The Triangle Algorithm: An Algorithmic Separation Theorem and its Applications

Bahman Kalantari
Department of Computer Science, Rutgers University, New Brunswick, New Jersey, USA

DIMACS Workshop On Distance Geometry
July 26, 2016
The Convex Hull Membership Problem (CHMP)

Definition

Given a subset $S = \{v_1, \ldots, v_n\} \subset \mathbb{R}^m$, and $p \in \mathbb{R}^m$, either give a certificate that proves $p \in \text{conv}(S)$, or one that proves $p \notin \text{conv}(S)$.

Fact

$p \in \text{conv}(S) \iff p = \sum_{i=1}^{n} \alpha_i v_i, \sum_{i=1}^{n} \alpha_i = 1, \alpha_i \geq 0$.

Remark

When $p \notin \text{conv}(S)$ a certificate is a separating hyperplane.
The Convex Hull Membership Problem (CHMP)

Definition

Given a subset $S = \{v_1, \ldots, v_n\} \subset \mathbb{R}^m$, and $p \in \mathbb{R}^m$, either give a certificate that proves $p \in \text{conv}(S)$, or one that proves $p \notin \text{conv}(S)$.  

Fact

$p \in \text{conv}(S) \iff p = \sum_{i=1}^{n} \alpha_i v_i$, where $\sum_{i=1}^{n} \alpha_i = 1$, and $\alpha_i \geq 0$.

Remark

When $p \notin \text{conv}(S)$ a certificate is a separating hyperplane.
The Convex Hull Membership Problem (CHMP)

Definition

Given a subset $S = \{v_1, \ldots, v_n\} \subset \mathbb{R}^m$, and $p \in \mathbb{R}^m$, either give a certificate that proves $p \in \text{conv}(S)$, or one that proves $p \notin \text{conv}(S)$.

Fact

$$p \in \text{conv}(S) \iff p = \sum_{i=1}^{n} \alpha_i v_i, \quad \sum_{i=1}^{n} \alpha_i = 1, \quad \alpha_i \geq 0.$$
The Convex Hull Membership Problem (CHMP)

**Definition**
Given a subset $S = \{v_1, \ldots, v_n\} \subset \mathbb{R}^m$, and $p \in \mathbb{R}^m$, either give a certificate that proves $p \in \text{conv}(S)$, or one that proves $p \notin \text{conv}(S)$.

**Fact**

$$p \in \text{conv}(S) \iff p = \sum_{i=1}^{n} \alpha_i v_i, \quad \sum_{i=1}^{n} \alpha_i = 1, \quad \alpha_i \geq 0.$$  

**Remark**
When $p \notin \text{conv}(S)$ a certificate is a separating hyperplane.
The Convex Hull Membership Problem (CHMP)

Definition
Given a subset \( S = \{ v_1, \ldots, v_n \} \subset \mathbb{R}^m \), and \( p \in \mathbb{R}^m \), either give a certificate that proves \( p \in \text{conv}(S) \), or one that proves \( p \notin \text{conv}(S) \).

Fact
\[
p \in \text{conv}(S) \iff p = \sum_{i=1}^{n} \alpha_i v_i, \quad \sum_{i=1}^{n} \alpha_i = 1, \quad \alpha_i \geq 0.
\]

Remark
When \( p \notin \text{conv}(S) \) a certificate is a separating hyperplane.
Homogeneous and Approximate Version of CHMP

Definition

(Homogeneous CHMP (H-CHMP) )

Given an $m \times n$ matrix $A$, either find $x$ satisfying $Ax = 0$, $e^T x = 1$, $x \geq 0$, or prove unsolvable. (test if $0 \in \text{conv}(A)$)

Definition ($\varepsilon$-approximate version of CHMP)

Given $\varepsilon \in (0, 1)$, either compute $p_\varepsilon \in \text{conv}(S)$ such that:

$d(p_\varepsilon, p) \leq \varepsilon \cdot R$,

$R = \max\{d(p, v_1), \ldots, d(p, v_n)\}$;

or prove $p \not\in \text{conv}(S)$. ($d(u, v) = \|u - v\|$, Euclidean distance)
Homogeneous and Approximate Version of CHMP

**Definition**

*(Homogeneous CHMP (H-CHMP) )*

Given an $m \times n$ matrix $A$, either find $x$ satisfying

\[ Ax = 0, \quad e^T x = 1, \quad x \geq 0, \]

or prove unsolvable. (test if $0 \in \text{conv}(A)$)
Homogeneous and Approximate Version of CHMP

Definition

(Homogeneous CHMP (H-CHMP))

Given an $m \times n$ matrix $A$, either find $x$ satisfying

$$Ax = 0, \quad e^T x = 1, \quad x \geq 0,$$

or prove unsolvable. (test if $0 \in \text{conv}(A)$)

Definition

($\varepsilon$-approximate version of CHMP)

$$d(p_{\varepsilon}, p) \leq \varepsilon \cdot R, \quad R = \max \{d(p, v_1), \ldots, d(p, v_n)\};$$

or prove $p \not\in \text{conv}(S)$. ($d(u, v) = \|u - v\|$, Euclidean distance)
Homogeneous and Approximate Version of CHMP

**Definition**

*(Homogeneous CHMP (H-CHMP))*

Given an $m \times n$ matrix $A$, either find $x$ satisfying

$$Ax = 0, \quad e^T x = 1, \quad x \geq 0,$$

or prove unsolvable. (test if $0 \in \text{conv}(A)$)

**Definition**

*(\(\varepsilon\)-approximate version of CHMP)*

Given $\varepsilon \in (0, 1)$, either compute $p_\varepsilon \in \text{conv}(S)$ such that:
Homogeneous and Approximate Version of CHMP

Definition

(\textbf{Homogeneous CHMP (H-CHMP) })
Given an $m \times n$ matrix $A$, either find $x$ satisfying

$$Ax = 0, \quad e^T x = 1, \quad x \geq 0,$$

or prove unsolvable. (test if $0 \in \text{conv}(A)$)

Definition

($\varepsilon$-approximate version of CHMP)
Given $\varepsilon \in (0, 1)$, either compute $p_\varepsilon \in \text{conv}(S)$ such that:

$$d(p_\varepsilon, p) \leq \varepsilon \cdot R, \quad R = \max\{d(p, v_1), \ldots, d(p, v_n)\};$$
Homogeneous and Approximate Version of CHMP

**Definition**

*(Homogeneous CHMP (H-CHMP))*

Given an \( m \times n \) matrix \( A \), either find \( x \) satisfying

\[
Ax = 0, \quad e^T x = 1, \quad x \geq 0,
\]

or prove unsolvable. (test if \( 0 \in \text{conv}(A) \))

**Definition**

*(\( \varepsilon \)-approximate version of CHMP)*

Given \( \varepsilon \in (0, 1) \), either compute \( p_\varepsilon \in \text{conv}(S) \) such that:

\[
d(p_\varepsilon, p) \leq \varepsilon \cdot R, \quad R = \max\{d(p, v_1), \ldots, d(p, v_n)\};
\]

or prove \( p \notin \text{conv}(S) \). (\( d(u, v) = \|u - v\| \), Euclidean distance)
Significance of CHMP and H-CHMP

Applications in approximation theory, machine learning, statistics, etc.

It has given rise to significant dualities and algorithms:
- Gordan’s Theorem (1873) (preceded Farkas Lemma),
- Diagonal Scaling Dualities,
- Distance Dualities (to be described).

In fact these are most fundamental problems in linear programming.

Historically speaking, the first two polynomial-time LP algorithms happened to be (implicitly) designed for solve H-CHMP:
Applications in approximation theory, machine learning, statistics, etc.
Significance of CHMP and H-CHMP

- Applications in approximation theory, machine learning, statistics, etc.
- It has given rise to significant dualities and algorithms:
  - Gordan's Theorem (1873) (preceded Farkas Lemma)
  - Diagonal Scaling Dualities
  - Distance Dualities (to be described)

In fact these are most fundamental problems in linear programming.

Historically speaking, the first two polynomial-time LP algorithm happened to be (implicitly) designed for solve H-CHMP.
Significance of CHMP and H-CHMP

- Applications in approximation theory, machine learning, statistics, etc.
- It has given rise to significant dualities and algorithms: Gordan’s Theorem (1873) (preceded Farkas Lemma),
Applications in approximation theory, machine learning, statistics, etc.
- It has given rise to significant dualities and algorithms:
  - Gordan’s Theorem (1873) (preceded Farkas Lemma),
  - Diagonal Scaling Dualities,
Applications in approximation theory, machine learning, statistics, etc.
It has given rise to significant dualities and algorithms:
Gordan’s Theorem (1873) (preceded Farkas Lemma), Diagonal Scaling Dualities, Distance Dualities (to be described).
Significance of CHMP and H-CHMP

- Applications in approximation theory, machine learning, statistics, etc.
- It has given rise to significant dualities and algorithms: Gordan’s Theorem (1873) (preceded Farkas Lemma), Diagonal Scaling Dualities, Distance Dualities (to be described).
- In fact these are most fundamental problems in linear programming.
Significance of CHMP and H-CHMP

- Applications in approximation theory, machine learning, statistics, etc.
- It has given rise to significant dualities and algorithms:
  - Gordan’s Theorem (1873) (preceded Farkas Lemma),
  - Diagonal Scaling Dualities,
  - Distance Dualities (to be described).
- In fact these are most fundamental problems in linear programming.
- Historically speaking, the first two polynomial-time LP algorithm happened to be (implicitly) designed for solve H-CHMP:
Significance of CHMP and H-CHMP

• Karmarkar projective algorithm (1984) solves:
  \[
  \begin{align*}
  \min & \quad c^T x \\
  \text{subject to} & \quad Ax = 0, \\
  & \quad e^T x = 1, \\
  & \quad x \geq 0
  \end{align*}
  \]
  (H-CHMP)

• Khachiyan ellipsoid algorithm (1979) solves:
  \[
  \begin{align*}
  \text{Is } & \quad Ax < b \text{ feasible?} \\
  \text{Is } & \quad 0 \in \text{conv}(B)? \\
  \text{where } B = (A^T 0 b^T 1)
  \end{align*}
  \]
  (dual of \(Ax < b\))

• Khachiyan-K. matrix scaling algorithm (1992):
  Given an \(n \times n\) symmetric psd matrix \(A\), test the solvability of the following nonlinear dual to H-CHMP (\(0 \in \text{conv}(A)\)):
  \[
  D A D e = e,
  \]
  where \(D = \text{diag}(d_1, \ldots, d_n)\), \(d_i > 0\), \(e = (1, \ldots, 1)^T\).
Significance of CHMP and H-CHMP

- Karmarkar projective algorithm (1984) solves:

$$\min_{x} \left\{ c^T x : Ax = 0, \ e^T x = 1, \ x \geq 0 \right\} = 0$$  \((H-CHMP)\)

- Khachiyan ellipsoid algorithm (1979) solves:

$$\text{Is } Ax < b \text{ feasible?}$$

$$\text{Is } 0 \in \text{conv}(B)?$$

\(B = (A^T 0, b^T 1)\) (dual of \(Ax < b\))


Given an \(n \times n\) symmetric psd matrix \(A\), test the solvability of the following nonlinear dual to \(H-CHMP\) \((0 \in \text{conv}(A)?)\):

$$DADe = e, \ D = \text{diag}(d_1, \ldots, d_n), \ d_i > 0, \ e = (1, \ldots, 1)^T.$$
Significance of CHMP and H-CHMP

- Karmarkar projective algorithm (1984) solves:

\[ \text{Is } \min \{ c^T x : Ax = 0, \quad e^T x = 1, \quad x \geq 0 \} = 0? \quad (\text{H-CHMP}) \]
Significance of CHMP and H-CHMP

- Karmarkar projective algorithm (1984) solves:

$$\text{Is } \min \{c^T x : Ax = 0, \ e^T x = 1, \ x \geq 0\} = 0? \quad (\text{H-CHMP})$$

- Khachiyan ellipsoid algorithm (1979) solves:


Given an $n \times n$ symmetric psd matrix $A$, test the solvability of the following nonlinear dual to H-CHMP ($0 \in \text{conv}(A)$):

$$D A D e = e,$$
$$D = \text{diag}(d_1, \ldots, d_n), \quad d_i > 0,$$
$$e = (1, \ldots, 1)^T.$$
Significance of CHMP and H-CHMP

- Karmarkar projective algorithm (1984) solves:
  \[\text{Is } \min\{c^T x : Ax = 0, \ e^T x = 1, x \geq 0\} = 0? \quad (\text{H-CHMP})\]

- Khachiyan ellipsoid algorithm (1979) solves:
  \[\text{Is } Ax < b \text{ feasible?}\]
  \[\text{Is } 0 \in \text{conv}(B)\? , \quad B = \begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix}. \quad (\text{dual of } Ax < b)\]
Significance of CHMP and H-CHMP

