The Semidefinite Programming - Fundamentals

K. Subramani[1]

[1] Lane Department of Computer Science and Electrical Engineering
West Virginia University

April 14, 2014

Outline

Outline

Outline

The Linear Programming Problem

The Linear Programming Problem

Feasibility version (Equality Form)

$$\begin{aligned} \mathbf{A} \cdot \mathbf{x} &= \mathbf{b} \\ \mathbf{x} &\geq \mathbf{0} \end{aligned}$$

The Linear Programming Problem

Feasibility version (Equality Form)

$$\begin{aligned} \mathbf{A} \cdot \mathbf{x} &= \mathbf{b} \\ \mathbf{x} &\geq \mathbf{0} \end{aligned}$$

Algebraic Interpretation

Can **b** be expressed as a *positive* linear combination of the columns of **A**?

The Linear Programming Problem

Feasibility version (Equality Form)

$$\begin{aligned} \mathbf{A} \cdot \mathbf{x} &= \mathbf{b} \\ \mathbf{x} &\geq \mathbf{0} \end{aligned}$$

Algebraic Interpretation

Can **b** be expressed as a *positive* linear combination of the columns of **A**?

Geometric Interpretation

Does **b** fall in the cone determined by the columns of **A**?

The Linear Programming Problem (contd.)

The Linear Programming Problem (contd.)

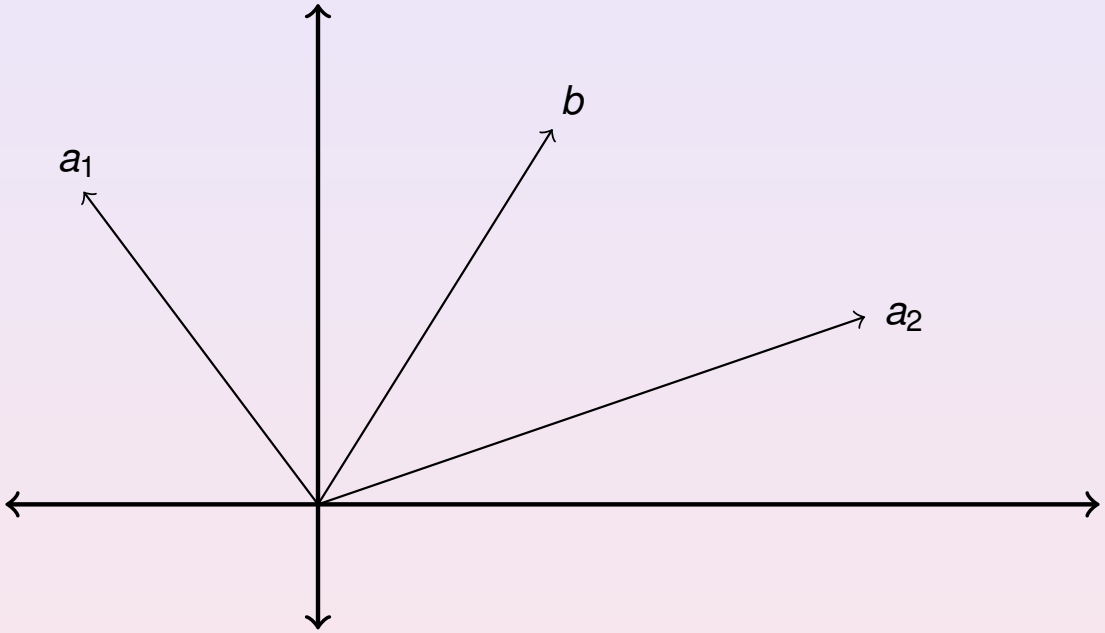

Figure: Geometric Interpretation of Linear Programming

The Linear Programming Problem (contd.)

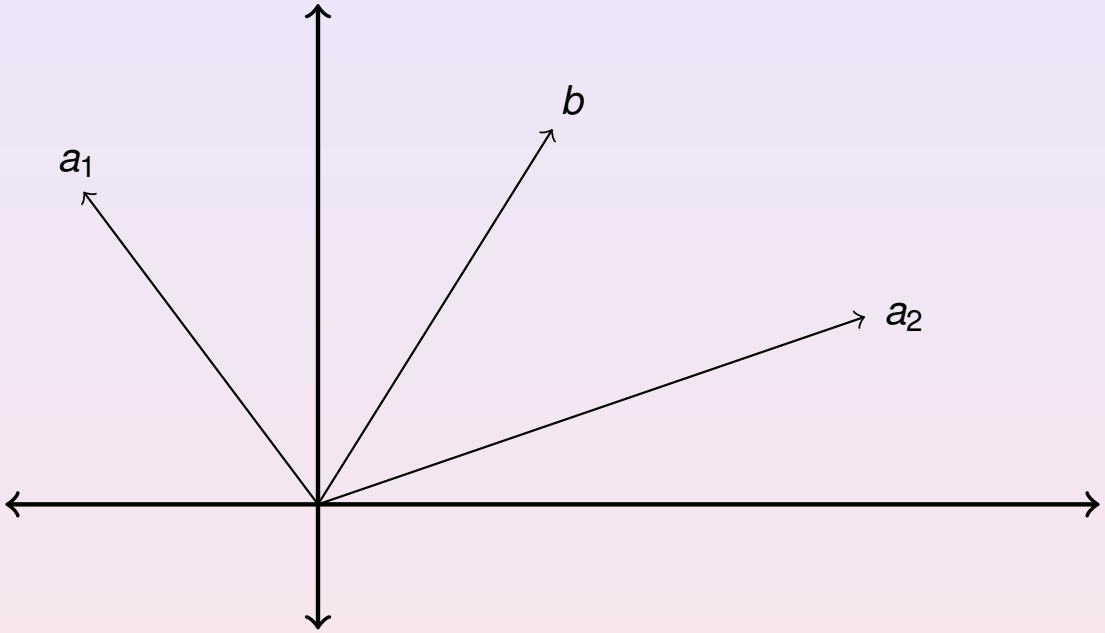

Figure: Geometric Interpretation of Linear Programming

Fact

Linear programming is a one person game.

The Fundamental Duality Theorem

The Fundamental Duality Theorem

Farkas' Lemma

Either $\exists\, \mathbf{x} \in \mathbb{R}^n_+\ \mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ or (mutually exclusively)
$\exists\, \mathbf{y} \in \mathbb{R}^m\ \mathbf{y} \cdot \mathbf{A} \geq \mathbf{0}$ and $\mathbf{y} \cdot \mathbf{b} < 0$.

The Fundamental Duality Theorem

Farkas' Lemma

Either $\exists\, \mathbf{x} \in \mathbb{R}^n_+\ \mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ or (mutually exclusively)
$\exists\, \mathbf{y} \in \mathbb{R}^m\ \mathbf{y} \cdot \mathbf{A} \geq \mathbf{0}$ and $\mathbf{y} \cdot \mathbf{b} < 0$.

Certifying algorithm

A certifying algorithm can either produce $\mathbf{x} \in \mathbb{R}^n_+$, such that $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$,

The Fundamental Duality Theorem

Farkas' Lemma

Either $\exists\, \mathbf{x} \in \mathbb{R}^n_+$ $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ or (mutually exclusively)
$\exists\, \mathbf{y} \in \mathbb{R}^m$ $\mathbf{y} \cdot \mathbf{A} \geq \mathbf{0}$ and $\mathbf{y} \cdot \mathbf{b} < 0$.

Certifying algorithm

A certifying algorithm can either produce $\mathbf{x} \in \mathbb{R}^n_+$, such that $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, or $\mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{y} \cdot \mathbf{A} \geq \mathbf{0}$ and $\mathbf{y} \cdot \mathbf{b} < 0$.

The Fundamental Duality Theorem

Farkas' Lemma

Either $\exists\, \mathbf{x} \in \mathbb{R}^n_+$ $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ or (mutually exclusively)
$\exists\, \mathbf{y} \in \mathbb{R}^m$ $\mathbf{y} \cdot \mathbf{A} \geq \mathbf{0}$ and $\mathbf{y} \cdot \mathbf{b} < 0$.

Certifying algorithm

A certifying algorithm can either produce $\mathbf{x} \in \mathbb{R}^n_+$, such that $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, or $\mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{y} \cdot \mathbf{A} \geq \mathbf{0}$ and $\mathbf{y} \cdot \mathbf{b} < 0$.

