The Semidefinite Programming - Fundamentals

K. Subramani

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

April 14, 2014
Outline

1 Mathematical Programming Frameworks
Outline

1. Mathematical Programming Frameworks
2. Rudiments of Semidefinite Programming
Outline

1. Mathematical Programming Frameworks
2. Rudiments of Semidefinite Programming
3. Separating hyper-planes and vector programs
The Linear Programming Problem

Feasibility version (Equality Form)

\[ A \cdot x = b \]
\[ x \geq 0 \]

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

Feasibility version (Equality Form)

\[ A \cdot x = b \]
\[ x \geq 0 \]
The Linear Programming Problem

Feasibility version (Equality Form)

\[ A \cdot x = b \]
\[ x \geq 0 \]

Algebraic Interpretation

Can \( b \) be expressed as a positive linear combination of the columns of \( A \)?
The Linear Programming Problem

Feasibility version (Equality Form)

\[ \mathbf{A} \cdot \mathbf{x} = \mathbf{b} \]
\[ \mathbf{x} \geq 0 \]

Algebraic Interpretation
Can \( \mathbf{b} \) be expressed as a positive linear combination of the columns of \( \mathbf{A} \)?

Geometric Interpretation
Does \( \mathbf{b} \) fall in the cone determined by the columns of \( \mathbf{A} \)?
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

Farkas' Lemma

Either

\[ \exists x \in \mathbb{R}^n : A \cdot x = b \]

or (mutually exclusively)

\[ \exists y \in \mathbb{R}^m : y \cdot A \geq 0 \text{ and } y \cdot b < 0. \]

Certifying algorithm

A certifying algorithm can either produce

\[ x \in \mathbb{R}^n \text{ such that } A \cdot x = b, \]

or

\[ y \in \mathbb{R}^m \text{ such that } y \cdot A \geq 0 \text{ and } y \cdot b < 0. \]

Geometric Interpretation

Either

\[ b \text{ falls in the cone determined by the columns of } A, \]

or there exists a

\[ y \text{ that forms an acute angle with each of the columns of } A \text{ while simultaneously forming an obtuse angle with } b. \]

Note that in the second case the hyperplane perpendicular to

\[ y \]

separates

\[ b \]

from the columns of

\[ A. \]
The Fundamental Duality Theorem

Farkas’ Lemma

Either \( \exists \ x \in \mathbb{R}^n \ A \cdot x = b \) or (mutually exclusively) \( \exists \ y \in \mathbb{R}^m \ y \cdot A \geq 0 \) and \( y \cdot b < 0 \).
The Fundamental Duality Theorem

**Farkas’ Lemma**

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

**Certifying algorithm**

A certifying algorithm can either produce \( x \in \mathbb{R}^n_+ \), such that \( A \cdot x = b \),
The Fundamental Duality Theorem

Farkas’ Lemma

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

Certifying algorithm

A certifying algorithm can either produce \( x \in \mathbb{R}_+^n \), such that \( A \cdot x = b \), or \( y \in \mathbb{R}^m \) such that \( y \cdot A \geq 0 \) and \( y \cdot b < 0 \).
The Fundamental Duality Theorem

Farkas’ Lemma

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

Certifying algorithm

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

Geometric Interpretation

Either \( b \) falls in the cone determined by the columns of \( A \),
The Fundamental Duality Theorem

**Farkas’ Lemma**

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

**Certifying algorithm**

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

**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 \( \exists \ x \in \mathbb{R}_+^n \ A \cdot x = b \) or (mutually exclusively) \( \exists \ y \in \mathbb{R}_+^m y \cdot A \geq 0 \) and \( y \cdot b < 0 \).