• Karmarkar projective algorithm (1984) solves:

$$\text{Is } \min \{c^T x : Ax = 0, \ e^T x = 1, \ x \geq 0\} = 0? \quad \text{(H-CHMP)}$$

• Khachiyan ellipsoid algorithm (1979) solves:

$$\text{Is } Ax < b \text{ feasible?}$$

$$\text{Is } 0 \in \text{conv}(B)??, \ B = \begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix}. \quad \text{(dual of } Ax < b)$$

• Khachiyan-K. matrix scaling algorithm (1992):
Significance of CHMP and H-CHMP

- Karmarkar projective algorithm (1984) solves:

\[ \text{Is } \min \{ c^T x : Ax = 0, \ e^T x = 1, \ x \geq 0 \} = 0? \quad (\text{H-CHMP}) \]

- Khachiyan ellipsoid algorithm (1979) solves:

\[ \text{Is } Ax < b \text{ feasible?} \]

\[ \text{Is } 0 \in \text{conv}(B) ?, \quad B = \begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix}. \quad \text{(dual of } Ax < b) \]

- Khachiyan-K. matrix scaling algorithm (1992): Given an \( n \times n \) symmetric psd matrix \( A \), test the solvability of the following nonlinear dual to H-CHMP \( (0 \in \text{conv}(A) ?) \):

\[ D \text{ADe} = e, \quad D = \text{diag}(d_1, \ldots, d_n), \quad d_i > 0, \quad e = (1, \ldots, 1)^T. \]
Indeed a generalization of H-CHMP, called Homogeneous Programming, is a special but significant conic programming problem: Given a homogeneous function \( \phi(x) \), is \( \phi(x) = 0 \) for \( x \neq 0 \), \( x \in K \), a pointed cone. The matrix scaling equation \( DADe = e \) holds with appropriate interrelation of \( D \) and \( A \) and \( e \), dependent on \( \phi \) and \( K \). Moreover, algorithmic applications of these for semidefinite programming and self-concordant programming have been analyzed, e.g. "Semidefinite programming and matrix scaling over the semidefinite cone," *Linear Algebra and its Applications*, 2003, B.K. (Rutgers) Triangle Algorithm July 26, 2016 6 / 57.
• Indeed a generalization of H-CHMP, called Homogeneous Programming, is a special but significant conic programming problem: Given a homogeneous function $\phi(x)$,
Indeed a generalization of H-CHMP, called *Homogeneous Programming*, is a special but significant conic programming problem: Given a homogeneous function $\phi(x)$,

$$\text{Is } \phi(x) = 0?, \ x \neq 0, \ x \in K, \ \text{a pointed cone.}$$
Indeed a generalization of H-CHMP, called *Homogeneous Programming*, is a special but significant conic programming problem: Given a homogeneous function \( \phi(x) \),

\[
\text{Is } \phi(x) = 0, \ x \neq 0, \ x \in K, \quad \text{a pointed cone.}
\]

The matrix scaling equation \( DADe = e \) holds with appropriate interrelation of \( D \) and \( A \) and \( e \), dependent on \( \phi \) and \( K \).
Indeed a generalization of H-CHMP, called *Homogeneous Programming*, is a special but significant conic programming problem: Given a homogeneous function $\phi(x)$,

\[
\text{Is } \phi(x) = 0?, \ x \neq 0, \ x \in K, \text{ a pointed cone.}
\]

The matrix scaling equation $DADe = e$ holds with appropriate interrelation of $D$ and $A$ and $e$, dependent on $\phi$ and $K$. Moreover, algorithmic applications of these for semidefinite programming and self-concordant programming have been analyzed, e.g. “Semidefinite programming and matrix scaling over the semidefinite cone,” *Linear Algebra and its Applications*, 2003, B.K.
Triangle Algorithm : A Geometric Algorithm for CHMP

Triangle Algorithm (S = \{v_1, \ldots, v_n\}, p)

Step 1. Given iterate \( p' = \sum_{i=1}^n \alpha_i v_i \in \text{conv}(S) \), check if there exists a pivot: \( v_j \in S \) s.t. \( d(p', v_j) \geq d(p, v_j) \).

If no pivot exists, then \( p' \) is a witness. Stop.

Step 2. Otherwise, compute \( p'' = \text{nearest}(p; p' v_j) \):

\[ p'' = (1 - \alpha)p' + \alpha v_j = \sum_{i=1}^n \alpha'_{i} v_i, \quad \alpha = \frac{(p - p')^T (v_j - p')}{d_2(v_j, p')}, \]

\( \alpha'_{i} = (1 - \alpha)\alpha_j + \alpha, \quad \forall i \neq j. \)

Replace \( p' \) with \( p'' \) and Go to Step 1.
Triangle Algorithm (\(S = \{v_1, \ldots, v_n\}, p\)
Triangle Algorithm $(S = \{v_1, \ldots, v_n\}, p)$

- **Step 1.** Given \( p' = \sum_{i=1}^{n} \alpha_i v_i \in \text{conv}(S) \), check if there exists a pivot: \( v_j \in S \) s.t. \( d(p', v_j) \geq d(p, v_j) \).
Triangle Algorithm \((S = \{v_1, \ldots, v_n\}, p)\)

**Step 1.** Given iterate \(p' = \sum_{i=1}^{n} \alpha_i v_i \in \text{conv}(S)\), check if there exists a **pivot** : \(v_j \in S\) s.t. \(d(p', v_j) \geq d(p, v_j)\).
Triangle Algorithm \((S = \{v_1, \ldots, v_n\}, p)\)

- **Step 1.** Given iterate \(p' = \sum_{i=1}^{n} \alpha_i v_i \in \text{conv}(S)\), check if there exists a **pivot** \(v_j \in S\) s.t. 
  \[d(p', v_j) \geq d(p, v_j).\]

![Diagram of Triangle Algorithm]

If no pivot exists, then \(p'\) is a **witness**. Stop.
Triangle Algorithm (\(S = \{v_1, \ldots, v_n\}, p\))

**Step 1.** Given iterate \(p' = \sum_{i=1}^{n} \alpha_i v_i \in \text{conv}(S)\), check if there exists a pivot \(v_j \in S\) s.t. \(d(p', v_j) \geq d(p, v_j)\).

If no pivot exists, then \(p'\) is a witness. Stop.

**Step 2.** Otherwise, compute \(p'' = \text{nearest}(p; p'v)\):

\[
p'' = (1-\alpha)p' + \alpha v_j = \sum_{i=1}^{n} \alpha'_i v_i, \quad \alpha = \frac{(p - p')^T(v_j - p')}{d^2(v_j, p')},
\]

\[
\alpha'_j = (1-\alpha)\alpha_j + \alpha, \quad \alpha_i = (1-\alpha)\alpha_i, \quad \forall i \neq j.
\]

Replace \(p'\) with \(p''\) and Go to Step 1.
Example of Triangle Algorithm for a Triangle

Figure: Triangle Algorithm for testing if $p \in \text{conv} \{v_1, v_2, v_3\}$. 

(Rutgers) 
Triangle Algorithm 
July 26, 2016
Example of Triangle Algorithm for a Triangle

Figure: Triangle Algorithm for testing if $p \in \text{conv}(\{v_1, v_2, v_3\})$. 
Example of Triangle Algorithm for a Triangle

Figure: Triangle Algorithm for testing if $p \in \text{conv} \{v_1, v_2, v_3\}$.
Example of Triangle Algorithm for a Triangle

Figure: Triangle Algorithm for testing if $p \in \text{conv}(\{v_1, v_2, v_3\})$. 
Example of Triangle Algorithm for a Triangle

Figure: Triangle Algorithm for testing if \( p \in \text{conv}\{v_1, v_2, v_3\} \).
Example of Triangle Algorithm for a Triangle

Figure: Triangle Algorithm for testing if $p \in \text{conv}(\{v_1, v_2, v_3\})$. 
Example of Triangle Algorithm for a Triangle

Figure: Triangle Algorithm for testing if $p \in \text{conv}\{v_1, v_2, v_3\}$. 
Figure: Triangle Algorithm for testing if $p \in \text{conv}(\{v_1, v_2, v_3\})$. 
Figure: Triangle Algorithm for testing if $p \in \text{conv}(\{v_1, v_2, v_3\})$. 
Example of Triangle Algorithm for a Triangle

**Figure:** Triangle Algorithm for testing if $p \in \text{conv}(\{v_1, v_2, v_3\})$. 
Example of Triangle Algorithm for A Triangle
Figure: A case of \( p \notin \text{conv}(\{v_1, v_2, v_3\}) \).
Example of Triangle Algorithm for A Triangle

Figure: A case of $p \notin \text{conv}(\{v_1, v_2, v_3\})$. 
Figure: A case of $p \notin \text{conv}(\{v_1, v_2, v_3\})$. 
Example of Triangle Algorithm for A Triangle

Figure: A case of $p \not\in \text{conv}(\{v_1, v_2, v_3\})$. 
Example of Triangle Algorithm for A Triangle

**Figure:** A case of \( p \notin \text{conv}(\{v_1, v_2, v_3\}) \).
Figure: A case of \( p \not\in \text{conv}(\{v_1, v_2, v_3\}) \).
Figure: A case of $p \notin \text{conv}(\{v_1, v_2, v_3\})$. 
Figure: A case of $p \notin \text{conv}(\{v_1, v_2, v_3\})$. 
Figure: When orthogonal bisector of $pp'$ separate $p$ from $\text{conv}(S)$ (left) and when it does not.
Figure: When orthogonal bisector of $pp'$ separate $p$ from $\text{conv}(S)$ (left) and when it does not.
Figure: When orthogonal bisector of $pp'$ separate $p$ from $\text{conv}(S)$ (left) and when it does not.
Theorem (Distance Duality)
Precisely one of the two conditions is satisfied:

(i): For each $p' \in \text{conv}(S)$, there exists $v \in S$ such that $d(p', v) \geq d(p, v)$ (v a pivot).

(ii): There exists $p' \in \text{conv}(S)$ such that $d(p', v) < d(p, v)$, for all $v \in S$ ($p'$ a witness).

Furthermore, (i) is valid if and only if $p \in \text{conv}(S)$.
Equivalently, (ii) is valid if and only if $p \notin \text{conv}(S)$.

Remark: H.W. Kuhn (1967), proves this in the Euclidean plane making use of several results, including Ville's Lemma. Some generalizations of the theorem over normed spaces is given by Durier and Michelot (1986).
Theorem

(Distance Duality)

Precisely one of the two conditions is satisfied:

(i): For each $p' \in \text{conv}(S)$, there exists $v \in S$ such that $d(p', v) \geq d(p, v)$ (v a pivot)

(ii): There exists $p' \in \text{conv}(S)$ such that $d(p', v) < d(p, v)$, for all $v \in S$ ($p'$ a witness).

Furthermore, (i) is valid if and only if $p \in \text{conv}(S)$.

Equivalently, (ii) is valid if and only if $p \notin \text{conv}(S)$.

Remark

H.W. Kuhn (1967), proves this in the Euclidean plane making use of several results, including Ville's Lemma. Some generalizations of the theorem over normed spaces is given by Durier and Michelot (1986).
Theorem

*(Distance Duality)*

*Precisely one of the two conditions is satisfied:*

\(\text{For each } p' \in \text{conv}(S), \text{ there exists } v \in S \text{ such that } d(p', v) \geq d(p, v) \text{ (a pivot)}\)

\(\text{There exists } p' \in \text{conv}(S) \text{ such that } d(p', v) < d(p, v), \text{ for all } v \in S \text{ (a witness)}\)

Furthermore, (i) is valid if and only if \(p \in \text{conv}(S)\).

Equivalently, (ii) is valid if and only if \(p \notin \text{conv}(S)\).

*Remark* H.W. Kuhn (1967), proves this in the Euclidean plane making use of several results, including Ville's Lemma. Some generalizations of the theorem over normed spaces is given by Durier and Michelot (1986).
Theorem

*(Distance Duality)*

Precisely one of the two conditions is satisfied:

(i): For each \( p' \in \text{conv}(S) \), there exists \( v \in S \) such that \( d(p', v) \geq d(p, v) \) (v a pivot)

(ii): There exists \( p' \in \text{conv}(S) \) such that \( d(p', v) < d(p, v) \), for all \( v \in S \) (p' a witness).

Furthermore, (i) is valid if and only if \( p \in \text{conv}(S) \).

Equivalently, (ii) is valid if and only if \( p \notin \text{conv}(S) \).

Remark

H.W. Kuhn (1967), proves this in the Euclidean plane making use of several results, including Ville's Lemma. Some generalizations of the theorem over normed spaces is given by Durier and Michelot (1986).
Theorem

*(Distance Duality)*

Precisely one of the two conditions is satisfied:

(i): For each $p' \in \text{conv}(S)$, there exists $v \in S$ such that $d(p', v) \geq d(p, v)$ (v a pivot)

(ii): There exists $p' \in \text{conv}(S)$ such that $d(p', v) < d(p, v)$, for all $v \in S$ (p' a witness).