Geometric Interpretation

Either $\mathbf{b}$ falls in the cone determined by the columns of $\mathbf{A}$,

The Fundamental Duality Theorem

Farkas' Lemma

Either $\exists\, \mathbf{x} \in \mathbb{R}^n_+$ $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ or (mutually exclusively)
$\exists\, \mathbf{y} \in \mathbb{R}^m$ $\mathbf{y} \cdot \mathbf{A} \geq \mathbf{0}$ and $\mathbf{y} \cdot \mathbf{b} < 0$.

Certifying algorithm

A certifying algorithm can either produce $\mathbf{x} \in \mathbb{R}^n_+$, such that $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, or $\mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{y} \cdot \mathbf{A} \geq \mathbf{0}$ and $\mathbf{y} \cdot \mathbf{b} < 0$.

Geometric Interpretation

Either **b** falls in the cone determined by the columns of **A**, or there exists a **y** that forms an acute angle with each of the columns of **A** while simultaneously forming an obtuse angle with **b**.

The Fundamental Duality Theorem

Farkas' Lemma

Either $\exists\, \mathbf{x} \in \mathbb{R}^n_+$ $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ or (mutually exclusively)
$\exists\, \mathbf{y} \in \mathbb{R}^m$ $\mathbf{y} \cdot \mathbf{A} \geq \mathbf{0}$ and $\mathbf{y} \cdot \mathbf{b} < 0$.

Certifying algorithm

A certifying algorithm can either produce $\mathbf{x} \in \mathbb{R}^n_+$, such that $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, or $\mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{y} \cdot \mathbf{A} \geq \mathbf{0}$ and $\mathbf{y} \cdot \mathbf{b} < 0$.

Geometric Interpretation

Either **b** falls in the cone determined by the columns of **A**, or there exists a **y** that forms an acute angle with each of the columns of **A** while simultaneously forming an obtuse angle with **b**.

Note that in the second case the hyperplane perpendicular to **y** separates **b** from the columns of **A**.

The Fundamental Duality Theorem (contd.)

The Fundamental Duality Theorem (contd.)

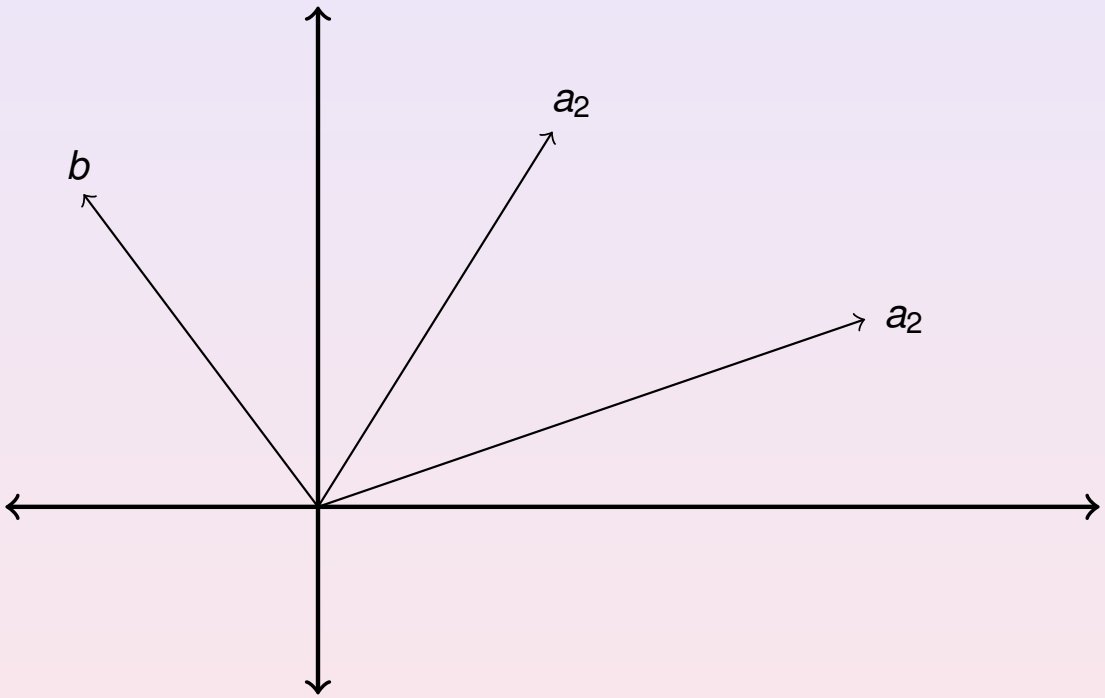

Figure: Geometric Interpretation of Farkas' Lemma

The Fundamental Duality Theorem (contd.)

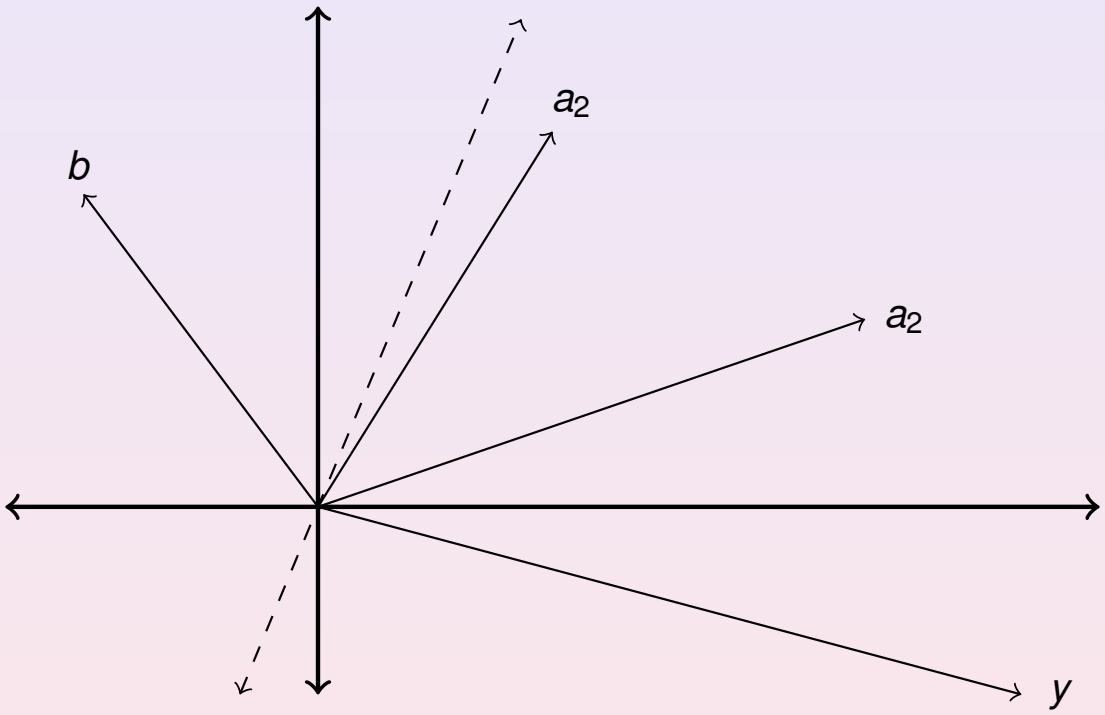

Figure: Geometric Interpretation of Farkas' Lemma

Algebraic Extensions of Linear Programming

Additional one person games

Algebraic Extensions of Linear Programming

Additional one person games

(a) Bilinear programming -

$$\begin{aligned} \min\ z \quad &= \quad \sum_{i=1}^{n} c_i \cdot (x_i \cdot y_i) \\ \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}, \quad & \mathbf{C} \cdot \mathbf{y} \leq \mathbf{d} \end{aligned}$$

Algebraic Extensions of Linear Programming

Additional one person games

(a) Bilinear programming -

$$\begin{aligned} \min\ z &= \sum_{i=1}^{n} c_i \cdot (x_i \cdot y_i) \\ \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}, &\quad \mathbf{C} \cdot \mathbf{y} \leq \mathbf{d} \end{aligned}$$

(b) Quadratic programming -

$$\begin{aligned} \min\ z &= \sum_{i=1}^{n} c_i \cdot (x_i)^2 + \sum_{i=1}^{n} d_i \cdot x_i \\ \mathbf{A} \cdot \mathbf{x} &\leq \mathbf{b} \end{aligned}$$

Algebraic Extensions of Linear Programming

Additional one person games

(a) Bilinear programming -

$$\begin{aligned} \min\ z &= \sum_{i=1}^{n} c_i \cdot (x_i \cdot y_i) \\ \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}, &\quad \mathbf{C} \cdot \mathbf{y} \leq \mathbf{d} \end{aligned}$$

(b) Quadratic programming -

$$\begin{aligned} \min\ z &= \sum_{i=1}^{n} c_i \cdot (x_i)^2 + \sum_{i=1}^{n} d_i \cdot x_i \\ \mathbf{A} \cdot \mathbf{x} &\leq \mathbf{b} \end{aligned}$$

(c) Polynomial programming - Replace the quadratic function with a polynomial function.