Certifying algorithm

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

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
Additional one person games

(a) Bilinear programming -
\[ \min z = \sum_{i=1}^{n} c_i \cdot (x_i \cdot y_i) \]
\[ A \cdot x \leq b, C \cdot y \leq d \]

(b) Quadratic programming -
\[ \min z = \sum_{i=1}^{n} c_i \cdot (x_i)^2 + \sum_{i=1}^{n} d_i \cdot x_i \]
\[ A \cdot x \leq b \]

(c) Polynomial programming - Replace the quadratic function with a polynomial function.
Additional one person games

(a) Bilinear programming -

\[
\begin{align*}
\min z &= \sum_{i=1}^{n} c_i \cdot (x_i \cdot y_i) \\
A \cdot x &\leq b, \quad C \cdot y \leq d
\end{align*}
\]
Additional one person games

(a) Bilinear programming -

\[
\min z = \sum_{i=1}^{n} c_i \cdot (x_i \cdot y_i) \\
A \cdot x \leq b, \quad C \cdot y \leq d
\]

(b) Quadratic programming -

\[
\min z = \sum_{i=1}^{n} c_i \cdot (x_i)^2 + \sum_{i=1}^{n} d_i \cdot x_i \\
A \cdot x \leq b
\]
Additional one person games

(a) Bilinear programming -

\[ \min z = \sum_{i=1}^{n} c_i \cdot (x_i \cdot y_i) \]
\[ A \cdot x \leq b, \quad C \cdot y \leq d \]

(b) Quadratic programming -

\[ \min z = \sum_{i=1}^{n} c_i \cdot (x_i)^2 + \sum_{i=1}^{n} d_i \cdot x_i \]
\[ A \cdot x \leq b \]

(c) Polynomial programming - Replace the quadratic function with a polynomial function.
Theorem
Quadratic Programming is NP-hard.

Proof.
1. Let $\phi = \phi_1 \land \phi_2 \ldots \phi_m$ 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
   
   $\sum_{i \in C^+_j} x_i + \sum_{i \in C^-_j} (1 - x_i) \geq 1, \quad \forall C_j$
   
   $0 \leq x_i \leq 1, \quad \forall i$

4. It is not hard to see that $\phi$ is satisfiable if and only if $z$ is zero.
Complexity of Quadratic Programming

Theorem

*Quadratic Programming is NP-hard.*
Complexity of Quadratic Programming

Theorem

*Quadratic Programming is NP-hard.*

Proof.

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

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 $C_j$.

Consider the following quadratic program:

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

subject to

$$\sum_{i \in C_j^+} x_i + \sum_{i \in C_j^-} (1 - x_i) \geq 1, \forall C_j$$

$$0 \leq x_i \leq 1, \forall i$$

It is not hard to see that $\phi$ is satisfiable if and only if $z$ is zero.
Theorem

*Quadratic Programming is NP-hard.*

Proof.

Let \( \phi = \phi_1 \land \phi_2 \ldots \land \phi_m \) 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 $\phi = \phi_1 \land \phi_2 \ldots \phi_m$ 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 $C_j$. 
Theorem

*Quadratic Programming is NP-hard.*

Proof.

1. Let \( \phi = \phi_1 \land \phi_2 \ldots \land \phi_m \) 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 $\phi = \phi_1 \land \phi_2 \ldots \phi_m$ 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 \( \phi = \phi_1 \land \phi_2 \ldots \land \phi_m \) 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 $\phi = \phi_1 \land \phi_2 \ldots \phi_m$ 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 $\phi = \phi_1 \land \phi_2 \ldots \phi_m$ 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 $\sum_{i \in C_j^+} x_i + \sum_{i \in C_j^-} (1 - x_i) \geq 1, \quad \forall C_j$
Complexity of Quadratic Programming

**Theorem**

*Quadratic Programming is NP-hard.*

**Proof.**

1. Let $\phi = \phi_1 \land \phi_2 \ldots \phi_m$ 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

$$\sum_{i \in C_j^+} x_i + \sum_{i \in C_j^-} (1 - x_i) \geq 1, \quad \forall C_j$$

$$0 \leq x_i \leq 1, \quad \forall i$$
Complexity of Quadratic Programming

**Theorem**

*Quadratic Programming is NP-hard.*

**Proof.**

1. Let $\phi = \phi_1 \land \phi_2 \ldots \phi_m$ 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

   $\sum_{i \in C_j^+} x_i + \sum_{i \in C_j^-} (1 - x_i) \geq 1, \quad \forall C_j$

   $0 \leq x_i \leq 1, \quad \forall i$

4. It is not hard to see that $\phi$ is satisfiable if and only if $z$ is zero.
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:

$$\max \sum_{1 \leq i < j \leq n} y_i \cdot y_j$$
subject to
$$y_i \cdot y_j = 1, \forall y_i$$
$$y_i \in \mathbb{Z}, \forall y_i$$
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

\[
\begin{align*}
\text{max} & \quad \sum_{1 \leq i < j \leq n} y_i \cdot y_j \\
\text{subject to} & \quad y_i \cdot y_j = 1, \quad \forall y_i, y_j \\
& \quad y_i, y_j \in \mathbb{Z}, \quad \forall y_i
\end{align*}
\]
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.
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