Remark

H.W. Kuhn (1967), proves this in the Euclidean plane making use of several results, including Ville's Lemma. Some generalizations of the theorem over normed spaces is given by Durier and Michelot (1986).
Geometry of Triangle Algorithm

Theorem

(\textit{Distance Duality})

Precisely one of the two conditions is satisfied:

(i): For each \( p' \in \text{conv}(S) \), there exists \( v \in S \) such that \( d(p', v) \geq d(p, v) \) (\( v \) a pivot)

(ii): There exists \( p' \in \text{conv}(S) \) such that \( d(p', v) < d(p, v) \), for all \( v \in S \) (\( p' \) a witness).

Furthermore, (i) is valid if and only if \( p \in \text{conv}(S) \).

Remark

H.W. Kuhn (1967), proves this in the Euclidean plane making use of several results, including Ville's Lemma. Some generalizations of the theorem over normed spaces is given by Durier and Michelot (1986).
Theorem

(Distance Duality)

Precisely one of the two conditions is satisfied:
(i): For each $p' \in \text{conv}(S)$, there exists $v \in S$ such that $d(p', v) \geq d(p, v)$ (v a pivot)
(ii): There exists $p' \in \text{conv}(S)$ such that $d(p', v) < d(p, v)$, for all $v \in S$ (p' a witness).

Furthermore, (i) is valid if and only if $p \in \text{conv}(S)$.
Equivalently, (ii) is valid if and only if $p \notin \text{conv}(S)$. 
Theorem

(Distance Duality)
Precisely one of the two conditions is satisfied:
(i): For each \( p' \in \text{conv}(S) \), there exists \( v \in S \) such that 
\[
\text{d}(p', v) \geq \text{d}(p, v) \quad (v \text{ a pivot})
\]
(ii): There exists \( p' \in \text{conv}(S) \) such that 
\[
\text{d}(p', v) < \text{d}(p, v), \quad \text{for all } v \in S 
\]
\( (p' \text{ a witness}) \).
Furthermore, (i) is valid if and only if \( p \in \text{conv}(S) \).
Equivalently, (ii) is valid if and only if \( p \notin \text{conv}(S) \).

Remark

H.W. Kuhn (1967), proves this in the Euclidean plane making use of several results, including Ville’s Lemma. Some generalizations of the theorem over normed spaces is given by Durier and Michelot (1986).
Theorem

Given two consecutive iterates $p', p''$, corresponding to the triangle $\triangle pp'v$ with $v$ a pivot, let $\delta = d(p', p)$, $\delta' = d(p'', p)$, and $r = d(p, v)$.

Then, if $\delta \leq r$, $\delta' \leq \delta \sqrt{1 - \frac{\delta^2}{4r^2}}$. 

(Rutgers)
Theorem

Given two consecutive iterates $p', p''$, corresponding to the triangle $\triangle pp'v$ with $v$ a pivot, let

$\delta = d(p', p)$,

$\delta' = d(p'', p)$,

$r = d(p, v)$.

Then, if $\delta \leq r$, $\delta' \leq \delta \sqrt{1 - \delta^2/4r^2}$. 

(Rutgers)
Theorem

Given two consecutive iterates $p'$, $p''$, corresponding to the triangle $\Delta pp'v$ with $v$ a pivot, let $\delta = d(p', p)$, $\delta' = d(p'', p)$, and $r = d(p, v)$. Then, if $\delta \leq r$, $\delta' \leq \delta \sqrt{1 - \frac{\delta^2}{4r^2}}$. 
Theorem

Given two consecutive iterates $p'$, $p''$, corresponding to the triangle $\triangle pp'v$ with $v$ a pivot, let $\delta = d(p', p)$, $\delta' = d(p'', p)$, and $r = d(p, v)$. Then, if $\delta \leq r$,

$$\delta' \leq \delta \sqrt{1 - \frac{\delta^2}{4r^2}}.$$
Theorem

(i) Suppose $p \in \text{conv}(S)$. Given $\varepsilon > 0$, the number of iterations to compute a point $p_\varepsilon$ in $\text{conv}(S)$ so that $d(p, p_\varepsilon) \leq \varepsilon R$, $R = \max\{d(p, v_1), \ldots, d(p, v_n)\}$ is

$$O\left(\frac{1}{\varepsilon^2}\right).$$
Theorem

(i) Suppose $p \in \text{conv}(S)$. Given $\varepsilon > 0$, the number of iterations to compute a point $p_\varepsilon$ in $\text{conv}(S)$ so that $d(p, p_\varepsilon) \leq \varepsilon R$, $R = \max\{d(p, v_1), \ldots, d(p, v_n)\}$ is

$$O\left(\frac{1}{\varepsilon^2}\right).$$

(ii) Suppose $p \notin \text{conv}(S)$. The number of iterations to compute a witness $p'$ in $\text{conv}(S)$ is

$$O\left(\frac{R^2}{\Delta^2}\right), \quad \Delta = \min \{d(x, p) : x \in \text{conv}(S)\}.$$
Remarks on the Triangle Algorithm

In straightforward implementation, worst-case complexity in each iteration is $O(mn)$ arithmetic operations.

With a preprocessing time of $O(mn^2)$, each iteration can be implemented in $O(m+n)$ arithmetic operations.

To find pivot Triangle Algorithm does not require taking square-roots:

$$
d(p', v) \geq d(p, v) \iff \|p'\|^2 - \|p\|^2 \geq 2v^T(p' - p).
$$
Remark

In straightforward implementation, worst-case complexity in each iteration is $O(mn)$ arithmetic operations.
Remark

In straightforward implementation, worst-case complexity in each iteration is $O(mn)$ arithmetic operations.

Remark

With a preprocessing time of $O(mn^2)$, each iteration can be implemented in $O(m + n)$ arithmetic operations.
Remarks on the Triangle Algorithm

Remark

In straightforward implementation, worst-case complexity in each iteration is \( O(mn) \) arithmetic operations.

Remark

With a preprocessing time of \( O(mn^2) \), each iteration can be implemented in \( O(m + n) \) arithmetic operations.

Remark

To find pivot Triangle Algorithm does not require taking square-roots:

\[
d(p', v) \geq d(p, v) \iff \|p'\|^2 - \|p\|^2 \geq 2v^T(p' - p).
\]
Remarks on Other Algorithms for Solving CHMP

• Simplex Method solves CHMP as Phase I.
• Sparse greedy approximation solves CHMP by conversion into a convex quadratic minimization over a simplex.
• Sparse greedy approximation is equivalent to Frank-Wolf, also Gilbert's algorithm.
• Motivation behind their iterative steps is algebraic - Triangle Algorithm is motivated by geometric properties.
• So-called fast gradient method of Nesterov can also be applied, an $O\left(\frac{1}{\varepsilon}\right)$ iteration algorithm, complexity of each iteration is $O(mn)$.
• Worst-case complexity of each iteration of Triangle Algorithm is $O(mn)$. However, even without preprocessing, often, each iteration requires only $O(m + n)$.
• Triangle Algorithm could outperform these due to distance duality, simplicity and degrees of freedom it offers.
Remarks on Other Algorithms for Solving CHMP

Remark

- *Simplex Method solves CHMP as Phase I.*
Remarks on Other Algorithms for Solving CHMP

Remark

• Simplex Method solves CHMP as Phase I.
• Sparse greedy approximation solves CHMP by conversion into a convex quadratic minimization over a simplex.
Remarks on Other Algorithms for Solving CHMP

Remark

- Simplex Method solves CHMP as Phase I.
- Sparse greedy approximation solves CHMP by conversion into a convex quadratic minimization over a simplex.
- Sparse greedy approximation is equivalent to Frank-Wolf, also Gilbert’s algorithm.
Remarks on Other Algorithms for Solving CHMP

Remark

- Simplex Method solves CHMP as Phase I.
- Sparse greedy approximation solves CHMP by conversion into a convex quadratic minimization over a simplex.
- Sparse greedy approximation is equivalent to Frank-Wolf, also Gilbert’s algorithm. Motivation behind their iterative steps is algebraic - Triangle Algorithm is motivated by geometric properties.
Remarks on Other Algorithms for Solving CHMP

**Remark**

- **Simplex Method** solves CHMP as Phase I.
- **Sparse greedy approximation** solves CHMP by conversion into a convex quadratic minimization over a simplex.
- **Sparse greedy approximation** is equivalent to Frank-Wolf, also Gilbert’s algorithm. Motivation behind their iterative steps is algebraic - Triangle Algorithm is motivated by geometric properties.
- So-called fast gradient method of Nesterov can also be applied, an $O(1/\varepsilon)$ iteration algorithm, complexity of each iteration is $O(mn)$. 
Remarks on Other Algorithms for Solving CHMP

**Remark**

- Simplex Method solves CHMP as Phase I.
- Sparse greedy approximation solves CHMP by conversion into a convex quadratic minimization over a simplex.
- Sparse greedy approximation is equivalent to Frank-Wolf, also Gilbert’s algorithm. Motivation behind their iterative steps is algebraic - Triangle Algorithm is motivated by geometric properties.
- So-called fast gradient method of Nesterov can also be applied, an $O(1/\varepsilon)$ iteration algorithm, complexity of each iteration is $O(mn)$.
- Worst-case complexity of each iteration of Triangle Algorithm is $O(mn)$. However, even without preprocessing, often, each iteration requires only $O(m + n)$.
Remarks on Other Algorithms for Solving CHMP

Remark

• Simplex Method solves CHMP as Phase I.
• Sparse greedy approximation solves CHMP by conversion into a convex quadratic minimization over a simplex.
• Sparse greedy approximation is equivalent to Frank-Wolf, also Gilbert’s algorithm. Motivation behind their iterative steps is algebraic - Triangle Algorithm is motivated by geometric properties.
• So-called fast gradient method of Nesterov can also be applied, an $O(1/\varepsilon)$ iteration algorithm, complexity of each iteration is $O(mn)$.
• Worst-case complexity of each iteration of Triangle Algorithm is $O(mn)$. However, even without preprocessing, often, each iteration requires only $O(m + n)$.
• Triangle Algorithm could outperform these due to distance duality, simplicity and degrees of freedom it offers.
Experimental Results with Triangle Algorithm
Experimental Results with Triangle Algorithm

Figure: Running time comparison as n grows
Experimental Results with Triangle Algorithm
Experimental Results with Triangle Algorithm

Figure: Running time comparison as m grows
Experimental Results with Triangle Algorithm

As the number of points \( n \) grow, the running time of the Simplex and Frank-Wolfe methods increase while the Triangle Algorithm performs very well with only a slight increase in the running time.

One explanation is the fact that the Triangle Algorithm does not need to make use of all the \( n \) points and thus spends less time than the simplex method and Frank-Wolfe in each iteration. Another is that by virtue of selecting a pivot it makes good reductions in each iteration.

<table>
<thead>
<tr>
<th>( n )</th>
<th># of points visited per iteration</th>
<th># of iterations</th>
<th>Running Time</th>
<th># of points visited per iteration</th>
<th># of iterations</th>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table: The performance of Triangle algorithm when \( m=100 \)
As the number of points $n$ grow, the running time of the Simplex and Frank-Wolfe methods increase while the Triangle Algorithm performs very well with only a slight increase in the running time.
Experimental Results with Triangle Algorithm

As the number of points $n$ grow, the running time of the Simplex and Frank-Wolfe methods increase while the Triangle Algorithm performs very well with only a slight increase in the running time.

One explanation is the fact that the Triangle Algorithm does not need to make use of all the $n$ points and thus spends less time than the simplex method and Frank-Wolfe in each iteration. Another is that by virtue of selecting a pivot it makes good reductions in each iteration.

<table>
<thead>
<tr>
<th>$n$</th>
<th># of points visited per iteration</th>
<th>iterations</th>
<th>running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>185</td>
<td>459.6</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>228.26</td>
<td>479.6</td>
<td></td>
</tr>
<tr>
<td>3000</td>
<td>240.37</td>
<td>540.4</td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>242.22</td>
<td>541.6</td>
<td></td>
</tr>
<tr>
<td>10000</td>
<td>254.84</td>
<td>535.4</td>
<td></td>
</tr>
</tbody>
</table>
Experimental Results with Triangle Algorithm

As the number of points $n$ grow, the running time of the Simplex and Frank-Wolfe methods increase while the Triangle Algorithm performs very well with only a slight increase in the running time.

One explanation is the fact that the Triangle Algorithm does not need to make use of all the $n$ points and thus spends less time than the simplex method and Frank-Wolfe in each iteration. Another is that by virtue of selecting a pivot it makes good reductions in each iteration.