Complexity of Quadratic Programming

Complexity of Quadratic Programming

Theorem

Quadratic Programming is **NP-hard**.

Complexity of Quadratic Programming

Theorem

Quadratic Programming is **NP-hard**.

Proof.

Complexity of Quadratic Programming

Theorem

Quadratic Programming is **NP-hard**.

Proof.

1. Let $\phi = \phi_1 \wedge \phi_2 \ldots \phi_m$ denote a 3SAT formula over the n variables $\{x_1, x_2, \ldots x_n\}$.

Complexity of Quadratic Programming

Theorem

Quadratic Programming is **NP-hard**.

Proof.

1. Let $\phi = \phi_1 \wedge \phi_2 \ldots \phi_m$ denote a 3SAT formula over the n variables $\{x_1, x_2, \ldots x_n\}$.
2. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j.

Complexity of Quadratic Programming

Theorem

Quadratic Programming is **NP-hard**.

Proof.

1. Let $\phi = \phi_1 \wedge \phi_2 \ldots \phi_m$ denote a 3SAT formula over the n variables $\{x_1, x_2, \ldots x_n\}$.
2. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.

Complexity of Quadratic Programming

Theorem

Quadratic Programming is **NP-hard**.

Proof.

1. Let $\phi = \phi_1 \wedge \phi_2 \ldots \phi_m$ denote a 3SAT formula over the n variables $\{x_1, x_2, \ldots x_n\}$.
2. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
3. Consider the following quadratic program:

Complexity of Quadratic Programming

Theorem

Quadratic Programming is **NP-hard**.

Proof.

1. Let $\phi = \phi_1 \wedge \phi_2 \ldots \phi_m$ denote a 3SAT formula over the n variables $\{x_1, x_2, \ldots x_n\}$.
2. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
3. Consider the following quadratic program:

$$z = \min \sum_{i=1}^{n} x_i \cdot (1 - x_i)$$

Complexity of Quadratic Programming

Theorem

Quadratic Programming is **NP-hard**.

Proof.

1. Let $\phi = \phi_1 \wedge \phi_2 \ldots \phi_m$ denote a 3SAT formula over the n variables $\{x_1, x_2, \ldots x_n\}$.
2. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
3. Consider the following quadratic program:

$$z = \min \sum_{i=1}^{n} x_i \cdot (1 - x_i)$$

subject to

Complexity of Quadratic Programming

Theorem

Quadratic Programming is **NP-hard**.

Proof.

1. Let $\phi = \phi_1 \wedge \phi_2 \ldots \phi_m$ denote a 3SAT formula over the n variables $\{x_1, x_2, \ldots x_n\}$.
2. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
3. Consider the following quadratic program:

$$\begin{aligned} z = \ & \min \textstyle\sum_{i=1}^{n} x_i \cdot (1 - x_i) \\ \text{subject to} \quad & \textstyle\sum_{i \in C_j^+} x_i + \sum_{i \in C_j^-} (1 - x_i) \geq 1, \quad \forall C_j \end{aligned}$$

Complexity of Quadratic Programming

Theorem

Quadratic Programming is **NP-hard**.

Proof.

1. Let $\phi = \phi_1 \wedge \phi_2 \ldots \phi_m$ denote a 3SAT formula over the n variables $\{x_1, x_2, \ldots x_n\}$.
2. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
3. Consider the following quadratic program:

$$\begin{aligned} z = \ & \min \textstyle\sum_{i=1}^{n} x_i \cdot (1 - x_i) \\ \text{subject to} \quad & \textstyle\sum_{i \in C_j^+} x_i + \sum_{i \in C_j^-} (1 - x_i) \geq 1, && \forall C_j \\ & 0 \leq x_i \leq 1, && \forall i \end{aligned}$$

Complexity of Quadratic Programming

Theorem

Quadratic Programming is **NP-hard**.

Proof.

1. Let $\phi = \phi_1 \wedge \phi_2 \ldots \phi_m$ denote a 3SAT formula over the n variables $\{x_1, x_2, \ldots x_n\}$.
2. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
3. Consider the following quadratic program:

$$\begin{aligned} z = \ & \min \textstyle\sum_{i=1}^{n} x_i \cdot (1 - x_i) \\ \text{subject to} \quad & \textstyle\sum_{i \in C_j^+} x_i + \sum_{i \in C_j^-} (1 - x_i) \geq 1, && \forall C_j \\ & 0 \leq x_i \leq 1, && \forall i \end{aligned}$$

4. It is not hard to see that ϕ is satisfiable if and only if z is zero.

□

Quadratically Constrained Programs

Quadratically Constrained Programs

Quadratically Constrained Programs

Quadratically Constrained Programs

Quadratically Constrained Programs

A quadratically constrained program (QCP) is the problem of optimizing (minimizing or maximizing) a quadratic function of integer valued variables, subject to quadratic constraints on these variables.

If each monomial in the objective function, as well as in each of the constraints, is of degree 0 (i.e., is a constant) or 2, then we will say that this is a strict QCP.

Quadratically Constrained Programs

Quadratically Constrained Programs

A quadratically constrained program (QCP) is the problem of optimizing (minimizing or maximizing) a quadratic function of integer valued variables, subject to quadratic constraints on these variables.

If each monomial in the objective function, as well as in each of the constraints, is of degree 0 (i.e., is a constant) or 2, then we will say that this is a strict QCP. If the program variables are discrete, the program is said to be a quadratically constrained integer program.

Quadratically Constrained Programs

Quadratically Constrained Programs

A quadratically constrained program (QCP) is the problem of optimizing (minimizing or maximizing) a quadratic function of integer valued variables, subject to quadratic constraints on these variables.

If each monomial in the objective function, as well as in each of the constraints, is of degree 0 (i.e., is a constant) or 2, then we will say that this is a strict QCP. If the program variables are discrete, the program is said to be a quadratically constrained integer program.

Example

Quadratically Constrained Programs

Quadratically Constrained Programs

A quadratically constrained program (QCP) is the problem of optimizing (minimizing or maximizing) a quadratic function of integer valued variables, subject to quadratic constraints on these variables.

If each monomial in the objective function, as well as in each of the constraints, is of degree 0 (i.e., is a constant) or 2, then we will say that this is a strict QCP. If the program variables are discrete, the program is said to be a quadratically constrained integer program.

Example

$$\max \sum_{1 \leq i < j \leq n} y_i \cdot y_j$$

Quadratically Constrained Programs

Quadratically Constrained Programs

A quadratically constrained program (QCP) is the problem of optimizing (minimizing or maximizing) a quadratic function of integer valued variables, subject to quadratic constraints on these variables.

If each monomial in the objective function, as well as in each of the constraints, is of degree 0 (i.e., is a constant) or 2, then we will say that this is a strict QCP. If the program variables are discrete, the program is said to be a quadratically constrained integer program.

Example

$$\begin{array}{lll} & \max \sum_{1 \leq i < j \leq n} y_i \cdot y_j & \\ \text{subject to} & y_i \cdot y_j = 1, & \forall y_i, y_j \end{array}$$

Quadratically Constrained Programs

Quadratically Constrained Programs

A quadratically constrained program (QCP) is the problem of optimizing (minimizing or maximizing) a quadratic function of integer valued variables, subject to quadratic constraints on these variables.

If each monomial in the objective function, as well as in each of the constraints, is of degree 0 (i.e., is a constant) or 2, then we will say that this is a strict QCP. If the program variables are discrete, the program is said to be a quadratically constrained integer program.

Example

$$\begin{array}{lll} & \max \sum_{1 \leq i < j \leq n} y_i \cdot y_j & \\ \text{subject to} & y_i \cdot y_j = 1, & \forall y_i, y_j \\ & y_i \in \mathbf{Z}, & \forall y_i \end{array}$$

Vector Programs

Vector Programs

Definition

A vector program is defined over n vector variables in $\Re^n$, say $\mathbf{v_1}, \ldots, \mathbf{v_n}$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, $1 \leq i \leq j \leq n$, subject to linear constraints on these inner products.