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

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

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

\[
\begin{align*}
\text{max } & \sum_{1 \leq i < j \leq n} y_i \cdot y_j \\
\text{subject to } & y_i \cdot y_j = 1, \quad \forall y_i, y_j
\end{align*}
\]
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

\[
\max \sum_{1 \leq i < j \leq n} y_i \cdot y_j \\
\text{subject to} \quad y_i \cdot y_j = 1, \quad \forall y_i, y_j \\
y_i \in \mathbb{Z}, \quad \forall y_i
\]
Vector Programs

Definition
A vector program is defined over $n$ vector variables in $\mathbb{R}^n$, say $v_1, \ldots, v_n$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $v_i \cdot v_j$, $1 \leq i \leq j \leq n$, 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 $\mathbb{R}^n$ 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:

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

subject to

$$v_i \cdot v_j = 1, \forall v_i, v_j$$

$$v_i \in \mathbb{R}^n, \forall v_i$$
**Definition**

A vector program is defined over \( n \) vector variables in \( \mathbb{R}^n \), 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 \( \mathbb{R}^n \), 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 \( \mathbb{R}^n \) 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:

\[
\begin{align*}
\text{max} & \quad \sum_{1 \leq i < j \leq n} \mathbf{v}_i \cdot \mathbf{v}_j \\
\text{subject to} & \quad \mathbf{v}_i \cdot \mathbf{v}_j = 1, \quad \forall \mathbf{v}_i, \mathbf{v}_j \\
& \quad \mathbf{v}_i \in \mathbb{R}^n, \quad \forall \mathbf{v}_i
\end{align*}
\]
## Vector Programs

### Definition

A vector program is defined over \( n \) vector variables in \( \mathbb{R}^n \), 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 $\mathbb{R}^n$, say $v_1, \ldots, v_n$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $v_i \cdot v_j$, $1 \leq i \leq j \leq n$, 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 $\mathbb{R}^n$ as follows:
Vector Programs

Definition

A vector program is defined over $n$ vector variables in $\mathbb{R}^n$, say $v_1, \ldots, v_n$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $v_i \cdot v_j$, $1 \leq i \leq j \leq n$, 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 $\mathbb{R}^n$ 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 $\mathbb{R}^n$, say $v_1, \ldots, v_n$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $v_i \cdot v_j$, $1 \leq i \leq j \leq n$, 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 $\mathbb{R}^n$ 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 $\mathbb{R}^n$, 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 $\mathbb{R}^n$ 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 \( \mathbb{R}^n \), say \( v_1, \ldots, v_n \), and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products \( v_i \cdot v_j, 1 \leq i \leq j \leq n \), 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 \( \mathbb{R}^n \) 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:

   \[
   \max \sum_{1 \leq i < j \leq n} v_i \cdot v_j
   \]
Vector Programs

**Definition**

A vector program is defined over \( n \) vector variables in \( \mathbb{R}^n \), 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 \( \mathbb{R}^n \) 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:

\[
\max \sum_{1 \leq i < j \leq n} \mathbf{v}_i \cdot \mathbf{v}_j \\
\text{subject to} \quad \mathbf{v}_i \cdot \mathbf{v}_j = 1, \quad \forall \mathbf{v}_i, \mathbf{v}_j
\]
Vector Programs

Definition

A vector program is defined over $n$ vector variables in $\mathbb{R}^n$, say $v_1, \ldots, v_n$, and is the problem of optimizing (minimizing or maximizing) a linear function of the inner products $v_i \cdot v_j$, $1 \leq i \leq j \leq n$, 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 $\mathbb{R}^n$ 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:

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

subject to

$$v_i \cdot v_j = 1, \quad \forall v_i, v_j$$

$$v_i \in \mathbb{R}^n, \quad \forall v_i$$
Semidefinite Programming

Rudiments of Semidefinite Programming

Semidefinite Matrix

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

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

**Definition**

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

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

Definition

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

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

2. All eigenvalues of $A$ are non-negative real numbers.

3. There exists an $n \times n$ real matrix $W$, such that $A = W^T \cdot W$ (Cholesky factorization).
Semidefinite Matrix