<table>
<thead>
<tr>
<th>$n$</th>
<th># of points visited per iteration</th>
<th>iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>185</td>
<td>459.6</td>
</tr>
<tr>
<td>1000</td>
<td>228.26</td>
<td>479.6</td>
</tr>
<tr>
<td>3000</td>
<td>240.37</td>
<td>540.4</td>
</tr>
<tr>
<td>5000</td>
<td>242.22</td>
<td>541.6</td>
</tr>
<tr>
<td>10000</td>
<td>254.84</td>
<td>535.4</td>
</tr>
</tbody>
</table>

**Table**: The performance of Triangle algorithm when $m=100$
Properties and Characterizations of Witnesses: Separation Property

Definition

Let $W_p$ be the set of all witnesses, i.e. points $p' \in \text{conv}(S)$ such that $d(p', v_i) < d(p, v_i), \forall i = 1, \ldots, n$.

Theorem

If $p' \in W_p$ the orthogonal bisecting hyperplane of the line segment $pp'$ separates $p$ from $\text{conv}(S)$.
Properties and Characterizations of Witnesses: Separation Property

Definition
Let \( W_p \) be the set of all witnesses, i.e. points \( p' \in \text{conv}(S) \) such that

\[
d(p', v_i) < d(p, v_i), \quad \forall i = 1, \ldots, n.\]
Properties and Characterizations of Witnesses: Separation Property

**Definition**

Let $W_p$ be the set of all witnesses, i.e. points $p' \in \text{conv}(S)$ such that

$$d(p', v_i) < d(p, v_i), \quad \forall i = 1, \ldots, n.$$ 

**Theorem**

If $p' \in W_p$ the orthogonal bisecting hyperplane of the line segment $pp'$ separates $p$ from $\text{conv}(S)$. 
Approximation of Distance to Convex Hull

Suppose \( p \not\in \text{conv}(S) = \text{conv}\{v_1,...,v_n\} \).

Let \( \Delta = d(p, \text{conv}(S)) = \min\{d(p, x) : x \in \text{conv}(S)\} \).

Then any witness \( p' \in W \) gives an estimate of \( \Delta \) to within a factor of two. More precisely,

\[
\frac{1}{2} d(p, p') \leq \Delta \leq d(p, p')
\]
Properties and Characterizations of Witnesses: Approximation of Distance to Convex Hull

Corollary

Suppose \( p \notin \text{conv}(S) = \text{conv}\{v_1, \ldots, v_n\} \).
Corollary

Suppose \( p \not\in \text{conv}(S) = \text{conv}(\{v_1, \ldots, v_n\}) \).

Let

\[
\Delta = d(p, \text{conv}(S)) = \min\{d(p, x) : x \in \text{conv}(S)\}.
\]
Corollary

Suppose $p \notin \text{conv}(S) = \text{conv}(\{v_1, \ldots, v_n\})$. Let

$$\Delta = d(p, \text{conv}(S)) = \min\{d(p, x) : x \in \text{conv}(S)\}.$$ 

Then any witness $p' \in W_p$ gives an estimate of $\Delta$ to within a factor of two. More precisely,
Suppose \( p \not\in \text{conv}(S) = \text{conv}(\{v_1, \ldots, v_n\}) \). Let
\[
\Delta = d(p, \text{conv}(S)) = \min \{d(p, x) : x \in \text{conv}(S)\}.
\]
Then any witness \( p' \in W_p \) gives an estimate of \( \Delta \) to within a factor of two. More precisely,
\[
\frac{1}{2} d(p, p') \leq \Delta \leq d(p, p').
\]
Intersection Ball Property

Given \( S = \{v_1, \ldots, v_n\} \) and \( p \) all in \( \mathbb{R}^m \), consider the set of open balls \( B_i \) balls centered at \( v_i \) with radius \( d(p, v_i) \), \( i = 1, \ldots, n \).

Then \( p \in \text{conv}(S) \) if and only if \( \bigcap_{i=1}^n B_i \cap \text{conv}(S) = \emptyset \).

Equivalently, \( p \in \text{conv}(S) \) if and only if \( \bigcap_{i=1}^n B_i \cap \text{conv}(S) = \emptyset \).
Corollary

Given $S = \{v_1, \ldots, v_n\}$ and $p$ all in $\mathbb{R}^m$, consider the set of open balls $B_i$ balls centered at $v_i$ with radius $d(p, v_i)$, $i = 1, \ldots, n$. Then $p \in \text{conv}(S)$ if and only if $(\bigcap_{i=1}^n B_i) \cap \text{conv}(S) = \emptyset$. Equivalently, $p \in \text{conv}(S)$ if and only if $(\bigcap_{i=1}^n \overline{B_i}) \cap \text{conv}(S) = \emptyset$. 
A Case with No Witness: $p \in \text{conv}(S)$
A Case with No Witness: $p \in \text{conv}(S)$

Figure: No witnesses: $p \in \text{conv}(S)$. The three discs intersect only at $p$.
Some Cases with Witnesses: $p \notin \text{conv}(S)$
Some Cases with Witnesses: \( p \notin conv(S) \)

Figure: Examples with \( W_p \neq \emptyset, p \notin conv(S) \). \( W_p \) is interior of gray areas: For any \( p' \in W_p \) the bisector of \( pp' \) separates \( p \) from \( conv(S) \).
Strict Distance Duality

Definition

Given $p' \in \text{conv}(S)$, we say $v \in S$ is a strict pivot if $\angle p'pv \geq \pi/2$.

Theorem (Strict Distance Duality) Assume $p \not\in S$. Then $p \in \text{conv}(S)$ if and only if for each $p' \in \text{conv}(S)$ there exists a strict pivot.
Strict Distance Duality

**Definition**

Given $p' \in \text{conv}(S)$, we say $v \in S$ is a *strict pivot* if $\angle p'pv \geq \pi/2$.
Strict Distance Duality

**Definition**

Given $p' \in \text{conv}(S)$, we say $v \in S$ is a *strict pivot* if $\angle p'pv \geq \pi/2$. 

![Diagram showing the definition of a strict pivot with points $p$, $v$, and $p''$, and angles $\delta$, $\delta'$, and $r$.]
**Definition**

Given $p' \in \text{conv}(S)$, we say $v \in S$ is a strict pivot if $\angle p'pv \geq \pi/2$.

**Theorem**

*Strict Distance Duality* Assume $p \notin S$. Then $p \in \text{conv}(S)$ if and only if for each $p' \in \text{conv}(S)$ there exists a strict pivot.
Theorem

Assume $p$ lies at the center of a ball of radius $\rho$ in the relative interior of $\text{conv}(S)$, and Triangle Algorithm uses strict pivot in each iteration. The number of iterations to compute $p \in \text{conv}(S)$ such that $d(p, p_\epsilon) < \epsilon$ satisfies

$$O\left(\left(\frac{R}{\rho}\right)^2 \log \frac{1}{\epsilon}\right).$$
Theorem

Assume $p$ lies at the center of a ball of radius $\rho$ in the relative interior of $\text{conv}(S)$, and Triangle Algorithm uses strict pivot in each iteration. The number of iterations to compute $p \in \text{conv}(S)$ such that $d(p, p_\varepsilon) < \varepsilon$, $R = \max\{d(p, v_i) : i = 1, \ldots, n\}$ satisfies

$$O\left((R \rho)^2 \log \frac{1}{\varepsilon}\right).$$
Theorem

Assume $p$ lies at the center of a ball of radius $\rho$ in the relative interior of $\text{conv}(S)$, and Triangle Algorithm uses strict pivot in each iteration.
Theorem

Assume \( p \) lies at the center of a ball of radius \( \rho \) in the relative interior of \( \text{conv}(S) \), and Triangle Algorithm uses strict pivot in each iteration. The number of iterations to compute \( p_\varepsilon \in \text{conv}(S) \) such that

\[
d(p, p_\varepsilon) < \varepsilon R, \quad R = \max\{d(p, v_i), i = 1, \ldots, n\}
\]

satisfies
Theorem

Assume \( p \) lies at the center of a ball of radius \( \rho \) in the relative interior of \( \text{conv}(S) \), and Triangle Algorithm uses strict pivot in each iteration. The number of iterations to compute \( p_\varepsilon \in \text{conv}(S) \) such that

\[
d(p, p_\varepsilon) < \varepsilon R, \quad R = \max\{d(p, v_i), i = 1, \ldots, n\}
\]

satisfies

\[
O\left(\left(\frac{R}{\rho}\right)^2 \log \frac{1}{\varepsilon}\right).
\]
Theorem
Given \( \epsilon \in (0, 1) \), the number of iterations of the Triangle Algorithm to test if there exists \( p \in \text{conv}(S) \) such that \( d(p, p') < \epsilon \),
\[ R = \max \{ d(p, v_i) \mid i = 1, \ldots, n \}, \]
is \( O\left(\frac{1}{c} \ln \frac{1}{\epsilon} \right) \), (1)
where \( c \) is the visibility factor, a constant satisfying the inequalities
\[ \sin(p'v') \leq \frac{1}{\sqrt{1 + c}}, \]
\[ c \geq \epsilon^2, \]
(2)
over all the iterates \( p' \) having corresponding pivot \( v' \).
Theorem

Given $\varepsilon \in (0, 1)$, the number of iterations of the Triangle Algorithm to test if there exists $p \in \text{conv}(S)$ such that $d(p, p') < \varepsilon R$, $R = \max\{d(p, v_i) : i = 1, \ldots, n\}$, is $O\left(\frac{1}{c} \ln \frac{1}{\varepsilon}\right)$, (1) where $c$ is the visibility factor, a constant satisfying the inequalities $\sin(\theta p v') \leq \frac{1}{\sqrt{1 + c}}, c \geq \varepsilon^2$, (2) over all the iterates $p'$ having corresponding pivot $v'$. 
Theorem

Given $\varepsilon \in (0, 1)$, the number of iterations of the Triangle Algorithm to test if there exists $p_\varepsilon \in \text{conv}(S)$ such that $d(p, p_\varepsilon) < \varepsilon R$, $R = \max\{d(p, v_i), i = 1, \ldots, n\}$, is

\[ O\left(\frac{1}{c} \ln \frac{1}{\varepsilon}\right) \]
Theorem

Given \( \varepsilon \in (0, 1) \), the number of iterations of the Triangle Algorithm to test if there exists \( p_\varepsilon \in \text{conv}(S) \) such that \( d(p, p_\varepsilon) < \varepsilon R \), where

\[
R = \max\{d(p, v_i), i = 1, \ldots, n\},
\]

is

\[
O\left(\frac{1}{c} \ln \frac{1}{\varepsilon}\right),
\]  

(1)

and

\[
\sin(\angle p p' v') \leq \frac{\sqrt{1 + \varepsilon^2}}{c},
\]

(2)

over all the iterates \( p' \) having corresponding pivot \( v' \).
Theorem

Given \( \varepsilon \in (0, 1) \), the number of iterations of the Triangle Algorithm to test if there exists \( p_\varepsilon \in \text{conv}(S) \) such that \( d(p, p_\varepsilon) < \varepsilon R \), \( R = \max\{d(p, v_i), i = 1, \ldots, n\} \), is

\[
O\left(\frac{1}{c} \ln \frac{1}{\varepsilon}\right),
\]

where \( c \) is the visibility factor, a constant satisfying the inequalities

\[
\sin(pp'v') \leq \frac{1}{\sqrt{1 + c}}, \quad c \geq \varepsilon^2,
\]

over all the iterates \( p' \) having corresponding pivot \( v' \).
We say $p' \in \text{conv}(S)$ is a strict witness if there is no strict pivot at $p'$. Equivalently, $p'$ is a strict witness if the orthogonal hyperplane to the line $p'p$ at $p$ separates $p$ from $\text{conv}(S)$. Denote the set of all strict witnesses by $\hat{W}_p$. $\hat{W}_p$ contains $W_p$.

Proposition

We have $\hat{W}_p = \{x \in \text{conv}(S) : (x - p)^T(v_i - p) > 0, i = 1, \ldots, n\}$. 

(Rutgers)
We say \( p' \in \text{conv}(S) \) is a **strict witness** if there is no strict pivot at \( p' \). Equivalently, \( p' \) is a strict witness if the orthogonal hyperplane to the line \( p'p \) at \( p \) separates \( p \) from \( \text{conv}(S) \). Denote the set of all strict witnesses by \( \hat{W}_p \).
**Definition**

We say \( p' \in \text{conv}(S) \) is a *strict witness* if there is no strict pivot at \( p' \). Equivalently, \( p' \) is a strict witness if the orthogonal hyperplane to the line \( p'p \) at \( p \) separates \( p \) from \( \text{conv}(S) \). Denote the set of all strict witnesses by \( \widehat{W}_p \).