Vector Programs

Definition

A vector program is defined over n vector variables in $\Re^n$, say $\mathbf{v_1}, \ldots, \mathbf{v_n}$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, $1 \leq i \leq j \leq n$, subject to linear constraints on these inner products.

Note

Vector Programs

Definition

A vector program is defined over n vector variables in $\Re^n$, say $\mathbf{v_1}, \ldots, \mathbf{v_n}$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, $1 \leq i \leq j \leq n$, subject to linear constraints on these inner products.

Note

1. *A vector program can be thought of as being obtained from a linear program by replacing each variable with an inner product of a pair of these vectors.*

Vector Programs

Definition

A vector program is defined over n vector variables in $\Re^n$, say $\mathbf{v_1}, \ldots, \mathbf{v_n}$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, $1 \leq i \leq j \leq n$, subject to linear constraints on these inner products.

Note

1. *A vector program can be thought of as being obtained from a linear program by replacing each variable with an inner product of a pair of these vectors.*
2. *A strict QCP over n integer variables, defines a vector program over n vector variables in $\Re^n$ as follows:*

Vector Programs

Definition

A vector program is defined over n vector variables in $\Re^n$, say $\mathbf{v_1}, \ldots, \mathbf{v_n}$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, $1 \leq i \leq j \leq n$, subject to linear constraints on these inner products.

Note

1. *A vector program can be thought of as being obtained from a linear program by replacing each variable with an inner product of a pair of these vectors.*
2. *A strict QCP over n integer variables, defines a vector program over n vector variables in $\Re^n$ as follows:*
   Establish a correspondence between the n integer variables and the n vector variables and replace each degree 2 term with the corresponding inner product.

Vector Programs

Definition

A vector program is defined over n vector variables in $\Re^n$, say $\mathbf{v_1}, \ldots, \mathbf{v_n}$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, $1 \leq i \leq j \leq n$, subject to linear constraints on these inner products.

Note

1. *A vector program can be thought of as being obtained from a linear program by replacing each variable with an inner product of a pair of these vectors.*
2. *A strict QCP over n integer variables, defines a vector program over n vector variables in $\Re^n$ as follows:*
   Establish a correspondence between the n integer variables and the n vector variables and replace each degree 2 term with the corresponding inner product.
3. *For the strict QCP example above, the vector program is:*

Vector Programs

Definition

A vector program is defined over n vector variables in $\Re^n$, say $\mathbf{v_1}, \ldots, \mathbf{v_n}$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, $1 \leq i \leq j \leq n$, subject to linear constraints on these inner products.

Note

1. *A vector program can be thought of as being obtained from a linear program by replacing each variable with an inner product of a pair of these vectors.*
2. *A strict QCP over n integer variables, defines a vector program over n vector variables in $\Re^n$ as follows:*
   Establish a correspondence between the n integer variables and the n vector variables and replace each degree 2 term with the corresponding inner product.
3. *For the strict QCP example above, the vector program is:*

Vector Programs

Definition

A vector program is defined over n vector variables in $\Re^n$, say $\mathbf{v_1}, \ldots, \mathbf{v_n}$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, $1 \leq i \leq j \leq n$, subject to linear constraints on these inner products.

Note

1. *A vector program can be thought of as being obtained from a linear program by replacing each variable with an inner product of a pair of these vectors.*
2. *A strict QCP over n integer variables, defines a vector program over n vector variables in $\Re^n$ as follows:*
   Establish a correspondence between the n integer variables and the n vector variables and replace each degree 2 term with the corresponding inner product.
3. *For the strict QCP example above, the vector program is:*

$$\max \sum_{1 \leq i < j \leq n} \mathbf{v_i} \cdot \mathbf{v_j}$$

Vector Programs

Definition

A vector program is defined over n vector variables in $\Re^n$, say $\mathbf{v_1}, \ldots, \mathbf{v_n}$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, $1 \leq i \leq j \leq n$, subject to linear constraints on these inner products.

Note

1. *A vector program can be thought of as being obtained from a linear program by replacing each variable with an inner product of a pair of these vectors.*
2. *A strict QCP over n integer variables, defines a vector program over n vector variables in $\Re^n$ as follows:*
   Establish a correspondence between the n integer variables and the n vector variables and replace each degree 2 term with the corresponding inner product.
3. *For the strict QCP example above, the vector program is:*

$$\begin{array}{lll} & \max \sum_{1 \leq i < j \leq n} \mathbf{v_i} \cdot \mathbf{v_j} & \\ \textit{subject to} & \mathbf{v_i} \cdot \mathbf{v_j} = 1, & \forall \mathbf{v_i}, \mathbf{v_j} \end{array}$$

Vector Programs

Definition

A vector program is defined over n vector variables in $\Re^n$, say $\mathbf{v_1}, \ldots, \mathbf{v_n}$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, $1 \leq i \leq j \leq n$, subject to linear constraints on these inner products.

Note

1. *A vector program can be thought of as being obtained from a linear program by replacing each variable with an inner product of a pair of these vectors.*
2. *A strict QCP over n integer variables, defines a vector program over n vector variables in $\Re^n$ as follows:*
   Establish a correspondence between the n integer variables and the n vector variables and replace each degree 2 term with the corresponding inner product.
3. *For the strict QCP example above, the vector program is:*

$$\begin{array}{lll} & \max \sum_{1 \leq i < j \leq n} \mathbf{v_i} \cdot \mathbf{v_j} & \\ \textit{subject to} & \mathbf{v_i} \cdot \mathbf{v_j} = 1, & \forall \mathbf{v_i}, \mathbf{v_j} \\ & \mathbf{v_i} \in \Re^n, & \forall \mathbf{v_i} \end{array}$$

Semidefinite Matrix

Semidefinite Matrix

Definition

An $n \times n$ real, symmetric matrix $\mathbf{A}$ is said to be positive semidefinite, written $\mathbf{A} \succeq 0$, if one of the following three equivalent conditions holds:

Semidefinite Matrix

Definition

An $n \times n$ real, symmetric matrix $\mathbf{A}$ is said to be positive semidefinite, written $\mathbf{A} \succeq 0$, if one of the following three equivalent conditions holds:

Semidefinite Matrix

Definition

An $n \times n$ real, symmetric matrix $\mathbf{A}$ is said to be positive semidefinite, written $\mathbf{A} \succeq 0$, if one of the following three equivalent conditions holds:

1. $(\forall \mathbf{x} \in \Re^n)\ \mathbf{x}^T \cdot \mathbf{A} \cdot \mathbf{x} \geq 0.$

Semidefinite Matrix

Definition

An $n \times n$ real, symmetric matrix $\mathbf{A}$ is said to be positive semidefinite, written $\mathbf{A} \succeq 0$, if one of the following three equivalent conditions holds:

1. $(\forall \mathbf{x} \in \Re^n)\ \mathbf{x}^T \cdot \mathbf{A} \cdot \mathbf{x} \geq 0$.
2. All eigenvalues of $\mathbf{A}$ are non-negative real numbers.

Semidefinite Matrix

Definition

An $n \times n$ real, symmetric matrix $\mathbf{A}$ is said to be positive semidefinite, written $\mathbf{A} \succeq 0$, if one of the following three equivalent conditions holds:

1. $(\forall \mathbf{x} \in \Re^n)\ \mathbf{x}^T \cdot \mathbf{A} \cdot \mathbf{x} \geq 0$.
2. All eigenvalues of $\mathbf{A}$ are non-negative real numbers.
3. There exists an $n \times n$ *real* matrix $\mathbf{W}$, such that $\mathbf{A} = \mathbf{W}^{\mathbf{T}} \cdot \mathbf{W}$.

Semidefinite Matrix

Definition

An $n \times n$ real, symmetric matrix $\mathbf{A}$ is said to be positive semidefinite, written $\mathbf{A} \succeq 0$, if one of the following three equivalent conditions holds:

1. $(\forall \mathbf{x} \in \Re^n)\ \mathbf{x}^T \cdot \mathbf{A} \cdot \mathbf{x} \geq 0$.
2. All eigenvalues of $\mathbf{A}$ are non-negative real numbers.
3. There exists an $n \times n$ *real* matrix $\mathbf{W}$, such that $\mathbf{A} = \mathbf{W}^{\mathbf{T}} \cdot \mathbf{W}$. (Cholesky factorization).