**Definition**

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

1. $(\forall x \in \mathbb{R}^n) \ x^T \cdot A \cdot x \geq 0$.
2. All eigenvalues of $A$ are non-negative real numbers.
An $n \times n$ real, symmetric matrix $A$ is said to be positive semidefinite, written $A \succeq 0$, if one of the following three equivalent conditions holds:

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

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

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

**Note**

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

2. $A$ is positive semidefinite, if and only if all the entries of $D$ are non-negative.

3. The decomposition of $A$ as $A = W^T \cdot W$ 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 $D$.

4. We will assume that $A$ can in fact be decomposed as $A = W^T \cdot W$ 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.
Observations on semidefiniteness

Note

1. Any real, symmetric matrix $A$ can be decomposed as $A = L \cdot D \cdot L^T$, where $D$ is a diagonal matrix with the diagonal entries denoting the eigenvalues of $A$. 
Observations on semidefiniteness

Note

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

Note

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

2. $A$ is positive semidefinite, if and only if all the entries of $D$ are non-negative.
Observations on semidefiniteness

**Note**

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

2. *\( A \) is positive semidefinite, if and only if all the entries of* \( D \) *are non-negative.*

3. *The decomposition of* \( A \) *as* \( A = W^T \cdot W \) *is not polynomial time, since the entries could be irrational.*
Observations on semidefiniteness

Note

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

2. $A$ is positive semidefinite, if and only if all the entries of $D$ are non-negative.

3. The decomposition of $A$ as $A = W^T \cdot W$ 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 $D$. 
Observations on semidefiniteness

Note

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

2. $A$ is positive semidefinite, if and only if all the entries of $D$ are non-negative.

3. The decomposition of $A$ as $A = W^T \cdot W$ 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 $D$.

4. We will assume that $A$ can in fact be decomposed as $A = W^T \cdot W$ in polynomial time;
Observations on semidefiniteness

Note

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

2. $A$ is positive semidefinite, if and only if all the entries of $D$ are non-negative.

3. The decomposition of $A$ as $A = W^T \cdot W$ 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 $D$.

4. We will assume that $A$ can in fact be decomposed as $A = W^T \cdot W$ in polynomial time; the inaccuracy will be absorbed into the approximation factor.
Observations on semidefiniteness

**Note**

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

2. $A$ is positive semidefinite, if and only if all the entries of $D$ are non-negative.

3. The decomposition of $A$ as $A = W^T \cdot W$ 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 $D$.

4. We will assume that $A$ can in fact be decomposed as $A = W^T \cdot W$ 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

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

$C \cdot Y = c_{11} \cdot y_{11} + c_{12} \cdot y_{12} + \cdots + c_{ij} \cdot y_{ij} + \cdots + c_{kl} \cdot y_{kl} = \sum_{i=1}^{k} \sum_{j=1}^{l} c_{ij} \cdot y_{ij}$

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

**Definition**

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

\[
C \cdot Y = c_{11} \cdot y_{11} + c_{12} \cdot y_{12} + \cdots + c_{ij} \cdot y_{ij} + \cdots + c_{kl} \cdot y_{kl} = \sum_{i=1}^{k} \sum_{j=1}^{l} c_{ij} \cdot y_{ij}
\]

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

Definition

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

Note

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

Definition

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

$$\mathbf{C} \cdot \mathbf{Y} = \sum_{i=1}^{k} \sum_{j=1}^{l} c_{ij} y_{ij}$$

Note

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

Definition

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

\[
\mathbf{C} \cdot \mathbf{Y} = c_{11} \cdot y_{11} + c_{12} \cdot y_{12} + \cdots + c_{ij} \cdot y_{ij} + \cdots + c_{kl} \cdot y_{kl}
\]
Frobenius Inner Product

Definition

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

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

$= \sum_{i=1}^{k} \sum_{j=1}^{l} c_{ij} \cdot y_{ij}$
Frobenius Inner Product

**Definition**

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

\[
C \cdot Y = c_{11} \cdot y_{11} + c_{12} \cdot y_{12} + \cdots + c_{ij} \cdot y_{ij} + \cdots + c_{kl} \cdot y_{kl}
\]

\[
= \sum_{i=1}^{k} \sum_{j=1}^{l} c_{ij} \cdot y_{ij}
\]