\( \widehat{W}_p \) contains \( W_p \).
Definition

We say $p' \in \text{conv}(S)$ is a strict witness if there is no strict pivot at $p'$. Equivalently, $p'$ is a strict witness if the orthogonal hyperplane to the line $p'p$ at $p$ separates $p$ from $\text{conv}(S)$. Denote the set of all strict witnesses by $\widehat{W}_p$.

$\widehat{W}_p$ contains $W_p$.

Proposition

We have

$$\widehat{W}_p = \left\{ x \in \text{conv}(S) : (x - p)^T(v_i - p) > 0, i = 1, \ldots, n \right\}.$$
Figure: Witness set $W_p$ (left) and Strict Witness set $\hat{W}_p$ (right).
Test if \( Ax < b \) is feasible, \( A \) is an \( m \times n \) matrix. (The problem Khachiyan considered in 1979).

\( Ax < b \) is feasible if and only if the following CHMP is infeasible:

\[
(A^T a \quad b^T s) y = (0 \quad 0), \quad y \geq 0, \quad s \geq 0.
\]

Denote rows of \( A \) by \( a^T i \).

Then columns of matrix in CHMP are \( v_i = (a^T i \quad b^T i) \), \( i = 1, \ldots, m \) and \( v_{m+1} = (0 \quad 1) \), all in \( \mathbb{R}^{n+1} \).

Suppose triangle algorithm for CHMP gives a witness \( p' = (x \quad \alpha) \).

Then, \( A(-x/\alpha) < b \).

In other words, triangle algorithm gives complete answer when testing the feasibility of \( Ax < b \), not just a yes answer.
Test if $Ax < b$ is feasible, $A$ is an $m \times n$ matrix.
Test if $Ax < b$ is feasible, $A$ is an $m \times n$ matrix. (The problem Khachiyan considered in 1979).
Test if $Ax < b$ is feasible, $A$ is an $m \times n$ matrix. 
(The problem Khachiyan considered in 1979).
$Ax < b$ is feasible if and only if the following CHMP is infeasible

\[
\begin{pmatrix}
A^T & 0 \\
b^T & 1
\end{pmatrix}
\begin{pmatrix}
y \\
s
\end{pmatrix} =
\begin{pmatrix}
0 \\
0
\end{pmatrix}, \quad \sum_{i=1}^{m} y_i + s = 1, \quad y \geq 0, \quad s \geq 0.
\]
Test if $Ax < b$ is feasible, $A$ is an $m \times n$ matrix. (The problem Khachiyan considered in 1979).

$Ax < b$ is feasible if and only if the following CHMP is infeasible

\[
\begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix} \begin{pmatrix} y \\ s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \sum_{i=1}^{m} y_i + s = 1, \quad y \geq 0, \quad s \geq 0.
\]

Denote rows of $A$ by $a_i^T$. 

(Rutgers)
Test if $Ax < b$ is feasible, $A$ is an $m \times n$ matrix. (The problem Khachiyan considered in 1979). $Ax < b$ is feasible if and only if the following CHMP is infeasible

\[
\begin{pmatrix}
A^T & 0 \\
b^T & 1
\end{pmatrix}
\begin{pmatrix}
y \\
s
\end{pmatrix}
= \begin{pmatrix}
0 \\
0
\end{pmatrix}, \quad \sum_{i=1}^{m} y_i + s = 1, \quad y \geq 0, \quad s \geq 0.
\]

Denote rows of $A$ by $a_i^T$. Then columns of matrix in CHMP are $v_i = \begin{pmatrix} a_i \\ b_i \end{pmatrix}$, $i = 1, \ldots, m$ and $v_{m+1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, all in $\mathbb{R}^{n+1}$. 

Test if $Ax < b$ is feasible, $A$ is an $m \times n$ matrix. (The problem Khachiyan considered in 1979).

$Ax < b$ is feasible if and only if the following CHMP is infeasible

\[
\begin{pmatrix}
A^T & 0 \\
b^T & 1
\end{pmatrix}
\begin{pmatrix}
y \\
s
\end{pmatrix} =
\begin{pmatrix}
0 \\
0
\end{pmatrix}, \quad \sum_{i=1}^{m} y_i + s = 1, \quad y \geq 0, \quad s \geq 0.
\]

Denote rows of $A$ by $a_i^T$. Then columns of matrix in CHMP are $v_i =
\begin{pmatrix}
a_i \\
b_i
\end{pmatrix}$, $i = 1, \ldots, m$ and $v_{m+1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, all in $\mathbb{R}^{n+1}$.

Suppose triangle algorithm for CHMP gives a witness $p' = \begin{pmatrix} x \\ \alpha \end{pmatrix}$. Then,
Test if $Ax < b$ is feasible, $A$ is an $m \times n$ matrix. (The problem Khachiyan considered in 1979).

$Ax < b$ is feasible if and only if the following CHMP is infeasible

$$
\begin{pmatrix}
A^T & 0 \\
b^T & 1
\end{pmatrix}
\begin{pmatrix}
y \\
s
\end{pmatrix} =
\begin{pmatrix}
0 \\
0
\end{pmatrix}, \quad \sum_{i=1}^{m} y_i + s = 1, \quad y \geq 0, \quad s \geq 0.
$$

Denote rows of $A$ by $a_i^T$. Then columns of matrix in CHMP are $v_i = (a_i, b_i), \ i = 1, \ldots, m$ and $v_{m+1} = (0, 1)$, all in $\mathbb{R}^{n+1}$.

Suppose triangle algorithm for CHMP gives a witness $p' = \begin{pmatrix} x \\ \alpha \end{pmatrix}$. Then,

$$
A(-x/\alpha) < b.
$$
Test if \( Ax < b \) is feasible, \( A \) is an \( m \times n \) matrix.
(The problem Khachiyan considered in 1979).
\( Ax < b \) is feasible if and only if the following CHMP is infeasible

\[
\begin{pmatrix}
A^T & 0 \\
b^T & 1
\end{pmatrix}
\begin{pmatrix}
y \\
s
\end{pmatrix}
=
\begin{pmatrix}
0 \\
0
\end{pmatrix},
\sum_{i=1}^{m} y_i + s = 1, \ y \geq 0, \ s \geq 0.
\]

Denote rows of \( A \) by \( a_i^T \). Then columns of matrix in CHMP are \( v_i = \begin{pmatrix} a_i \\ b_i \end{pmatrix}, i = 1, \ldots, m \) and \( v_{m+1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \), all in \( \mathbb{R}^{n+1} \).

Suppose triangle algorithm for CHMP gives a witness \( p' = \begin{pmatrix} x \\ \alpha \end{pmatrix} \). Then,

\[
A(-x/\alpha) < b.
\]

In other words, triangle algorithm gives complete answer when testing the feasibility of \( Ax < b \), not just a yes answer.
Nonstandard Application of Triangle Algorithm: Solving A Linear System

Consider solving $Ax = b$, with $A$ invertible.

Suppose it is known that $x = A^{-1}b \geq 0$.

We can apply the Triangle Algorithm to test if $0 \in \text{conv}(\begin{bmatrix} A \end{bmatrix}, -b)$.

The algorithm produces $\varepsilon$-approximate solution $\|Ax\| \leq \varepsilon \|b\|$.
Consider solving \( Ax = b \), with \( A \) invertible.
Consider solving $Ax = b$, with $A$ invertible. Suppose it is known that $x = A^{-1}b \geq 0$. 

Consider solving $Ax = b$, with $A$ invertible. Suppose it is known that $x = A^{-1}b \geq 0$. We can apply the Triangle Algorithm to test if

$$0 \in \text{conv}([A, -b]).$$
Consider solving $Ax = b$, with $A$ invertible. Suppose it is known that $x = A^{-1}b \geq 0$. We can apply the Triangle Algorithm to test if

$$0 \in \text{conv}(A, -b).$$

The algorithm produces $\varepsilon$-approximate solution

$$\|Ax_\varepsilon - b\| \leq \varepsilon \|b\|.$$
Incremental Triangle Algorithm: solving $Ax = b$

There exists $t^* \geq 0$ such that for any $t \geq t^*$ the solution of $A(x - te) = b$ is nonnegative ($e$ the vector of ones). Thus $0 \in \text{conv}(\{A, - (b + tu)\})$.

A convex hull problem is inherent to a linear system.

Incremental Triangle Algorithm: Given $\varepsilon$, and $t$ (initially zero), test if $0 \in \text{conv}(\{A, - (b + tu)\})$.

If $x_t = A^{-1}(b + tu) \geq 0$, Triangle Algorithm produces $x_\varepsilon$ satisfying $\|Ax_\varepsilon - b\| \leq \varepsilon \|b\|$.

Otherwise, by the distance duality, the algorithm computes a witness certifying that $x_t \not\geq 0$.

Using the witness, we increment $t$ and repeat.
There exists $t_* \geq 0$ such that for any $t \geq t_*$ the solution of $A(x - te) = b$ is nonnegative ($e$ the vector of ones). Thus
There exists $t_* \geq 0$ such that for any $t \geq t_*$ the solution of $A(x - te) = b$ is nonnegative ($e$ the vector of ones). Thus

$$0 \in \text{conv}(\left[ A, -(b + tu) \right]), \quad u = Ae.$$ 

A convex hull problem is inherent to a linear system.
There exists $t_\star \geq 0$ such that for any $t \geq t_\star$ the solution of $A(x - te) = b$ is nonnegative (e the vector of ones). Thus

$$0 \in \text{conv}([A, -(b + tu)]), \quad u = Ae.$$  

A convex hull problem is inherent to a linear system.

**Incremental Triangle Algorithm**: Given $\varepsilon$, and $t$ (initially zero), test if

$$0 \in \text{conv}([A, -(b + tu)]).$$
There exists $t^* \geq 0$ such that for any $t \geq t^*$ the solution of $A(x - te) = b$ is nonnegative ($e$ the vector of ones). Thus

$$0 \in \text{conv}([A, -(b + tu)]), \quad u = Ae.$$ 

A convex hull problem is inherent to a linear system.

**Incremental Triangle Algorithm**: Given $\varepsilon$, and $t$ (initially zero), test if

$$0 \in \text{conv}([A, -(b + tu)]).$$

If $x_t = A^{-1}(b + tu) \geq 0$, Triangle Algorithm produces $x_\varepsilon$ satisfying

$$\|Ax_\varepsilon - b\| \leq \varepsilon \|b\|.$$

Otherwise, by the *distance duality*, the algorithm computes a *witness* certifying that $x_t \not\geq 0$. 
There exists $t_* \geq 0$ such that for any $t \geq t_*$ the solution of $A(x - te) = b$ is nonnegative ($e$ the vector of ones). Thus

$$0 \in \text{conv}([A, -(b + tu)]), \quad u = Ae.$$  

A convex hull problem is inherent to a linear system.

**Incremental Triangle Algorithm:** Given $\varepsilon$, and $t$ (initially zero), test if

$$0 \in \text{conv}([A, -(b + tu)]).$$

If $x_t = A^{-1}(b + tu) \geq 0$, Triangle Algorithm produces $x_\varepsilon$ satisfying

$$\|Ax_\varepsilon - b\| \leq \varepsilon \|b\|.$$  

Otherwise, by the *distance duality*, the algorithm computes a *witness* certifying that $x_t \not\geq 0$. Using the witness, we increment $t$ and repeat.
Numerical Experiments for Solving $Ax = b$

In several experiments performed by DIMACS REU student, MS students, a Postdoc generating different systems, including those from finite difference discretization, Incremental Triangle Algorithm has outperformed Jacobi, Gauss-Seidel, SOR, and AOR, taking much fewer iterations than these methods.
In several experiments performed by DIMACS REU student, MS students, a Postdoc: generating different systems, including those from finite difference discretization, Incremental Triangle Algorithm has outperformed Jacobi, Gauss-Seidel, SOR, and AOR, taking much fewer iterations than these methods.
Nonstandard Application of Triangle Algorithm: Solving Google PageRank Matrix

The problem is solving $Ax = x$, where $x \geq 0$, $e^T x = 1$, for some square matrix $A$ with nonnegative entries, usually huge but sparse. Usually solved as an eigenvalue problem via the power method. Triangle Algorithm required fewer iterations than the power method. In some examples triangle algorithm used only one iteration to compute solutions to absolute accuracy $10^{-10}$. In particular, in an example (from Stanford) where the dimension of $A$ was approximately $300,000$. (Rutgers MS thesis of Hao Shen (2014-2015) includes details.)
The problem is solving $Ax = x$, where $x \geq 0$, $e^T x = 1$, for some square matrix $A$ with nonnegative entries, usually huge but sparse.
The problem is solving $Ax = x$, where $x \geq 0$, $e^T x = 1$, for some square matrix $A$ with nonnegative entries, usually huge but sparse.

Usually solved as an eigenvalue problem via the power method.
The problem is solving $Ax = x$, where $x \geq 0$, $e^T x = 1$, for some square matrix $A$ with nonnegative entries, usually huge but sparse.