Observations on semidefiniteness

Observations on semidefiniteness

Note

Observations on semidefiniteness

Note

1. *Any real, symmetric matrix* $\mathbf{A}$ *can be decomposed as* $\mathbf{A} = \mathbf{L} \cdot \mathbf{D} \cdot \mathbf{L}^{\mathsf{T}}$*, where* $\mathbf{D}$ *is a diagonal matrix with the diagonal entries denoting the eigenvalues of* $\mathbf{A}$*.*

Observations on semidefiniteness

Note

1. *Any real, symmetric matrix* $\mathbf{A}$ *can be decomposed as* $\mathbf{A} = \mathbf{L} \cdot \mathbf{D} \cdot \mathbf{L}^{\mathsf{T}}$*, where* $\mathbf{D}$ *is a diagonal matrix with the diagonal entries denoting the eigenvalues of* $\mathbf{A}$*. This decomposition can be done in polynomial time.*

Observations on semidefiniteness

Note

1. *Any real, symmetric matrix* $\mathbf{A}$ *can be decomposed as* $\mathbf{A} = \mathbf{L} \cdot \mathbf{D} \cdot \mathbf{L}^{\mathsf{T}}$*, where* $\mathbf{D}$ *is a diagonal matrix with the diagonal entries denoting the eigenvalues of* $\mathbf{A}$*. This decomposition can be done in polynomial time.*
2. $\mathbf{A}$ *is positive semidefinite, if and only if all the entries of* $\mathbf{D}$ *are non-negative.*

Observations on semidefiniteness

Note

1. *Any real, symmetric matrix* $\mathbf{A}$ *can be decomposed as* $\mathbf{A} = \mathbf{L} \cdot \mathbf{D} \cdot \mathbf{L}^{\mathbf{T}}$*, where* $\mathbf{D}$ *is a diagonal matrix with the diagonal entries denoting the eigenvalues of* $\mathbf{A}$*. This decomposition can be done in polynomial time.*
2. $\mathbf{A}$ *is positive semidefinite, if and only if all the entries of* $\mathbf{D}$ *are non-negative.*
3. *The decomposition of* $\mathbf{A}$ *as* $\mathbf{A} = \mathbf{W}^{\mathbf{T}} \cdot \mathbf{W}$ *is not polynomial time, since the entries could be irrational.*

Observations on semidefiniteness

Note

1. *Any real, symmetric matrix* $\mathbf{A}$ *can be decomposed as* $\mathbf{A} = \mathbf{L} \cdot \mathbf{D} \cdot \mathbf{L}^{\mathbf{T}}$*, where* $\mathbf{D}$ *is a diagonal matrix with the diagonal entries denoting the eigenvalues of* $\mathbf{A}$*. This decomposition can be done in polynomial time.*
2. $\mathbf{A}$ *is positive semidefinite, if and only if all the entries of* $\mathbf{D}$ *are non-negative.*
3. *The decomposition of* $\mathbf{A}$ *as* $\mathbf{A} = \mathbf{W}^{\mathbf{T}} \cdot \mathbf{W}$ *is not polynomial time, since the entries could be irrational. However, the decomposition can be approximated to any desired degree by approximating the square roots of the diagonal entries in* $\mathbf{D}$*.*

Observations on semidefiniteness

Note

1. *Any real, symmetric matrix* $\mathbf{A}$ *can be decomposed as* $\mathbf{A} = \mathbf{L} \cdot \mathbf{D} \cdot \mathbf{L}^{\mathbf{T}}$*, where* $\mathbf{D}$ *is a diagonal matrix with the diagonal entries denoting the eigenvalues of* $\mathbf{A}$*. This decomposition can be done in polynomial time.*
2. $\mathbf{A}$ *is positive semidefinite, if and only if all the entries of* $\mathbf{D}$ *are non-negative.*
3. *The decomposition of* $\mathbf{A}$ *as* $\mathbf{A} = \mathbf{W}^{\mathbf{T}} \cdot \mathbf{W}$ *is not polynomial time, since the entries could be irrational. However, the decomposition can be approximated to any desired degree by approximating the square roots of the diagonal entries in* $\mathbf{D}$*.*
4. *We will assume that* $\mathbf{A}$ *can in fact be decomposed as* $\mathbf{A} = \mathbf{W}^{\mathbf{T}} \cdot \mathbf{W}$ *in polynomial time;*

Observations on semidefiniteness

Note

1. *Any real, symmetric matrix* $\mathbf{A}$ *can be decomposed as* $\mathbf{A} = \mathbf{L} \cdot \mathbf{D} \cdot \mathbf{L}^{\mathbf{T}}$*, where* $\mathbf{D}$ *is a diagonal matrix with the diagonal entries denoting the eigenvalues of* $\mathbf{A}$*. This decomposition can be done in polynomial time.*
2. $\mathbf{A}$ *is positive semidefinite, if and only if all the entries of* $\mathbf{D}$ *are non-negative.*
3. *The decomposition of* $\mathbf{A}$ *as* $\mathbf{A} = \mathbf{W}^{\mathbf{T}} \cdot \mathbf{W}$ *is not polynomial time, since the entries could be irrational. However, the decomposition can be approximated to any desired degree by approximating the square roots of the diagonal entries in* $\mathbf{D}$*.*
4. *We will assume that* $\mathbf{A}$ *can in fact be decomposed as* $\mathbf{A} = \mathbf{W}^{\mathbf{T}} \cdot \mathbf{W}$ *in polynomial time; the inaccuracy will be absorbed into the approximation factor.*

Observations on semidefiniteness

Note

1. *Any real, symmetric matrix* $\mathbf{A}$ *can be decomposed as* $\mathbf{A} = \mathbf{L} \cdot \mathbf{D} \cdot \mathbf{L}^{\mathbf{T}}$*, where* $\mathbf{D}$ *is a diagonal matrix with the diagonal entries denoting the eigenvalues of* $\mathbf{A}$*. This decomposition can be done in polynomial time.*
2. $\mathbf{A}$ *is positive semidefinite, if and only if all the entries of* $\mathbf{D}$ *are non-negative.*
3. *The decomposition of* $\mathbf{A}$ *as* $\mathbf{A} = \mathbf{W}^{\mathbf{T}} \cdot \mathbf{W}$ *is not polynomial time, since the entries could be irrational. However, the decomposition can be approximated to any desired degree by approximating the square roots of the diagonal entries in* $\mathbf{D}$*.*
4. *We will assume that* $\mathbf{A}$ *can in fact be decomposed as* $\mathbf{A} = \mathbf{W}^{\mathbf{T}} \cdot \mathbf{W}$ *in polynomial time; the inaccuracy will be absorbed into the approximation factor.*
5. *The sum of two positive semidefinite matrices is also positive semidefinite and so is any convex combination.*

Frobenius Inner Product

Frobenius Inner Product

Definition

Frobenius Inner Product

Definition

Let $\mathbf{C} = \{c_{ij}\}_{k \times l}$ and let $\mathbf{Y} = \{y_{ij}\}_{k \times l}$.

Frobenius Inner Product

Definition

Let $\mathbf{C} = \{c_{ij}\}_{k \times l}$ and let $\mathbf{Y} = \{y_{ij}\}_{k \times l}$.

$$\mathbf{C} \bullet \mathbf{Y} \quad =$$

Frobenius Inner Product

Definition

Let $\mathbf{C} = \{c_{ij}\}_{k \times l}$ and let $\mathbf{Y} = \{y_{ij}\}_{k \times l}$.

$$\mathbf{C} \bullet \mathbf{Y} \quad = \quad c_{11} \cdot y_{11} + c_{12} \cdot y_{12} + \cdots c_{ij} \cdot y_{ij} + \cdot + c_{kl} \cdot y_{kl}$$

Frobenius Inner Product

Definition

Let $\mathbf{C} = \{c_{ij}\}_{k \times l}$ and let $\mathbf{Y} = \{y_{ij}\}_{k \times l}$.

$$\begin{aligned} \mathbf{C} \bullet \mathbf{Y} &= c_{11} \cdot y_{11} + c_{12} \cdot y_{12} + \cdots c_{ij} \cdot y_{ij} + \cdot + c_{kl} \cdot y_{kl} \\ &= \sum_{i=1}^{k} \sum_{j=1}^{l} c_{ij} \cdot y_{ij} \end{aligned}$$