**Note**

1. \( C \cdot Y \) is an inner product of \( C \) and \( Y \).
2. \( C \cdot Y \) is precisely \( \text{trace}(C^T \cdot Y) \).
3. \( C \cdot Y \) is a linear function of variables \( y_{ij} \) with coefficients \( c_{ij} \).
Frobenius Inner Product

**Definition**

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

\[
C \bullet Y = c_{11} \cdot y_{11} + c_{12} \cdot y_{12} + \cdots + c_{ij} \cdot y_{ij} + \cdots + c_{kl} \cdot y_{kl}
\]

\[
= \sum_{i=1}^{k} \sum_{j=1}^{l} c_{ij} \cdot y_{ij}
\]

**Note**

1. $C \bullet Y$ is an inner product of $C$ and $Y$. 
Frobenius Inner Product

**Definition**

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

\[
C \cdot Y = c_{11} \cdot y_{11} + c_{12} \cdot y_{12} + \cdots + c_{ij} \cdot y_{ij} + \cdots + c_{kl} \cdot y_{kl}
\]

\[
= \sum_{i=1}^{k} \sum_{j=1}^{l} c_{ij} \cdot y_{ij}
\]

**Note**

1. \( C \cdot Y \) is an inner product of \( C \) and \( Y \).
2. \( C \cdot Y \) is precisely \( \text{trace}(C^T \cdot Y) \).
Frobenius Inner Product

**Definition**

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

\[
C \cdot Y = c_{11} \cdot y_{11} + c_{12} \cdot y_{12} + \cdots + c_{ij} \cdot y_{ij} + \cdots + c_{kl} \cdot y_{kl} = \sum_{i=1}^{k} \sum_{j=1}^{l} c_{ij} \cdot y_{ij}
\]

**Note**

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

The Model

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

Let $C, D_1, D_2, \ldots, D_k \in M_n$ and let $d_1, d_2, \ldots, d_k \in \mathbb{R}$. The semidefinite programming problem (denoted by $S$) is defined as follows:

$$\max C \cdot Y$$
subject to

$$D_i \cdot Y = d_i, \quad 1 \leq i \leq k$$

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

Let $C, D_1, D_2, \ldots, D_k \in M_n$ and let $d_1, d_2, \ldots, d_k \in \mathbb{R}$. The semidefinite programming problem (denoted by $S$) is defined as follows:

$$\max C \cdot Y \quad \text{subject to} \quad D_i \cdot Y = d_i, \quad 1 \leq i \leq k \quad Y \succeq 0, \quad Y \in M_n$$
Semidefinite Programming

**The Model**

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

Let $C, D_1, D_2, \ldots, D_k \in M_n$ and let $d_1, d_2, \ldots, d_k \in \mathbb{R}$. The semidefinite programming problem (denoted by $S$) is defined as follows:

$$\max C \cdot Y$$
$$\text{subject to } D_i \cdot Y = d_i, \quad 1 \leq i \leq k$$
$$Y \succeq 0, \quad Y \in M_n$$
Let $\mathbb{M}_n$ denote the cone of symmetric $n \times n$ matrices. If $A \in \mathbb{M}_n$, we use $A \succeq 0$ to denote the fact that $A$ is positive semidefinite.

Let $C, D_1, D_2, \ldots, D_k \in \mathbb{M}_n$ and let $d_1, d_2, \ldots, d_k \in \mathbb{R}$. The semidefinite programming problem (denoted by $S$) is defined as follows: 

max $C \cdot Y$
subject to $D_i \cdot Y = d_i$, $1 \leq i \leq k$
$Y \succeq 0$, $Y \in \mathbb{M}_n$
Let $M_n$ denote the cone of symmetric $n \times n$ matrices. If $A \in M_n$, we use $A \succeq 0$ to denote the fact that $A$ is positive semidefinite.

Let $C, D_1, D_2, \ldots, D_k \in M_n$ and let $d_1, d_2, \ldots, d_k \in \mathbb{R}$. The semidefinite programming problem (denoted by $S$) is defined as follows:

$$\max C \cdot Y$$
Let $M_n$ denote the cone of symmetric $n \times n$ matrices. If $A \in M_n$, we use $A \succeq 0$ to denote the fact that $A$ is positive semidefinite.

