pobj: 9.5999998e+01 Ad: 1.00e+00 Dobj:
9.6000001e+01 Iter: 12 Ap: 9.00e-01 Pobj: 9.6000000e+01 Ad: 1.00e+00 Dobj:
9.6000000e+01
Success: SDP solved
Primal objective value: 9.6000000e+01
Dual objective value: 9.6000000e+01
Relative primal infeasibility: 2.71e-10
Relative dual infeasibility: 3.30e-11
Real Relative Gap: 8.59e-10
XZ Relative Gap: 1.34e-09
DIMACS error measures: 4.06e-10 0.00e+00 6.60e-11 0.00e+00 8.59e-10 1.34e-09
Elements time: 0.000114
Factor time: 0.000026
Other time: 0.146613
Total time: 0.146753
```

```
Solution:
6.803568439823433778e+01 -6.231030688341003820e-01 -1.465566783148969243e+01
-6.379912196018396209e+00 4.899989571851686065e+01 4.017669631137942510e+00
-2.663732109682846794e+00 6.728580685520780591e+00 -7.501875109767191852e+00
1.150209635379108519e+01 2.073043064280190251e+01 1.111192206617806164e+02
1 1 1 1 4.444768826486286457e+02
1 1 1 2 -2.226027316663216311e+02
1 1 1 3 -2.232385519455731639e+02
1 1 1 4 -2.212384413235612328e+02
1 1 1 5 -2.226036566449621716e+02
1 1 2 2 2.906227942128041377e+02
1 1 2 3 1.114846378880911004e+02
1 1 2 4 1.111192206617806164e+02
1 1 2 5 -5.000000000000000000e+01
1 1 3 3 2.732387669398175944e+02
1 1 3 4 -5.000000000000000000e+01
1 1 3 5 1.111192206617806164e+02
1 1 4 4 2.712383370435842380e+02
1 1 4 5 1.114846378880911004e+02
1 1 5 5 2.773130998548965067e+02
2 1 1 1 1.000000000181110904e+00
2 1 1 2 2.376167431058704333e-09
2 1 1 3 9.999999976597725437e-01
2 1 1 4 9.999999975371330896e-01
2 1 1 5 2.432204255985238665e-09
2 1 2 2 2.414396749908329419e-09
2 1 2 3 -1.042168640638686077e-11
2 1 2 4 -2.742189118088287405e-11
2 1 2 5 2.312112553495608790e-09
2 1 3 3 9.999999976930158407e-01
2 1 3 4 9.999999745448275 86e-01
2 1 3 5 1.698277311473289179e-11
2 1 4 4 9.999999747416624609e-01
2 1 4 5 5.695538803599437962e-12
2 1 5 5 2.469924055218392698e-09
```

Which, by the results of the second matrix,

$$Y_X := \begin{pmatrix} x_0 & x^T \\ x & xx^T \end{pmatrix} = \left(\begin{array}{c|cc|cc} 1 & x_{11} & x_{21} & x_{12} & x_{22} \\ \hline x_{11} & x_{11}^2 & x_{21}x_{11} & x_{12}x_{11} & x_{22}x_{11} \\ x_{21} & x_{11}x_{21} & x_{21}^2 & x_{12}x_{21} & x_{22}x_{21} \\ \hline x_{12} & x_{11}x_{12} & x_{21}x_{12} & x_{12}^2 & x_{22}x_{12} \\ x_{22} & x_{11}x_{22} & x_{21}x_{22} & x_{12}x_{22} & x_{22}^2 \end{array}\right)$$

$$\approx \left(\begin{array}{c|cc|cc} 1 & 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ \hline 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

$$\begin{aligned} \implies x_{11}^2 &= 0 \\ x_{21}^2 &= 1 \\ x_{12}^2 &= 1 \\ x_{22}^2 &= 0 \end{aligned}$$

$$\implies X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

which are the results expected. Thus, we have demonstrated how to use the SDP relaxation of the QAP and the NEOS solver to solve for a solution and interpret its results.

Chapter 6

Conclusion and Future Work

Semidefinite programming is a powerful way to model combinatorial optimization problems to approximate their solutions. In this report we examined an interior point algorithm for solving SDPs and discussed the limitations and areas for expansion of the MATLAB code that implements this algorithm. Besides, modifying the MATLAB code for this method, other algorithms could also be studied that could provide solutions that converge faster.

We then went on to study an SDP relaxation of the QAP and used the NEOS solver to obtain solutions to this relaxation. Similar to the algorithm, other relaxations exist that could be examined to provide tighter bounds on the problem, creating more efficient optimization schemes.

Appendix A

MATLAB Code