Usually solved as an eigenvalue problem via the power method.

Triangle Algorithm required fewer iterations than the power method.
Nonstandard Application of Triangle Algorithm: Solving Google PageRank Matrix

The problem is solving $Ax = x$, where $x \geq 0$, $e^T x = 1$, for some square matrix $A$ with nonnegative entries, usually huge but sparse.

Usually solved as an eigenvalue problem via the power method.

Triangle Algorithm required fewer iterations than the power method.

In some examples triangle algorithm used only one iteration to compute solutions to absolute accuracy $10^{-10}$.
Nonstandard Application of Triangle Algorithm: Solving Google PageRank Matrix

The problem is solving $Ax = x$, where $x \geq 0$, $e^T x = 1$, for some square matrix $A$ with nonnegative entries, usually huge but sparse.

Usually solved as an eigenvalue problem via the power method.

Triangle Algorithm required fewer iterations than the power method.

In some examples triangle algorithm used only one iteration to compute solutions to absolute accuracy $10^{-10}$. In particular, in an example (from Stanford) where the dimension of $A$ was approximately 300,000. (Rutgers MS thesis of Hao Shen (2014-2015) includes details.)
Separation of Convex Sets

Definition

Given two compact convex subsets $K, K'$ of $\mathbb{R}^m$, we say $H = \{ x : h^T x = a \}$ is a separating hyperplane if $h^T x < a, \forall x \in K$, $h^T x < a, \forall x \in K'$.

Definition

$\delta^* = d(K, K') = \min \{ d(p, p') : p \in K, p' \in K' \} = d(p^*, p'^*)$.

Fact

Then $\delta^* = 0$ if and only if $K \cap K' \neq \emptyset$. 
**Definition**

Given two compact convex subsets $K, K'$ of $\mathbb{R}^m$, we say $H = \{ x : h^T x = a \}$ is a separating hyperplane if

$$h^T x < a, \quad \forall x \in K, \quad h^T x < a, \quad \forall x \in K'.$$
Definition

Given two compact convex subsets $K, K'$ of $\mathbb{R}^m$, we say $H = \{x : h^T x = a\}$ is a separating hyperplane if

$$h^T x < a, \quad \forall x \in K, \quad h^T x < a, \quad \forall x \in K'.$$

Definition

$$\delta_* = d(K, K') = \min \{d(p, p') : p \in K, p' \in K'\} = d(p_*, p_*').$$
Separation of Convex Sets

Definition

Given two compact convex subsets $K, K'$ of $\mathbb{R}^m$, we say $H = \{ x : h^T x = a \}$ is a separating hyperplane if

$$h^T x < a, \quad \forall x \in K, \quad h^T x < a, \quad \forall x \in K'.$$

Definition

$$\delta_* = d(K, K') = \min\{d(p, p') : p \in K, p' \in K'\} = d(p_*, p'_*).$$

Fact

Then $\delta_* = 0$ if and only if $K \cap K' \neq \emptyset$. 
Four Problems Associated to A Pair of Convex Sets

1. Test if $K$ and $K'$ intersect: Find $(p, p') \in K \times K'$ with $d(p, p')$ small.

2. If $K$ and $K'$ do not intersect:
   - Find a separating hyperplane
   - Estimate $\delta^* = d(K, K')$.
   - Find near-optimal pair of parallel supporting hyperplanes.

Figure: $(p^*, p'^*)$ optimal pair, $(H^*, H'^*)$ optimal support; $(p, p')$ a pair whose orthogonal bisector separator $H$; $(H_1, H'_1)$ a supporting pair.
(1) Test if \( K \) and \( K' \) intersect:
Four Problems Associated to A Pair of Convex Sets

(1) Test if $K$ and $K'$ intersect: Find $(p, p') \in K \times K'$ with $d(p, p')$ small.
(1) Test if $K$ and $K'$ intersect: Find $(p, p') \in K \times K'$ with $d(p, p')$ small. If $K$ and $K'$ do not intersect:
(1) Test if $K$ and $K'$ intersect: Find $(p, p') \in K \times K'$ with $d(p, p')$ small. If $K$ and $K'$ do not intersect:
(2) Find a separating hyperplane
Four Problems Associated to A Pair of Convex Sets

(1) Test if $K$ and $K'$ intersect: Find $(p, p') \in K \times K'$ with $d(p, p')$ small.

If $K$ and $K'$ do not intersect:
(2) Find a separating hyperplane
(3) Estimate $\delta_* = d(K, K')$. 

Figure: $(p_*, p'_*)$ optimal pair, $(H_*, H'_*)$ optimal support; $(p, p')$ a pair whose orthogonal bisector separator $(H_1, H'_1)$ a supporting pair.
(1) Test if $K$ and $K'$ intersect: Find $(p, p') \in K \times K'$ with $d(p, p')$ small. If $K$ and $K'$ do not intersect:
(2) Find a separating hyperplane
(3) Estimate $\delta_* = d(K, K')$.
(4) Find near-optimal pair of parallel supporting hyperplanes.
Four Problems Associated to A Pair of Convex Sets

(1) Test if $K$ and $K'$ intersect: Find $(p, p') \in K \times K'$ with $d(p, p')$ small. If $K$ and $K'$ do not intersect:
(2) Find a separating hyperplane
(3) Estimate $\delta_* = d(K, K')$.
(4) Find near-optimal pair of parallel supporting hyperplanes.

Figure: $(p_*, p'_*)$ optimal pair, $(H_*, H'_*)$ optimal support; $(p, p')$ a pair whose orthogonal bisector separator $H$; $(H_1, H'_1)$ a supporting pair.
Computing Approximate Intersection Point

Definition
Suppose $\delta^* = 0$. We say a pair $(p, p') \in \mathcal{K} \times \mathcal{K}'$ is an $\varepsilon$-approximation solution if $d(p, p') \leq \varepsilon d(p, v)$, for some $v \in \mathcal{K}$, or $d(p, p') \leq \varepsilon d(p', v')$, for some $v' \in \mathcal{K}'$.

Definition
Given $(p, p') \in \mathcal{K} \times \mathcal{K}'$, we say it is a witness pair if the orthogonal bisecting hyperplane of the line segment $pp'$ separates $\mathcal{K}$ and $\mathcal{K}'$. 

(Rutgers) Triangle Algorithm
July 26, 2016 37 / 57
Suppose $\delta_* = 0$. We say a pair $(p, p') \in K \times K'$ is an $\varepsilon$-approximation solution if $d(p, p') \leq \varepsilon d(p, v)$, for some $v \in K$, or $d(p, p') \leq \varepsilon d(p', v')$, for some $v' \in K'$. Given $(p, p') \in K \times K'$, we say it is a witness pair if the orthogonal bisecting hyperplane of the line segment $pp'$ separates $K$ and $K'$. 
Definition

Suppose \( \delta^* = 0 \). We say a pair \((p, p') \in K \times K'\) is an \( \varepsilon \)-approximation solution if

\[
\text{either } d(p, p') \leq \varepsilon d(p, v) \text{ or } d(p, p') \leq \varepsilon d(p', v'),
\]

for some \( v \in K \) and \( v' \in K' \).
Computing Approximate Intersection Point

Definition

Suppose \( \delta_* = 0 \). We say a pair \( (p, p') \in K \times K' \) is an \( \varepsilon \)-approximation solution if

\[
d(p, p') \leq \varepsilon d(p, v), \quad \text{for some} \quad v \in K,
\]

or

\[
d(p, p') \leq \varepsilon d(p', v'), \quad \text{for some} \quad v' \in K'.
\]
Suppose $\delta_\star = 0$. We say a pair $(p, p') \in K \times K'$ is an $\varepsilon$-approximation solution if

$$d(p, p') \leq \varepsilon d(p, v), \quad \text{for some } v \in K,$$

or

$$d(p, p') \leq \varepsilon d(p', v'), \quad \text{for some } v' \in K'.$$
Definition
Suppose $\delta_* = 0$. We say a pair $(p, p') \in K \times K'$ is an $\varepsilon$-approximation solution if
\[
d(p, p') \leq \varepsilon d(p, v), \quad \text{for some} \quad v \in K,
\]

or
\[
d(p, p') \leq \varepsilon d(p', v'), \quad \text{for some} \quad v' \in K'.
\]

Definition
Given $(p, p') \in K \times K'$, we say it is a witness pair if the orthogonal bisecting hyperplane of the line segment $pp'$ separates $K$ and $K'$. 
Triangle Algorithm I (Testing if $K$ and $K'$ intersect)

The algorithm computes $(p, p') \in K \times K'$ such that $d(p, p')$ is within a prescribed precision, or $d(p, p')$ is a witness pair.
Triangle Algorithm I (Testing if $K$ and $K'$ intersect)

The algorithm computes $(p, p') \in K \times K'$ such that $d(p, p')$ is within a prescribed precision, or $d(p, p')$ is a witness pair.
The algorithm computes \((p, p') \in K \times K'\) such that
The algorithm computes \((p, p') \in K \times K'\) such that 
\[d(p, p')\] is within a prescribed precision,
The algorithm computes \((p, p') \in K \times K'\) such that \(d(p, p')\) is within a prescribed precision,

or \(d(p, p')\) is a witness pair.
Pivot Points

Definition

Given a pair $(p, p') \in K \times K'$, we say $v \in K$ is a $p'$-pivot for $p$ if $d(p, v) \geq d(p', v)$.

We say $v' \in K'$ is a $p$-pivot for $p'$ if $d(p', v') \geq d(p, v')$.

Figure: $v$ is $p'$-pivot for $p$ (left); $v'$ is $p$-pivot for $p'$ (right).
Pivot Points

Definition

Given a pair \((p, p') \in K \times K'\),

we say \(v \in K\) is a \(p'\)-pivot for \(p\) if

\[d(p, v) \geq d(p', v)\]

We say \(v' \in K'\) is a \(p\)-pivot for \(p'\) if

\[d(p', v') \geq d(p, v')\]

Figure: \(v\) is \(p'\)-pivot for \(p\) (left); \(v'\) is \(p\)-pivot for \(p'\) (right).
Pivot Points

Definition

Given a pair \((p, p') \in K \times K'\), we say \(v \in K\) is a \(p'\)-pivot for \(p\) if

\[
d(p, v) \geq d(p', v).
\]
Pivot Points

Definition

Given a pair \((p, p') \in K \times K'\), we say \(v \in K\) is a \(p'\)-pivot for \(p\) if

\[ d(p, v) \geq d(p', v). \]

We say \(v' \in K'\) is a \(p\)-pivot for \(p'\) if

\[ d(p', v') \geq d(p, v'). \]
Pivot Points

Definition

Given a pair \((p, p') \in K \times K'\), we say \(v \in K\) is a \(p'\)-pivot for \(p\) if

\[ d(p, v) \geq d(p', v). \]

We say \(v' \in K'\) is a \(p\)-pivot for \(p'\) if

\[ d(p', v') \geq d(p, v'). \]

Figure: \(v\) is \(p'\)-pivot for \(p\) (left); \(v'\) is \(p\)-pivot for \(p'\) (right).
Consider the Voronoi diagram of the two points set \( \{ p, p' \} \), \((p, p') \in K \times K'\) and the corresponding Voronoi cells:

\[
V(p) = \{ x : d(x, p) < d(x, p') \}, \\
V(p') = \{ x : d(x, p') < d(x, p) \}.
\]

Let \( H \) be the orthogonal bisecting hyperplane of the line \( pp' \).

\( H \) intersects \( K \) \iff there exists \( v \in K \) that is a \( p' \)-pivot for \( p \),

\( H \) intersects \( K' \) \iff there exists \( v' \in K' \) that is a \( p \)-pivot for \( p' \).

Figure: In the Figure, the point \( v \) and \( v' \) are pivots for \( p' \) and \( p \), respectively.
Consider the Voronoi diagram of the two points set \( \{p, p'\} \), \((p, p') \in K \times K'\) and the corresponding Voronoi cells:

\[
V(p) = \{ x : d(x, p) < d(x, p') \}, \\
V(p') = \{ x : d(x, p') < d(x, p) \}.
\]

Let \( H \) be the orthogonal bisecting hyperplane of the line \( pp' \).

\( H \) intersects \( K \) \iff \text{there exists } v \in K \text{ that is a } p' \text{-pivot for } p, \\
\( H \) intersects \( K' \) \iff \text{there exists } v' \in K' \text{ that is a } p \text{-pivot for } p'.