Let $C, D_1, D_2, \ldots, D_k \in M_n$ and let $d_1, d_2, \ldots, d_k \in \mathbb{R}$. The semidefinite programming problem (denoted by $S$) is defined as follows:

$$\max \ C \cdot Y$$
subject to $D_i \cdot Y = d_i, \quad 1 \leq i \leq k$
Semidefinite Programming

The Model

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

2. Let $C, D_1, D_2, \ldots, D_k \in M_n$ and let $d_1, d_2, \ldots, d_k \in \mathbb{R}$. The semidefinite programming problem (denoted by $S$) is defined as follows:

$$\begin{align*}
\max & \quad C \cdot Y \\
\text{subject to} & \quad D_i \cdot Y = d_i, \quad 1 \leq i \leq k \\
& \quad Y \succeq 0,
\end{align*}$$
Let $M_n$ denote the cone of symmetric $n \times n$ matrices. If $A \in M_n$, we use $A \succeq 0$ to denote the fact that $A$ is positive semidefinite.

Let $C, D_1, D_2, \ldots, D_k \in M_n$ and let $d_1, d_2, \ldots, d_k \in \mathbb{R}$. The semidefinite programming problem (denoted by $S$) is defined as follows:

$$\max \quad C \cdot Y$$

subject to

$$D_i \cdot Y = d_i, \quad 1 \leq i \leq k$$

$$Y \succeq 0,$$

$$Y \in M_n$$
Properties of semidefinite programs

1. Semidefinite programming generalizes linear programming. If the matrices $C, D_1, D_2, \ldots, D_k$ are all diagonal, then the semidefinite program $S$ 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 $A \in \mathbb{R}^{n \times n}$ be an infeasible point. Let $C \in \mathbb{R}^{n \times n}$. A hyperplane $C \cdot Y \leq b$ is called a separating hyperplane for $A$, if all feasible points satisfy it and point $A$ 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.
Semidefinite Programming

Observation

Properties of semidefinite programs

Semidefinite programming generalizes linear programming. In the semidefinite program $S$, if the matrices $C$, $D_1$, $D_2$, ..., $D_k$ are all diagonal, then $S$ becomes a linear programming problem.

Inequality constraints can be added, without affecting the form of $S$.

The set of feasible solutions of $S$ forms a convex set (since the convex combination of semidefinite matrices is semidefinite).

Let $A \in \mathbb{R}^{n \times n}$ be an infeasible point. Let $C \in \mathbb{R}^{n \times n}$. A hyperplane $C \cdot Y \leq b$ is called a separating hyperplane for $A$, if all feasible points satisfy it and point $A$ does not satisfy it.

Most effective for maximization problems involving constraints between pairs of variables.

Leads to reasonably good approximation ratios, although in most cases, not the best possible.
Semidefinite Programming

Rudiments of Semidefinite Programming

Properties of semidefinite programs

Observation

1. **Semidefinite programming generalizes linear programming.**
Semidefinite programming generalizes linear programming. In the semidefinite program $S$,
if the matrices $C, D_1, D_2, \ldots, D_k$ 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 $C, D_1, D_2, \ldots, D_k$ 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 $C, D_1, D_2, \ldots, D_k$ 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 $C, D_1, D_2, \ldots, D_k$ 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 $C, D_1, D_2, \ldots, D_k$ 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 $A \in \mathbb{R}^{n \times n}$ be an infeasible point. Let $C \in \mathbb{R}^{n \times n}$. 
Properties of semidefinite programs

Observation

1. Semidefinite programming generalizes linear programming. In the semidefinite program $S$, if the matrices $C, D_1, D_2, \ldots, D_k$ 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 $A \in \mathbb{R}^{n \times n}$ be an infeasible point. Let $C \in \mathbb{R}^{n \times n}$. A hyperplane $C \cdot Y \leq b$ is called a separating hyperplane for $A$, if all feasible points satisfy it and point $A$ does not satisfy it.
Properties of semidefinite programs

Observation

1. **Semidefinite programming generalizes linear programming.** In the semidefinite program $S$, if the matrices $C, D_1, D_2, \ldots, D_k$ 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 $A \in \mathbb{R}^{n \times n}$ be an infeasible point.** Let $C \in \mathbb{R}^{n \times n}$. A hyperplane $C \cdot Y \leq b$ is called a separating hyperplane for $A$, if all feasible points satisfy it and point $A$ does not satisfy it.