```
function [phi,X, y] = psd_ip(C)
% solves: max trace(CX)  s.t. X psd, diag(X) = a; a = ones(n,1)/4
%         min a'y        s.t. Diag(y) − C psd, y unconstrained,
% input:  C ... symmetric matrix
% output: phi ... optimal value of primal, phi =trace(CX)
%         X   ... optimal primal matrix
%         y   ... optimal dual vector
% call:   [phi, X, y] = psd_ip(C);

%outputArg1 = inputArg1;
%outputArg2 = inputArg2;

digits = 6;                  % 6 significant digits of phi (tolerance)
[n,nl] = size(C);            % problem size
a = ones(n,1)/4;             % any a>0 works just as well
X = diag(a);                 % initial primal matrix is pos. def.
y = sum(abs(C))' * 1.1;      % initial y is chosen so that...
Z = diag(y) − C;             % intial dual slack Z is pos. def.
phi = a'*y;                  % initial dual cost
psi = C(:)' * X(:);          % initial primal cost
mu = Z(:)' * X(:)/(2*n);     % initial complementarity, (Eq. 2.6) ...
    with no inequality constraints
iter=0;                      % iteration count

disp(['    iter    alphap   alphad     gap     lower    upper']);

while phi − psi > max([1,abs(phi)]) * 10^(−digits)
    iter = iter + 1;     % start a new iteration
    Zi = inv (Z);       % inv(Z) is needed explicitly
    Zi = (Zi + Zi')/2; % symmetrize inv(Z)
    dy = (Zi.*X) \ (mu * diag(Zi) −a);  %solve for dy from (Eq. 2.13)
    dX = - Zi * diag(dy) * X + mu * Zi - X; %back substitute for ...
    dX (Eq. 2.12)
    dX = (dX + dX')/2; %symmetrize dX

%line search on primal
    alphap = 1;     %initial steplength
    [dummy,posdef] = chol(X + alphap * dX); %test if pos def
    while posdef > 0 % not pos def
        alphap = alphap * .8; %scale back (went too far with step ...
    size)
        [dummy,posdef] = chol(X + alphap *dX); %check to now see ...
    if pos def
    end
    if alphap < 1, alphap = alphap * .95; end % stay away from ...
    boundary
% line search on dual; dZ is handled implicitly: dZ = diag(dy) ...
    (Eq. 2.11);
    alphad = 1;     %initial steplength
    [dummy,posdef] = chol(Z + alphad * diag(dy));
    while posdef > 0 % not positive definite
        alphad = alphad * .8;
        [dummy,posdef] = chol(Z + alphad *diag(dy));
    end
    if alphad < 1, alphad = alphad * .95; end  % stay away from ...
    boundary
%update
    X = X + alphap * dX; %(Eq. 2.15)
    y = y + alphad * dy; %(Eq. 2.16)
    Z = Z + alphad * diag(dy); %(Eq. 2.18)
    mu = X(:)' * Z(:) / (2*n);  %(Eq. 2.6)
    if alphap + alphad > 1.8, mu = mu/2; end   %speed up for long ...
    steps
    phi = a' * y; psi = C(:)' * X(:);
%display current iteration
    disp([ iter dy' y' alphap alphad (phi-psi) psi phi]);