Figure: In the Figure, the point \( v \) and \( v' \) are pivots for \( p' \) and \( p \), respectively.
Consider the Voronoi diagram of the two points set \( \{p, p'\} \), \((p, p') \in K \times K'\) and the corresponding Voronoi cells:

\[
V(p) = \{x : d(x, p) < d(x, p')\}, \quad V(p') = \{x : d(x, p') < d(x, p)\}.
\]
Consider the Voronoi diagram of the two points set \( \{p, p'\} \), \((p, p') \in K \times K' \) and the corresponding Voronoi cells:

\[
V(p) = \{x : d(x, p) < d(x, p')\}, \quad V(p') = \{x : d(x, p') < d(x, p)\}.
\]

Let \( H \) be the orthogonal bisecting hyperplane of the line \( pp' \).
Consider the Voronoi diagram of the two points set \( \{p, p'\} \), 
\((p, p') \in K \times K' \) and the corresponding Voronoi cells:
\[ V(p) = \{x : d(x, p) < d(x, p')\}, \quad V(p') = \{x : d(x, p') < d(x, p)\}. \]
Let \( H \) be the orthogonal bisecting hyperplane of the line \( pp' \).

\( H \) intersects \( K \) \( \iff \) there exists \( v \in K \) that is a \( p' \)-pivot for \( p \),
Consider the Voronoi diagram of the two points set \( \{p, p'\} \), \((p, p') \in K \times K'\) and the corresponding Voronoi cells:

\[
V(p) = \{x : d(x, p) < d(x, p')\}, \quad V(p') = \{x : d(x, p') < d(x, p)\}.
\]

Let \( H \) be the orthogonal bisecting hyperplane of the line \( pp' \).

\( H \) intersects \( K \) \iff there exists \( v \in K \) that is a \( p' \)-pivot for \( p \),

\( H \) intersects \( K' \) \iff there exists \( v' \in K' \) that is a \( p \)-pivot for \( p' \).
Consider the Voronoi diagram of the two points set \( \{p, p'\} \), \((p, p') \in K \times K'\) and the corresponding Voronoi cells:

\[
V(p) = \{ x : d(x, p) < d(x, p') \}, \quad V(p') = \{ x : d(x, p') < d(x, p) \}.
\]

Let \( H \) be the orthogonal bisecting hyperplane of the line \( pp' \).

\( H \) intersects \( K \iff \) there exists \( v \in K \) that is a \( p' \)-pivot for \( p \),

\( H \) intersects \( K' \iff \) there exists \( v' \in K' \) that is a \( p \)-pivot for \( p' \).

![Voronoi Diagram](image)

**Figure:** In the Figure, the point \( v \) and \( v' \) are pivots for \( p' \) and \( p \), respectively.
Theorem (Krein-Milman)

Let $K$ be a compact convex subset of $\mathbb{R}^m$. Then $K$ is the convex hull of its extreme points. In notation, $K = \text{conv}(\text{ex}(K))$. 

(Rutgers) 

Triangle Algorithm

July 26, 2016 41 / 57
Theorem (Krein-Milman) Let $K$ be a compact convex subset of $\mathbb{R}^m$. Then $K$ is the convex hull of its extreme points. In notation, $K = \text{conv}(\text{ex}(K))$. 
A New Separating Hyperplane Theorem

Theorem (Distance Duality) Let $K, K'$ be compact convex subsets in $\mathbb{R}^m$, with $\text{ex}(K)$ and $\text{ex}(K')$ as their corresponding set of extreme points. Let $S$ be a subset of $K$ containing $\text{ex}(K)$, and $S'$ a subset of $K'$ containing $\text{ex}(K')$. Then, $K \cap K' \neq \emptyset$ if and only if for each $(p, p') \in K \times K'$, either there exists $v \in S$ such that $d(p', v) \geq d(p, v)$, or there exists $v' \in S'$ such that $d(p, v') \geq d(p', v')$.

An alternative description of the Distance Duality is the following. Theorem (Distance Duality) Let $K, K'$ be compact convex subsets in $\mathbb{R}^m$, with $\text{ex}(K)$ and $\text{ex}(K')$ as their corresponding set of extreme points. Then, $K \cap K' = \emptyset$ if and only if there exists $(p, p') \in K \times K'$ such that $d(p, v) < d(p', v)$ for all $v \in \text{ex}(K)$ and $d(p', v') < d(p, v')$ for all $v' \in \text{ex}(K')$. (Such pair is necessarily a witness pair).
Theorem

(Distance Duality) Let $K, K'$ be compact convex subsets in $\mathbb{R}^m$, with $\text{ex}(K)$ and $\text{ex}(K')$ as their corresponding set of extreme points. Let $S$ be a subset of $K$ containing $\text{ex}(K)$, and $S'$ a subset of $K'$ containing $\text{ex}(K')$. Then, $K \cap K' \neq \emptyset$ if and only if for each $(p, p') \in K \times K'$, either there exists $v \in S$ such that $d(p', v) \geq d(p, v)$, or there exists $v' \in S'$ such that $d(p, v') \geq d(p', v')$. (Such pair is necessarily a witness pair)
Theorem

(Distance Duality) Let $K, K'$ be compact convex subsets in $\mathbb{R}^m$, with $\text{ex}(K)$ and $\text{ex}(K')$ as their corresponding set of extreme points. Let $S$ be a subset of $K$ containing $\text{ex}(K)$, and $S'$ a subset of $K'$ containing $\text{ex}(K')$. Then, $K \cap K' \neq \emptyset$ if and only if for each $(p, p') \in K \times K'$, either there exists $v \in S$ such that $d(p', v) \geq d(p, v)$, or there exists $v' \in S'$ such that $d(p, v') \geq d(p', v')$.

An alternative description of the Distance Duality is the following.

Theorem

(Distance Duality) Let $K, K'$ be compact convex subsets in $\mathbb{R}^m$, with $\text{ex}(K)$ and $\text{ex}(K')$ as their corresponding set of extreme points. Then, $K \cap K' = \emptyset$ if and only if there exists $(p, p') \in K \times K'$ such that $d(p, v) < d(p', v)$ for all $v \in \text{ex}(K)$ and $d(p', v') < d(p, v')$ for all $v' \in \text{ex}(K')$. (Such pair is necessarily a witness pair)
Iterative Step in Triangle Algorithm I

Each iteration of Triangle Algorithm I computes for given pair \((p, p') \in K \times K'\), either \(v \in K\) that is a \(p'\)-pivot for \(p\); or \(v' \in K'\), a \(p\)-pivot for \(p'\).

These are equivalent to checking if
\[
2v^T(p'-p) \geq \|p'\|^2 - \|p\|^2,
\]
\[
2v'^T(p-p') \geq \|p\|^2 - \|p'\|^2.
\]

These can be computed by solving the convex programs:
\[
\max \{ (p'-p)^T v : v \in K \},
\]
\[
\max \{ (p-p')^T v' : v' \in K' \}.
\]

Let \(T_K, T_{K'}\) be the arithmetic complexities of solving these problems, respectively.

Thus the worst-case complexity in each iteration is
\[
T = \max \{ T_K, T_{K'} \}.
\]
Each iteration of Triangle Algorithm I computes for given pair \((p, p') \in K \times K'\), either \(v \in K\) that is a \(p'\)-pivot for \(p\); or \(v' \in K'\), a \(p\)-pivot for \(p'\).
Each iteration of Triangle Algorithm I computes for given pair \((p, p') \in K \times K'\), either \(v \in K\) that is a \(p'\)-pivot for \(p\); or \(v' \in K'\), a \(p\)-pivot for \(p'\). These are equivalent to checking if

\[
\begin{align*}
2v^T(p' - p) &\geq \|p'\|^2 - \|p\|^2, \\
2v'^T(p - p') &\geq \|p\|^2 - \|p'\|^2.
\end{align*}
\]
Each iteration of Triangle Algorithm I computes for given pair \((p, p') \in K \times K'\), either \(v \in K\) that is a \(p'\)-pivot for \(p\); or \(v' \in K'\), a \(p\)-pivot for \(p'\). These are equivalent to checking if

\[
2v^T(p' - p) \geq \|p'\|^2 - \|p\|^2, \quad 2v'^T(p - p') \geq \|p\|^2 - \|p'\|^2.
\]
Each iteration of Triangle Algorithm I computes for given pair \((p, p') \in K \times K'\), either \(v \in K\) that is a \(p'\)-pivot for \(p\); or \(v' \in K'\), a \(p\)-pivot for \(p'\). These are equivalent to checking if

\[
2v^T(p' - p) \geq \|p'\|^2 - \|p\|^2, \quad 2v'^T(p - p') \geq \|p\|^2 - \|p'\|^2.
\]

These can be computed by solving the convex programs:

\[
\max \{v^T(p' - p) : v \in K\}, \quad \max \{v'^T(p - p') : v' \in K'\}.
\]
Each iteration of Triangle Algorithm I computes for given pair \((p, p') \in K \times K'\), either \(v \in K\) that is a \(p'\)-pivot for \(p\); or \(v' \in K'\), a \(p\)-pivot for \(p'\). These are equivalent to checking if

\[
2v^T(p' - p) \geq \|p'\|^2 - \|p\|^2, \quad 2v'^T(p - p') \geq \|p\|^2 - \|p'\|^2.
\]

These can be computed by solving the convex programs:

\[
\max\{(p' - p)^Tv : v \in K\}, \quad \max\{(p - p')^Tv' : v' \in K'\}.
\]
Each iteration of Triangle Algorithm I computes for given pair \((p, p') \in K \times K'\), either \(v \in K\) that is a \(p'\)-pivot for \(p\); or \(v' \in K'\), a \(p\)-pivot for \(p'\). These are equivalent to checking if

\[
2v^T(p' - p) \geq \|p'\|^2 - \|p\|^2, \quad 2v'^T(p - p') \geq \|p\|^2 - \|p'\|^2.
\]

These can be computed by solving the convex programs:

\[
\max \{ (p' - p)^T v : v \in K \}, \quad \max \{ (p - p')^T v' : v' \in K' \}.
\]

Let \(T_K, T_{K'}\) be the arithmetic complexities of solving these problems, respectively.
Each iteration of Triangle Algorithm I computes for given pair \((p, p') \in K \times K'\), either \(v \in K\) that is a \(p'\)-pivot for \(p\); or \(v' \in K'\), a \(p\)-pivot for \(p'\). These are equivalent to checking if

\[
2v^T (p' - p) \geq \|p'\|^2 - \|p\|^2, \quad 2v'^T (p - p') \geq \|p\|^2 - \|p'\|^2.
\]

These can be computed by solving the convex programs:

\[
\max \{(p' - p)^T v : v \in K\}, \quad \max \{(p - p')^T v' : v' \in K'\}.
\]

Let \(T_K, T_{K'}\) be the arithmetic complexities of solving these problems, respectively. Thus the worst-case complexity in each iteration is

\[
T = \max \{T_K, T_{K'}\}.
\]
Triangle Algorithm I

Triangle Algorithm I \((p_0, p'_0) \in K \times K', \varepsilon \in (0, 1)\)

Step 0. Set \(p = v = p_0, p' = v' = p'_0\).

Step 1. If \(d(p, p') \leq \varepsilon d(p, v)\), or \(d(p, p') \leq \varepsilon d(p', v')\), stop.

Step 2. Test if there exists \(v \in K\) that is a \(p'\)-pivot for \(p\), i.e.
\[
2v^T(p' - p) \geq \|p'\|^2 - \|p\|^2
\]
(e.g. by solving \(\max \{ (p' - p)^T v : v \in K\}\)). If such pivot exists, set \(p \leftarrow \) nearest \((p'; pv)\), and go to Step 1.

Step 3. Test if there exists \(v' \in K'\) that is a \(p\)-pivot for \(p'\), i.e.
\[
2v'^T(p - p') \geq \|p\|^2 - \|p'\|^2
\]
(e.g. by solving \(\max \{ (p - p')^T v' : v' \in K'\}\)). If such pivot exists, set \(p' \leftarrow \) nearest \((p; p'v')\), and go to Step 1.

Step 4. Output \((p, p')\) as a witness pair, stop \((K \cap K' = \emptyset)\).
Triangle Algorithm I \(( (p_0, p'_0) \in K \times K', \varepsilon \in (0, 1) )\)

**Step 0.** Set \( p = v = p_0, p' = v' = p'_0 \). 

**Step 1.** If \( d(p, p') \leq \varepsilon d(p, v) \), or \( d(p, p') \leq \varepsilon d(p', v') \), stop. 

**Step 2.** Test if there exists \( v \in K \) that is a \( p' \)-pivot for \( p \), i.e.

\[
2v^T(p' - p) \geq \|p'\|^2 - \|p\|^2
\]

(e.g. by solving \( \max \{(p' - p)^T v : v \in K\} \)). If such pivot exists, set \( p \leftarrow \text{nearest}(p'; pv) \), and go to Step 1.

**Step 3.** Test if there exists \( v' \in K' \) that is a \( p \)-pivot for \( p' \), i.e.

\[
2v'^T(p - p') \geq \|p\|^2 - \|p'\|^2.
\]

(e.g. by solving \( \max \{(p - p')^T v' : v' \in K'\} \)). If such pivot exists, set \( p' \leftarrow \text{nearest}(p; p'v') \), and go to Step 1.