5. **Most effective for maximization problems involving constraints between pairs of variables.**
Semidefinite Programming

Rudiments of Semidefinite Programming

Properties of semidefinite programs

Observation

1. Semidefinite programming generalizes linear programming. In the semidefinite program S, if the matrices \( C, D_1, D_2, \ldots, D_k \) 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 \( A \in \mathbb{R}^{n \times n} \) be an infeasible point. Let \( C \in \mathbb{R}^{n \times n} \). A hyperplane \( C \bullet Y \leq b \) is called a separating hyperplane for \( A \), if all feasible points satisfy it and point \( A \) 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.
Theorem

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

We can determine, in polynomial time, whether $A$ is feasible for $S$ and, if it is not, find a separating hyperplane.

Proof.

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

2. If $A$ is not positive semidefinite, then it has a negative eigenvalue, say $\lambda$. Let $v$ be the corresponding eigenvector. Now $(v \cdot v^T) \cdot Y = v^T \cdot Y \cdot v \geq 0$ is a separating hyperplane. (Violation of $Y \succeq 0$.)

3. If any of the linear constraints is violated, it directly yields a separating hyperplane. (Violation of $D_i \cdot Y = d_i$, for some $i = 1, 2, ..., k$.)

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

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

Proof.

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

2. If $A$ is not positive semidefinite, then it has a negative eigenvalue, say $\lambda$. Let $v$ be the corresponding eigenvector. Now $(v \cdot v^T) \cdot Y = v^T \cdot Y \cdot v \geq 0$ is a separating hyperplane. (Violation of $Y \succeq 0$.)

3. If any of the linear constraints is violated, it directly yields a separating hyperplane. (Violation of $D_i \cdot Y = d_i$, for some $i = 1, 2, \ldots, k$.)
Theorem

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

Proof.

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

2. If $A$ is not positive semidefinite, then it has a negative eigenvalue, say $\lambda$. Let $v$ be the corresponding eigenvector. Now $(v \cdot v^T) \cdot Y = v^T \cdot Y \cdot v \geq 0$ is a separating hyperplane. (Violation of $Y \succeq 0$.)

3. If any of the linear constraints is violated, it directly yields a separating hyperplane. (Violation of $D_i \cdot Y = d_i$, for some $i = 1, 2, \ldots, k$.)

Theorem

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

Proof.

If $A$ is feasible, then $A$ should be symmetric,
Separating hyperplane theorem

**Theorem**

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

**Proof.**

If $A$ is feasible, then $A$ should be symmetric, positive semidefinite,
**Theorem**

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

**Proof.**

If $A$ is feasible, then $A$ should be symmetric, positive semidefinite, and satisfy all the linear constraints.
Separating hyperplane theorem

Theorem

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

Proof.

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

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

Proof.

If $A$ is feasible, then $A$ 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:
Theorem

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

- If $\mathbf{A}$ is not symmetric, then $a_{ij} > a_{ji}$, for some $i, j$. 
Let $S$ be a semidefinite programming problem, and let $A$ be a point in $\mathbb{R}^{n \times n}$. We can determine, in polynomial time, whether $A$ is feasible for $S$ and, if it is not, find a separating hyperplane.

Proof.

If $A$ is feasible, then $A$ 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:

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

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

Proof.

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

Separating hyper-planes and vector programs

Theorem

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

Proof.

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

Theorem

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

Proof.

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

2. If $A$ is not positive semidefinite, then it has a negative eigenvalue, say $\lambda$. Let $v$ be the corresponding eigenvector.
Semidefinite Programming

Separating hyper-planes and vector programs

Separating hyperplane theorem

Theorem

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

Proof.

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

2. If $A$ is not positive semidefinite, then it has a negative eigenvalue, say $\lambda$. Let $v$ be the corresponding eigenvector. Now $(v \cdot v^T) \cdot Y = v^T \cdot Y \cdot v \geq 0$ is a separating hyperplane.
Semidefinite Programming

Separating hyper-planes and vector programs

Separating hyperplane theorem

Theorem

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

Proof.