Frobenius Inner Product

Definition

Let $\mathbf{C} = \{c_{ij}\}_{k \times l}$ and let $\mathbf{Y} = \{y_{ij}\}_{k \times l}$.

$$\begin{aligned} \mathbf{C} \bullet \mathbf{Y} &= c_{11} \cdot y_{11} + c_{12} \cdot y_{12} + \cdots c_{ij} \cdot y_{ij} + \cdot + c_{kl} \cdot y_{kl} \\ &= \sum_{i=1}^{k} \sum_{j=1}^{l} c_{ij} \cdot y_{ij} \end{aligned}$$

Note

Frobenius Inner Product

Definition

Let $\mathbf{C} = \{c_{ij}\}_{k \times l}$ and let $\mathbf{Y} = \{y_{ij}\}_{k \times l}$.

$$\begin{aligned} \mathbf{C} \bullet \mathbf{Y} &= c_{11} \cdot y_{11} + c_{12} \cdot y_{12} + \cdots c_{ij} \cdot y_{ij} + \cdot + c_{kl} \cdot y_{kl} \\ &= \sum_{i=1}^{k} \sum_{j=1}^{l} c_{ij} \cdot y_{ij} \end{aligned}$$

Note

1. $\mathbf{C} \bullet \mathbf{Y}$ *is an inner product of* $\mathbf{C}$ *and* $\mathbf{Y}$.

Frobenius Inner Product

Definition

Let $\mathbf{C} = \{c_{ij}\}_{k \times l}$ and let $\mathbf{Y} = \{y_{ij}\}_{k \times l}$.

$$\begin{aligned} \mathbf{C} \bullet \mathbf{Y} &= c_{11} \cdot y_{11} + c_{12} \cdot y_{12} + \cdots c_{ij} \cdot y_{ij} + \cdot + c_{kl} \cdot y_{kl} \\ &= \sum_{i=1}^{k} \sum_{j=1}^{l} c_{ij} \cdot y_{ij} \end{aligned}$$

Note

1. $\mathbf{C} \bullet \mathbf{Y}$ *is an inner product of* $\mathbf{C}$ *and* $\mathbf{Y}$.
2. $\mathbf{C} \bullet \mathbf{Y}$ *is precisely trace*$(\mathbf{C}^{\mathbf{T}} \cdot \mathbf{Y})$.

Frobenius Inner Product

Definition

Let $\mathbf{C} = \{c_{ij}\}_{k \times l}$ and let $\mathbf{Y} = \{y_{ij}\}_{k \times l}$.

$$\begin{aligned} \mathbf{C} \bullet \mathbf{Y} &= c_{11} \cdot y_{11} + c_{12} \cdot y_{12} + \cdots c_{ij} \cdot y_{ij} + \cdot + c_{kl} \cdot y_{kl} \\ &= \sum_{i=1}^{k} \sum_{j=1}^{l} c_{ij} \cdot y_{ij} \end{aligned}$$

Note

1. $\mathbf{C} \bullet \mathbf{Y}$ *is an inner product of* $\mathbf{C}$ *and* $\mathbf{Y}$.
2. $\mathbf{C} \bullet \mathbf{Y}$ *is precisely trace*$(\mathbf{C}^{\mathbf{T}} \cdot \mathbf{Y})$.
3. $\mathbf{C} \bullet \mathbf{Y}$ *is a linear function of variables* y_{ij} *with coefficients* c_{ij}.

Semidefinite Programming

Semidefinite Programming

The Model

Semidefinite Programming

The Model

1. Let $\mathbf{M_n}$ denote the cone of symmetric $n \times n$ matrices.

Semidefinite Programming

The Model

1. Let $\mathbf{M_n}$ denote the cone of symmetric $n \times n$ matrices. If $\mathbf{A} \in \mathbf{M_n}$, we use $\mathbf{A} \succeq \mathbf{0}$ to denote the fact that $\mathbf{A}$ is positive semidefinite.

Semidefinite Programming

The Model

1. Let $\mathbf{M_n}$ denote the cone of symmetric $n \times n$ matrices. If $\mathbf{A} \in \mathbf{M_n}$, we use $\mathbf{A} \succeq \mathbf{0}$ to denote the fact that $\mathbf{A}$ is positive semidefinite.
2. Let $\mathbf{C}, \mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_k} \in \mathbf{M_n}$ and let $d_1, d_2, \ldots d_k \in \Re$.

Semidefinite Programming

The Model

1. Let $\mathbf{M_n}$ denote the cone of symmetric $n \times n$ matrices. If $\mathbf{A} \in \mathbf{M_n}$, we use $\mathbf{A} \succeq \mathbf{0}$ to denote the fact that $\mathbf{A}$ is positive semidefinite.
2. Let $\mathbf{C}, \mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_k} \in \mathbf{M_n}$ and let $d_1, d_2, \ldots d_k \in \Re$. The semidefinite programming problem (denoted by S) is defined as follows:

Semidefinite Programming

The Model

1. Let $\mathbf{M_n}$ denote the cone of symmetric $n \times n$ matrices. If $\mathbf{A} \in \mathbf{M_n}$, we use $\mathbf{A} \succeq \mathbf{0}$ to denote the fact that $\mathbf{A}$ is positive semidefinite.
2. Let $\mathbf{C}, \mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_k} \in \mathbf{M_n}$ and let $d_1, d_2, \ldots d_k \in \Re$. The semidefinite programming problem (denoted by S) is defined as follows:

$$\max \ \mathbf{C} \bullet \mathbf{Y}$$

Semidefinite Programming

The Model

1. Let $\mathbf{M_n}$ denote the cone of symmetric $n \times n$ matrices. If $\mathbf{A} \in \mathbf{M_n}$, we use $\mathbf{A} \succeq \mathbf{0}$ to denote the fact that $\mathbf{A}$ is positive semidefinite.
2. Let $\mathbf{C}, \mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_k} \in \mathbf{M_n}$ and let $d_1, d_2, \ldots d_k \in \Re$. The semidefinite programming problem (denoted by S) is defined as follows:

$$\begin{aligned} &\max \ \mathbf{C} \bullet \mathbf{Y} \\ \text{subject to} \quad &\mathbf{D_i} \bullet \mathbf{Y} = d_i, \quad 1 \leq i \leq k \end{aligned}$$

Semidefinite Programming

The Model

1. Let $\mathbf{M_n}$ denote the cone of symmetric $n \times n$ matrices. If $\mathbf{A} \in \mathbf{M_n}$, we use $\mathbf{A} \succeq \mathbf{0}$ to denote the fact that $\mathbf{A}$ is positive semidefinite.
2. Let $\mathbf{C}, \mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_k} \in \mathbf{M_n}$ and let $d_1, d_2, \ldots d_k \in \Re$. The semidefinite programming problem (denoted by S) is defined as follows:

$$\begin{aligned} & \max \ \mathbf{C} \bullet \mathbf{Y} \\ \text{subject to} \quad & \mathbf{D_i} \bullet \mathbf{Y} = d_i, \quad 1 \leq i \leq k \\ & \mathbf{Y} \succeq \mathbf{0}, \end{aligned}$$

Semidefinite Programming

The Model

1. Let $\mathbf{M_n}$ denote the cone of symmetric $n \times n$ matrices. If $\mathbf{A} \in \mathbf{M_n}$, we use $\mathbf{A} \succeq \mathbf{0}$ to denote the fact that $\mathbf{A}$ is positive semidefinite.
2. Let $\mathbf{C}, \mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_k} \in \mathbf{M_n}$ and let $d_1, d_2, \ldots d_k \in \Re$. The semidefinite programming problem (denoted by S) is defined as follows:

$$\begin{aligned} & \max \ \mathbf{C} \bullet \mathbf{Y} \\ \text{subject to} \quad & \mathbf{D_i} \bullet \mathbf{Y} = d_i, \quad 1 \leq i \leq k \\ & \mathbf{Y} \succeq \mathbf{0}, \\ & \mathbf{Y} \in \mathbf{M_n} \end{aligned}$$

Properties of semidefinite programs

Properties of semidefinite programs

Observation

Properties of semidefinite programs

Observation

1. *Semidefinite programming generalizes linear programming.*

Properties of semidefinite programs

Observation

1. *Semidefinite programming generalizes linear programming. In the semidefinite program S, if the matrices* $\mathbf{C}, \mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_k}$ *are all diagonal, then S becomes a linear programming problem.*