end % end of main loop
```

Appendix B

Interior Point Method Equation Cheat Sheet

Semidefinite Primal (SDP)

$$\begin{array}{rl} \text{maximize} & tr\,CX \\ \text{subject to} & A(X) \;= a \\ & B(X) \;\leq b \\ & X \;\succeq 0 \end{array}$$

where $\mathcal{M}_n$ is the vector space of symmetric $n \,\mathrm{x}\, n$ matrices

$$A : \mathcal{M}_n \to \mathbb{R}^k, \quad B : \mathcal{M}_n \to \mathbb{R}^m$$

$$C \in \mathcal{M}_n,\ a \in \mathbb{R}^k,\ b \in \mathbb{R}^m$$

$$A(X) = \begin{bmatrix} \langle A_1, X\rangle \\ \langle A_2, X\rangle \\ \vdots \\ \langle A_k, X\rangle \end{bmatrix}, \quad B(X) = \begin{bmatrix} \langle B_1, X\rangle \\ \langle B_2, X\rangle \\ \vdots \\ \langle B_m, X\rangle \end{bmatrix}$$

Semidefinite Dual

$$\begin{array}{rl} \text{minimize} & a^T y + b^T t \\ \text{subject to} & A^T(y) + B^T(t) - C \succeq 0 \\ & y \in \mathbb{R}^k, t \in \mathbb{R}^m_+ \end{array}$$

Dual Barrier Problem

$$\begin{array}{rl} \text{minimize} & a^T y + b^T t - \mu(\log \det Z + e^T \log t) \\ \text{subject to} & A^T(y) + B^T(t) - C \;= Z \\ & t \geq 0, Z \succeq 0 \end{array}$$

Lagrangian

$$L_\mu(X, y, t, Z) = a^T y + b^T t - \mu(\log\,\det Z + e^T \log t) + \langle Z + C - A^T(y) - B^T(t), X\rangle$$

First Order Conditions

$$\begin{aligned}
\nabla_X L_\mu &= Z + C - A^T(y) - B^T(t) = 0\\
\nabla_y L_\mu &= a - A(X) = 0\\
\nabla_t L_\mu &= b - B(X) - \mu t^{-1} = 0\\
\nabla_Z L_\mu &= X - \mu Z^{-1} = 0
\end{aligned}$$

Bibliography

[1] R. J. Vanderbei. Linear programming: Foundations and extensions. 4, 2014. doi: https://doi.org/10.1007/978-1-4614-7630-6.

[2] R. Vanderbei C. Helmberg, F. Rendl and H. Wolkowicz. An interior-point method for semidefinite programming. *SIAM Optimization*, 6(2):342–361, May 1996.

[3] E. de Klerk. Aspects of semidefinite programming: Interior point algorithms and selected applications. 1, 2002. doi: https://doi.org/10.1007/b105286.

[4] MATLAB. chol - cholesky factorization documentation. 2019. doi: https://www.mathworks.com/help/matlab/ref/chol.html.

[5] Renata Sotirov. Sdp relaxations for some combinatorial optimization problems. *Handbook of Semidefinite, Conic and Polynomial Optimization*, 166, 1 2012. doi: 10.1007/978-1-4614-0769-0_27.

[6] University of Wisconsin Madison. Neos solver. doi: https://neos-guide.org/.

[7] J.W. Dickey and J.W. Hopkins. Campus building arrangement using topaz. *Transportation Research*, 6:59–68, 1972. doi: https://doi.org/10.1016/0041-1647(72)90111-6.

[8] A.N. Elshafei. Hospital layout as a quadratic assignment problem. *Operational Research Quarterly (1970-1977)*, 28:167–179, 1977. doi: https://www.jstor.org/stable/3008789.

[9] L. Steinberg. The backboard wiring problem: A placement algorithm. *SIAM Review*, 3:37–50, 1961. doi: https://www.jstor.org/stable/2027247.

[10] R. E. Burkard and J. Offermann. Entwurf von schreibmaschinentastaturen mittels quadratischer zuordnungsprobleme, z. *Operations Research*, 21:B121–B132, 1997. doi: https://doi.org/10.1007/BF01918175.

[11] F. Rendl Henry Wolkowicz Q. Zhao, S. E. Karisch. Semidefinite programming relaxations for the quadratic assignment problem. *Journal of Combinatorial Optimization*, 2:71–109, 3 1997. doi: https://doi.org/10.1023/A:1009795911987.

[12] B. Borchers. Sdplib 1.2, a library of semidefinite programming test problems. *11*, 1:683–690, 1999. doi: https://doi.org/10.1080/10556789908805769.