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

2. If $A$ is not positive semidefinite, then it has a negative eigenvalue, say $\lambda$. Let $v$ be the corresponding eigenvector. Now $(v \cdot v^T) \cdot Y = v^T \cdot Y \cdot v \geq 0$ is a separating hyperplane. (Violation of $Y \succeq 0$.)
Theorem

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

Proof.

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

2. If $A$ is not positive semidefinite, then it has a negative eigenvalue, say $\lambda$. Let $v$ be the corresponding eigenvector. Now $(v \cdot v^T) \cdot Y = v^T \cdot Y \cdot v \geq 0$ is a separating hyperplane. (Violation of $Y \succeq 0$.)

3. If any of the linear constraints is violated, it directly yields a separating hyperplane. (Violation of $D_i \cdot Y = d_i$, for some $i = 1, 2, \ldots, k$.)
Lemma

Let V be a vector program on n n-dimensional vector variables $v_1, v_2, ..., v_n$.
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 $v_i \cdot v_j$ 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 $Y$, whose $(i, j)$th entry is $y_{ij}$, be symmetric and positive semidefinite.

Vector program V is equivalent to semidefinite program S.
Lemma

Let $V$ be a vector program on $n$ $n$-dimensional vector variables $v_1, v_2, \ldots, v_n$. 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 $v_i \cdot v_j$ 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 $Y$, whose $(i, j)$th entry is $y_{ij}$, be symmetric and positive semidefinite.

Vector program $V$ is equivalent to semidefinite program $S$. 

Equivalence between vector programs and semidefinite programs
Lemma

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

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 $v_i \cdot v_j$ 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 $Y$, whose $(i,j)$th entry is $y_{ij}$, be symmetric and positive semidefinite.

Vector program $V$ is equivalent to semidefinite program $S$. 

Equivalence between vector programs and semidefinite programs
Equivalence between vector programs and semidefinite programs

Lemma

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

Let $V$ be a vector program on $n$ $n$-dimensional vector variables $v_1, v_2, \ldots, v_n$. 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 $v_i \cdot v_j$ occurring in $V$ by the variable $y_{ij}$. 
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.
Lemma

Let $V$ be a vector program on $n$ $n$-dimensional vector variables $v_1, v_2, \ldots, v_n$. 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 $v_i \cdot v_j$ 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 $Y$, 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.

1. Let \( a_1, a_2, \ldots, a_n \) denote a feasible solution to the vector program \( V \).

2. Let \( W \) be the matrix whose columns are \( a_1, a_2, \ldots, a_n \).

3. \( A = W^T \cdot W \) is a feasible solution to the SDP \( S \), having the same objective function value.

4. Let \( A \) be a feasible solution to \( S \). Since \( A \) is positive semidefinite, there must exist an \( n \times n \) matrix \( W \), such that \( A = W^T \cdot W \).

5. Let \( a_1, a_2, \ldots, a_n \) denote the columns of \( W \).

6. These columns form a feasible solution to the vector program \( V \), having the same objective function value.
Proof of Equivalence

Proof.

Let $a_1, a_2, ..., a_n$ denote a feasible solution to the vector program $V$.

Let $W$ be the matrix whose columns are $a_1, a_2, ..., a_n$.

$A = W^T \cdot W$ is a feasible solution to the SDP $S$, having the same objective function value.

Let $A$ be a feasible solution to $S$. Since $A$ is positive semidefinite, there must exist an $n \times n$ matrix $W$, such that $A = W^T \cdot W$.

Let $a_1, a_2, ..., a_n$ denote the columns of $W$.

These columns form a feasible solution to the vector program $V$, having the same objective function value.
Proof.

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

2. Let $W$ be the matrix whose columns are $a_1, a_2, \ldots, a_n$.

3. $A = W^T \cdot W$ is a feasible solution to the SDP $S$, having the same objective function value.

4. Let $A$ be a feasible solution to $S$. Since $A$ is positive semidefinite, there must exist an $n \times n$ matrix $W$, such that $A = W^T \cdot W$.

5. Let $a_1, a_2, \ldots, a_n$ denote the columns of $W$.

6. These columns form a feasible solution to the vector program $V$, having the same objective function value.
Proof of Equivalence

Proof.

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

Proof.

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

Proof.

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