Properties of semidefinite programs

Observation

1. *Semidefinite programming generalizes linear programming. In the semidefinite program S, if the matrices* $\mathbf{C}, \mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_k}$ *are all diagonal, then S becomes a linear programming problem.*
2. *Inequality constraints can be added, without affecting the form of S.*

Properties of semidefinite programs

Observation

1. *Semidefinite programming generalizes linear programming. In the semidefinite program S, if the matrices* $\mathbf{C}, \mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_k}$ *are all diagonal, then S becomes a linear programming problem.*
2. *Inequality constraints can be added, without affecting the form of S.*
3. *The set of feasible solutions of S forms a convex set*

Properties of semidefinite programs

Observation

1. *Semidefinite programming generalizes linear programming. In the semidefinite program S, if the matrices* $\mathbf{C}, \mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_k}$ *are all diagonal, then S becomes a linear programming problem.*
2. *Inequality constraints can be added, without affecting the form of S.*
3. *The set of feasible solutions of S forms a convex set (since the convex combination of semidefinite matrices is semidefinite).*

Properties of semidefinite programs

Observation

1. *Semidefinite programming generalizes linear programming. In the semidefinite program S, if the matrices* $\mathbf{C}, \mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_k}$ *are all diagonal, then S becomes a linear programming problem.*
2. *Inequality constraints can be added, without affecting the form of S.*
3. *The set of feasible solutions of S forms a convex set (since the convex combination of semidefinite matrices is semidefinite).*
4. *Let* $\mathbf{A} \in \Re^{n \times n}$ *be an infeasible point. Let* $\mathbf{C} \in \Re^{n \times n}$.

Properties of semidefinite programs

Observation

1. *Semidefinite programming generalizes linear programming. In the semidefinite program S, if the matrices* $\mathbf{C}, \mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_k}$ *are all diagonal, then S becomes a linear programming problem.*
2. *Inequality constraints can be added, without affecting the form of S.*
3. *The set of feasible solutions of S forms a convex set (since the convex combination of semidefinite matrices is semidefinite).*
4. *Let* $\mathbf{A} \in \Re^{n \times n}$ *be an infeasible point. Let* $\mathbf{C} \in \Re^{n \times n}$*. A hyperplane* $\mathbf{C} \bullet \mathbf{Y} \leq b$ *is called a separating hyperplane for* $\mathbf{A}$*, if all feasible points satisfy it and point* $\mathbf{A}$ *does not satisfy it.*

Properties of semidefinite programs

Observation

1. *Semidefinite programming generalizes linear programming. In the semidefinite program S, if the matrices* $\mathbf{C}, \mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_k}$ *are all diagonal, then S becomes a linear programming problem.*
2. *Inequality constraints can be added, without affecting the form of S.*
3. *The set of feasible solutions of S forms a convex set (since the convex combination of semidefinite matrices is semidefinite).*
4. *Let* $\mathbf{A} \in \Re^{n \times n}$ *be an infeasible point. Let* $\mathbf{C} \in \Re^{n \times n}$*. A hyperplane* $\mathbf{C} \bullet \mathbf{Y} \leq b$ *is called a separating hyperplane for* $\mathbf{A}$*, if all feasible points satisfy it and point* $\mathbf{A}$ *does not satisfy it.*
5. *Most effective for maximization problems involving constraints between* pairs *of variables.*

Properties of semidefinite programs

Observation

1. *Semidefinite programming generalizes linear programming. In the semidefinite program S, if the matrices* $\mathbf{C}, \mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_k}$ *are all diagonal, then S becomes a linear programming problem.*
2. *Inequality constraints can be added, without affecting the form of S.*
3. *The set of feasible solutions of S forms a convex set (since the convex combination of semidefinite matrices is semidefinite).*
4. *Let* $\mathbf{A} \in \Re^{n \times n}$ *be an infeasible point. Let* $\mathbf{C} \in \Re^{n \times n}$*. A hyperplane* $\mathbf{C} \bullet \mathbf{Y} \leq b$ *is called a separating hyperplane for* $\mathbf{A}$*, if all feasible points satisfy it and point* $\mathbf{A}$ *does not satisfy it.*
5. *Most effective for maximization problems involving constraints between* pairs *of variables.*
6. *Leads to reasonably good approximation ratios, although in most cases, not the best possible.*

Separating hyperplane theorem

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ *be a point in* $\Re^{n \times n}$.

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n \times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n \times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n \times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

If $\mathbf{A}$ is feasible, then $\mathbf{A}$ should be

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n \times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

If $\mathbf{A}$ is feasible, then $\mathbf{A}$ should be *symmetric*,

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n\times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

If $\mathbf{A}$ is feasible, then $\mathbf{A}$ should be *symmetric*, *positive semidefinite*,

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n \times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

If $\mathbf{A}$ is feasible, then $\mathbf{A}$ should be *symmetric*, *positive semidefinite*, and *satisfy all the linear constraints.*

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n \times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

If $\mathbf{A}$ is feasible, then $\mathbf{A}$ should be *symmetric*, *positive semidefinite*, and *satisfy all the linear constraints.*
Clearly, the above checks can be carried out in polynomial time.

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n\times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

If $\mathbf{A}$ is feasible, then $\mathbf{A}$ should be *symmetric*, *positive semidefinite*, and *satisfy all the linear constraints.*
Clearly, the above checks can be carried out in polynomial time.

The separating hyperplane can be found as follows:

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n\times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

If $\mathbf{A}$ is feasible, then $\mathbf{A}$ should be *symmetric*, *positive semidefinite*, and *satisfy all the linear constraints.*
Clearly, the above checks can be carried out in polynomial time.

The separating hyperplane can be found as follows:

1. If $\mathbf{A}$ is not symmetric, then $a_{ij} > a_{ji}$, for some i, j.

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n\times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

If $\mathbf{A}$ is feasible, then $\mathbf{A}$ should be *symmetric*, *positive semidefinite*, and *satisfy all the linear constraints.*
Clearly, the above checks can be carried out in polynomial time.

The separating hyperplane can be found as follows:

1. If $\mathbf{A}$ is not symmetric, then $a_{ij} > a_{ji}$, for some i, j. Then, $y_{ij} \leq y_{ji}$ is a separating hyperplane.

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n\times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

If $\mathbf{A}$ is feasible, then $\mathbf{A}$ should be *symmetric*, *positive semidefinite*, and *satisfy all the linear constraints.*
Clearly, the above checks can be carried out in polynomial time.

The separating hyperplane can be found as follows:

1. If $\mathbf{A}$ is not symmetric, then $a_{ij} > a_{ji}$, for some i, j. Then, $y_{ij} \leq y_{ji}$ is a separating hyperplane. (Violation of $\mathbf{Y} \in \mathbf{M_n}$.)

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n\times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

If $\mathbf{A}$ is feasible, then $\mathbf{A}$ should be *symmetric*, *positive semidefinite*, and *satisfy all the linear constraints.*
Clearly, the above checks can be carried out in polynomial time.

The separating hyperplane can be found as follows:

1. If $\mathbf{A}$ is not symmetric, then $a_{ij} > a_{ji}$, for some i, j. Then, $y_{ij} \leq y_{ji}$ is a separating hyperplane. (Violation of $\mathbf{Y} \in \mathbf{M_n}$.)
2. If $\mathbf{A}$ is not positive semidefinite, then it has a negative eigenvalue, say λ.

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n\times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

If $\mathbf{A}$ is feasible, then $\mathbf{A}$ should be *symmetric*, *positive semidefinite*, and *satisfy all the linear constraints.*
Clearly, the above checks can be carried out in polynomial time.

The separating hyperplane can be found as follows:

1. If $\mathbf{A}$ is not symmetric, then $a_{ij} > a_{ji}$, for some i, j. Then, $y_{ij} \leq y_{ji}$ is a separating hyperplane. (Violation of $\mathbf{Y} \in \mathbf{M_n}$.)
2. If $\mathbf{A}$ is not positive semidefinite, then it has a negative eigenvalue, say λ. Let $\mathbf{v}$ be the corresponding eigenvector.

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n\times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

If $\mathbf{A}$ is feasible, then $\mathbf{A}$ should be *symmetric*, *positive semidefinite*, and *satisfy all the linear constraints.*
Clearly, the above checks can be carried out in polynomial time.

The separating hyperplane can be found as follows:

1. If $\mathbf{A}$ is not symmetric, then $a_{ij} > a_{ji}$, for some i, j. Then, $y_{ij} \leq y_{ji}$ is a separating hyperplane. (Violation of $\mathbf{Y} \in \mathbf{M_n}$.)
2. If $\mathbf{A}$ is not positive semidefinite, then it has a negative eigenvalue, say λ. Let $\mathbf{v}$ be the corresponding eigenvector. Now $(\mathbf{v} \cdot \mathbf{v^T}) \bullet \mathbf{Y} = \mathbf{v^T} \cdot \mathbf{Y} \cdot \mathbf{v} \geq 0$ is a separating hyperplane.

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n\times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

If $\mathbf{A}$ is feasible, then $\mathbf{A}$ should be *symmetric*, *positive semidefinite*, and *satisfy all the linear constraints.*
Clearly, the above checks can be carried out in polynomial time.

The separating hyperplane can be found as follows:

1. If $\mathbf{A}$ is not symmetric, then $a_{ij} > a_{ji}$, for some i, j. Then, $y_{ij} \leq y_{ji}$ is a separating hyperplane. (Violation of $\mathbf{Y} \in \mathbf{M_n}$.)
2. If $\mathbf{A}$ is not positive semidefinite, then it has a negative eigenvalue, say λ. Let $\mathbf{v}$ be the corresponding eigenvector. Now $(\mathbf{v}\cdot\mathbf{v^T}) \bullet \mathbf{Y} = \mathbf{v^T}\cdot\mathbf{Y}\cdot\mathbf{v} \geq 0$ is a separating hyperplane. (Violation of $\mathbf{Y} \succeq \mathbf{0}$.)

Separating hyperplane theorem

Theorem

Let S be a semidefinite programming problem, and let $\mathbf{A}$ be a point in $\Re^{n\times n}$. We can determine, in polynomial time, whether $\mathbf{A}$ is feasible for S and, if it is not, find a separating hyperplane.

Proof.

If $\mathbf{A}$ is feasible, then $\mathbf{A}$ should be *symmetric*, *positive semidefinite*, and *satisfy all the linear constraints.*
Clearly, the above checks can be carried out in polynomial time.

The separating hyperplane can be found as follows:

1. If $\mathbf{A}$ is not symmetric, then $a_{ij} > a_{ji}$, for some i, j. Then, $y_{ij} \leq y_{ji}$ is a separating hyperplane. (Violation of $\mathbf{Y} \in \mathbf{M_n}$.)
2. If $\mathbf{A}$ is not positive semidefinite, then it has a negative eigenvalue, say λ. Let $\mathbf{v}$ be the corresponding eigenvector. Now $(\mathbf{v}\cdot\mathbf{v^T}) \bullet \mathbf{Y} = \mathbf{v^T}\cdot\mathbf{Y}\cdot\mathbf{v} \geq 0$ is a separating hyperplane. (Violation of $\mathbf{Y} \succeq \mathbf{0}$.)
3. If any of the linear constraints is violated, it directly yields a separating hyperplane. (Violation of $\mathbf{D_i} \bullet \mathbf{Y} = d_i$, for some $i = 1, 2, \ldots, k$.)

□

Equivalence between vector programs and semidefinite programs

Equivalence between vector programs and semidefinite programs

Lemma

Equivalence between vector programs and semidefinite programs

Lemma

Let V be a vector program on n n-dimensional vector variables $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$.

Equivalence between vector programs and semidefinite programs

Lemma

Let V be a vector program on n n-dimensional vector variables $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$. Define the corresponding semidefinite program, S, over n^2 variables $y_{ij}, 1 \leq i, j \leq n$, as follows:

Equivalence between vector programs and semidefinite programs

Lemma

Let V be a vector program on n n-dimensional vector variables $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$. Define the corresponding semidefinite program, S, over n^2 variables $y_{ij}, 1 \leq i, j \leq n$, as follows:

1. *Replace each inner product $\mathbf{v_i} \cdot \mathbf{v_j}$ occurring in V by the variable y_{ij}.*

Equivalence between vector programs and semidefinite programs

Lemma

Let V be a vector program on n n-dimensional vector variables $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$. Define the corresponding semidefinite program, S, over n^2 variables $y_{ij}, 1 \leq i, j \leq n$, as follows:

1. *Replace each inner product $\mathbf{v_i} \cdot \mathbf{v_j}$ occurring in V by the variable y_{ij}.*
2. *The objective function and constraints are now linear in the y_{ij}'s.*

Equivalence between vector programs and semidefinite programs

Lemma

Let V be a vector program on n n-dimensional vector variables $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$. Define the corresponding semidefinite program, S, over n^2 variables $y_{ij}, 1 \leq i, j \leq n$, as follows:

1. *Replace each inner product $\mathbf{v_i} \cdot \mathbf{v_j}$ occurring in V by the variable y_{ij}.*
2. *The objective function and constraints are now linear in the y_{ij}'s.*
3. *Additionally, require that matrix $\mathbf{Y}$, whose $(i,j)^{th}$ entry is y_{ij}, be symmetric and positive semidefinite.*

Equivalence between vector programs and semidefinite programs

Lemma

Let V be a vector program on n n-dimensional vector variables $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$. Define the corresponding semidefinite program, S, over n^2 variables $y_{ij}, 1 \leq i, j \leq n$, as follows:

1. *Replace each inner product $\mathbf{v_i} \cdot \mathbf{v_j}$ occurring in V by the variable y_{ij}.*
2. *The objective function and constraints are now linear in the y_{ij}'s.*
3. *Additionally, require that matrix $\mathbf{Y}$, whose $(i,j)^{th}$ entry is y_{ij}, be symmetric and positive semidefinite.*

Vector program V is equivalent to semidefinite program S.

Proof of Equivalence

Proof of Equivalence

Proof.

Proof of Equivalence

Proof.

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote a feasible solution to the vector program V.

Proof of Equivalence

Proof.

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote a feasible solution to the vector program V.
2. Let $\mathbf{W}$ be the matrix whose columns are $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$.

Proof of Equivalence

Proof.

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote a feasible solution to the vector program V.
2. Let $\mathbf{W}$ be the matrix whose columns are $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$.
3. $\mathbf{A} = \mathbf{W^T} \cdot \mathbf{W}$ is a feasible solution to the SDP S, having the same objective function value.

Proof of Equivalence

Proof.

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote a feasible solution to the vector program V.
2. Let $\mathbf{W}$ be the matrix whose columns are $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$.
3. $\mathbf{A} = \mathbf{W^T} \cdot \mathbf{W}$ is a feasible solution to the SDP S, having the same objective function value.
4. Let $\mathbf{A}$ be a feasible solution to S.

Proof of Equivalence

Proof.

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote a feasible solution to the vector program V.
2. Let $\mathbf{W}$ be the matrix whose columns are $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$.
3. $\mathbf{A} = \mathbf{W^T} \cdot \mathbf{W}$ is a feasible solution to the SDP S, having the same objective function value.
4. Let $\mathbf{A}$ be a feasible solution to S. Since $\mathbf{A}$ is positive semidefinite, there must exist an $n \times n$ matrix $\mathbf{W}$, such that $\mathbf{A} = \mathbf{W^T} \cdot \mathbf{W}$.

Proof of Equivalence

Proof.

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote a feasible solution to the vector program V.
2. Let $\mathbf{W}$ be the matrix whose columns are $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$.
3. $\mathbf{A} = \mathbf{W^T} \cdot \mathbf{W}$ is a feasible solution to the SDP S, having the same objective function value.
4. Let $\mathbf{A}$ be a feasible solution to S. Since $\mathbf{A}$ is positive semidefinite, there must exist an $n \times n$ matrix $\mathbf{W}$, such that $\mathbf{A} = \mathbf{W^T} \cdot \mathbf{W}$.
5. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the columns of $\mathbf{W}$.

Proof of Equivalence

Proof.

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote a feasible solution to the vector program V.
2. Let $\mathbf{W}$ be the matrix whose columns are $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$.
3. $\mathbf{A} = \mathbf{W^T} \cdot \mathbf{W}$ is a feasible solution to the SDP S, having the same objective function value.
4. Let $\mathbf{A}$ be a feasible solution to S. Since $\mathbf{A}$ is positive semidefinite, there must exist an $n \times n$ matrix $\mathbf{W}$, such that $\mathbf{A} = \mathbf{W^T} \cdot \mathbf{W}$.
5. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the columns of $\mathbf{W}$.
6. These columns form a feasible solution to the vector program V, having the same objective function value.

$\square$