**Step 4.** Output \( (p, p') \) as a witness pair, stop \((K \cap K' = \emptyset)\).
When $\delta^* = 0$, the number of iterations to get $\epsilon$-approximate solution is $O\left(\frac{1}{\epsilon^2}\right)$.

When $\delta^* > 0$, the number of iterations of Triangle Algorithm I to compute a witness pair $(p, p') \in K \times K'$ is $O\left((\rho^* \delta^*)^2\right)$, where

$\rho^* = \max\{\Delta_0, \Delta_0'\}$,

$\Delta_0 = \text{diam}(K)$,

$\Delta_0' = \text{diam}(K')$. 

(Rutgers)
When $\delta_* = 0$, the number of iterations to get $\varepsilon$-approximate solution is $O\left(\frac{1}{\varepsilon^2}\right)$.
When $\delta_* = 0$, the number of iterations to get $\varepsilon$-approximate solution is

$$O\left(\frac{1}{\varepsilon^2}\right).$$

When $\delta_* > 0$, the number of iterations of Triangle Algorithm I to compute a witness pair $(p, p') \in K \times K'$ is

$$O\left(\left(\frac{\rho_*}{\delta_*}\right)^2\right),$$

where $\rho_* = \max\{\Delta_0, \Delta'_0\}$, $\Delta_0 = \text{diam}(K)$, and $\Delta'_0 = \text{diam}(K')$. 

Testing Separation of Convex Sets

Definition
Suppose $\delta^* > 0$. We say a witness pair $(p, p') \in K \times K'$ is an $\varepsilon$-approximation solution if $d(p, p') - \delta^* \leq \varepsilon d(p, v)$ for some $v \in K$, or $d(p, p') - \delta^* \leq \varepsilon d(p', v')$ for some $v' \in K'$.

Definition
Suppose $\delta^* > 0$. We say a pair of hyperplanes $(H, H')$ supports $(K, K')$, if they are parallel, $H$ supports $K$ and $H'$ supports $K'$. 

(Rutgers)
Definition

Suppose \( \delta_* > 0 \).

We say a witness pair \((p, p') \in K \times K'\) is an \( \varepsilon \)-approximation solution if

\[
d(p, p') - \delta_* \leq \varepsilon d(p, v),
\]
for some \( v \in K \), or

\[
d(p, p') - \delta_* \leq \varepsilon d(p', v'),
\]
for some \( v' \in K' \).
Suppose $\delta_* > 0$. We say a witness pair $(p, p') \in K \times K'$ is an $\varepsilon$-approximation solution if

$$d(p, p') - \delta_* \leq \varepsilon d(p, v), \quad \text{for some} \quad v \in K,$$

or

$$d(p, p') - \delta_* \leq \varepsilon d(p', v'), \quad \text{for some} \quad v' \in K'.$$
Testing Separation of Convex Sets

Definition
Suppose $\delta_* > 0$. We say a witness pair $(p, p') \in K \times K'$ is an $\varepsilon$-approximation solution if

$$d(p, p') - \delta_* \leq \varepsilon d(p, v), \quad \text{for some} \quad v \in K,$$

or

$$d(p, p') - \delta_* \leq \varepsilon d(p', v'), \quad \text{for some} \quad v' \in K'.$$

Definition
Suppose $\delta_* > 0$. We say a pair of hyperplanes $(H, H')$ supports $(K, K')$, if they are parallel, $H$ supports $K$ and and $H'$ supports $K'$. 
Definition
Suppose $\delta^* > 0$. We say a witness pair $(p, p') \in K \times K'$ gives an $\varepsilon$-approximate supporting hyperplane, if it is an $\varepsilon$-approximate solution and there exists a pair of supporting hyperplane $(H, H')$, parallel to the orthogonal bisecting hyperplane of $(p, p')$, satisfying

$$
\delta^* - d(H, H') \leq \varepsilon d(p, v),
$$

or

$$
\delta^* - d(H, H') \leq \varepsilon d(p', v'),
$$

for some $v \in K$ or $v' \in K'$. 

(Rutgers)
Definition

Suppose $\delta_* > 0$. We say a witness pair $(p, p') \in K \times K'$ gives an $\varepsilon$-approximate supporting hyperplane, if it is an $\varepsilon$-approximate solution and there exists a pair of supporting hyperplanes $(H, H')$, parallel to the orthogonal bisecting hyperplane of $(p, p')$, satisfying

$$\delta_* - d(H, H') \leq \varepsilon d(p, v),$$

for some $v \in K$, or

$$\delta_* - d(H, H') \leq \varepsilon d(p', v'),$$

for some $v' \in K'$. 

(Rutgers)
Definition
Suppose $\delta_\ast > 0$. We say a witness pair $(p, p') \in K \times K'$ gives an $\varepsilon$-approximate supporting hyperplane, if it is an $\varepsilon$-approximate solution and there exists a pair or supporting hyperplane $(H, H')$, parallel to the orthogonal bisecting hyperplane of $(p, p')$, satisfying

$$\delta_\ast - d(H, H') \leq \varepsilon d(p, v), \quad \text{for some} \quad v \in K,$$

or

$$\delta_\ast - d(H, H') \leq \varepsilon d(p', v'), \quad \text{for some} \quad v' \in K'.$$
Triangle Algorithm II (Start With a Witness Pair)

Given a witness pair \((p, p') \in K \times K'\), it computes an \(\varepsilon\)-approximate solution, i.e. such that \(d(p, p')\) approximates \(\delta^* = d(K, K')\).

Since \((p, p')\) is a witness-pair, there is no pivot for \(p\), or a pivot for \(p'\). However, if \(d(p, p')\) does not sufficiently approximate \(\delta^*\), we will make use of weak-pivot.
Given a witness pair \((p, p') \in K \times K'\), it computes an \(\varepsilon\)-approximate solution, i.e. such that \(d(p, p')\) approximates \(\delta_* = d(K, K')\).
Given a witness pair \((p, p') \in K \times K'\), it computes an \(\varepsilon\)-approximate solution, i.e. such that \(d(p, p')\) approximates \(\delta_* = d(K, K')\).

Since \((p, p')\) is a witness-pair, there is no pivot for \(p\), or a pivot for \(p'\).
Given a witness pair \((p, p') \in K \times K'\), it computes an \(\varepsilon\)-approximate solution, i.e. such that \(d(p, p')\) approximates \(\delta_* = d(K, K')\).

Since \((p, p')\) is a witness-pair, there is no pivot for \(p\), or a pivot for \(p'\).

However, if \(d(p, p')\) does not sufficiently approximate \(\delta_*\), we will make use of weak-pivot, to defined.
Algorithm for Approximation of Distance

Figure: Depiction of the orthogonal bisector hyperplane $H$ to $pp'$ and parallel supporting hyperplanes $Hv$ and $Hv'$ that separate $K$ and $K'$. 

$$\delta v + \delta v' = d(Hv, Hv') < \delta^* < d(p, p').$$
Figure: Depiction of the orthogonal bisector hyperplane $H$ to $pp'$ and parallel supporting hyperplanes $H_v$ and $H_{v'}$ that separate $K$ and $K'$.

\[ \delta_v + \delta_{v'} = d(H_v, H_{v'}) < \delta_* < d(p, p'). \]
Suppose \((p, p') \in K \times K'\) is a witness pair. Let the orthogonal bisecting hyperplane to the line \(pp'\) be \(H = \{x : h^T x = (p - p')^T x = a\}\). Let \(v = \arg\min \{h^T x : x \in K\}\), \(v' = \arg\max \{h^T x : x \in K'\}\), \(H_v = \{x : h^T x = h^T v\}\), \(H_{v'} = \{x : h^T x = h^T v'\}\). Then \(H_v\) and \(H_{v'}\) are supporting hyperplane to \(K\) and \(K'\), respectively.

Also, if \(\delta_v = d(v, H)\), \(\delta_{v'} = d(v', H)\), \(\delta = \delta_v + \delta_{v'}\), we have \(d(H_v, H_{v'}) = \delta \leq \delta^* \leq \delta = d(p, p')\).
Theorem

Suppose \((p, p') \in K \times K'\) is a witness pair. Let the orthogonal bisecting hyperplane to the line \(pp'\) be \(H = \{x : h^T x = (p - p')^T x = a\}\). Let \(v = \text{argmin}\{h^T x : x \in K\}\), \(v' = \text{argmax}\{h^T x : x \in K'\}\),

\[
H_v = \{x : h^T x = h^T v\}, \quad H_{v'} = \{x : h^T x = h^T v'\}.
\]

Then \(H_v\) and \(H_{v'}\) are supporting hyperplane to \(K\) and \(K'\), respectively.

Also, if \(\delta_v = d(v, H)\), \(\delta_{v'} = d(v', H)\), \(\bar{\delta} = \delta_v + \delta_{v'}\), we have

\[
d(H_v, H_{v'}) = \bar{\delta} = \frac{h^T v - h^T v'}{\|h\|},
\]

\[
\bar{\delta} \leq \delta_* \leq \delta = d(p, p').
\]
Definition

Given a witness pair \((p, p') \in K \times K\), let \(H\) be the orthogonal bisecting hyperplane of \(pp'\). We shall say \(v \in K\) is a weak \(p'\)-pivot for \(p\) if \(d(p, H) > d(v, H)\).

Similarly, we shall say \(v' \in K'\) is a weak \(p\)-pivot for \(p'\) if \(d(p', H) > d(v', H)\).
Definition

Given a witness pair \((p, p') \in K \times K'\), let \(H\) be the orthogonal bisecting hyperplane of \(pp'\). We shall say \(v \in K\) is a \textit{weak} \(p'\)-pivot for \(p\) if

\[
d(p, H) > d(v, H).
\]

Similarly, we shall say \(v' \in K'\) is a \textit{weak} \(p\)-pivot for \(p'\) if

\[
d(p', H) > d(v', H).
\]
Definition

Given a witness pair \((p, p') \in K \times K'\), let \(H\) be the orthogonal bisecting hyperplane of \(pp'\). We shall say \(v \in K\) is a weak \(p'\)-pivot for \(p\) if

\[
d(p, H) > d(v, H).
\]

Similarly, we shall say \(v' \in K'\) is a weak \(p\)-pivot for \(p'\) if

\[
d(p', H) > d(v', H).
\]
Theorem

Let \( \Delta_0 = \text{diam}(K) \), \( \Delta_0' = \text{diam}(K') \), \( \rho^* = \max\{\Delta_0, \Delta_0'\} \).

The total arithmetic complexity of Triangle Algorithm II is \( O(T(\rho^* \delta^* \varepsilon))^2 \ln \rho^* \delta^* \varepsilon) \).

In particular, when \( K \) or \( K' \) is a singleton we have \( O(T(\rho^* \delta^* \varepsilon))^2) \).
Theorem

Let

\[ \Delta_0 = \text{diam}(K), \quad \Delta'_0 = \text{diam}(K'), \]
\[ \rho_* = \max\{\Delta_0, \Delta'_0\}. \]

The total arithmetic complexity of Triangle Algorithm II is

\[ O\left( T\left( \frac{\rho_*}{\delta_* \varepsilon} \right)^2 \ln \frac{\rho_*}{\delta_*} \right). \]

In particular, when \( K \) or \( K' \) is a singleton we have

\[ O\left( T\left( \frac{\rho_*}{\delta_* \varepsilon} \right)^2 \right). \]
Triangle Algorithm II

Triangle Algorithm II begins with a witness pair \((p_0, p'_0)\). However, in subsequent iterations the pair \((p_k, p'_k)\) may or may not be a witness pair. Thus, the algorithm requires searching for a weak-pivot or a pivot to reduce the gap \(\delta_k = d(p_k, p'_k)\) until the desired approximation is attained.
Triangle Algorithm II begins with a witness pair \((p_0, p'_0)\). However, in subsequent iterations the pair \((p_k, p'_k) \in K \times K'\) may or may not be a witness pair.
Triangle Algorithm II begins with a witness pair \((p_0, p_0')\). However, in subsequent iterations the pair \((p_k, p_k') \in K \times K'\) may or many not be a witness pair. Thus, the algorithm requires searching for a weak-pivot or a pivot to reduce the gap \(\delta_k = d(p_k, p_k')\) until the desired approximation is attained.
Let $T$ be the worst-case complexity of computing a pivot for a point in $K$, or $K'$. The total number of arithmetic operations in Triangle Algorithm I to get an $\varepsilon$-approximate solution when $\delta^* = 0$, or a witness pair is $O(T\varepsilon^2)$.

The total number of arithmetic operations in Triangle Algorithm II to get an $\varepsilon$-approximate solution to $\delta^*$ is $O(T(\rho^*\delta^*)^2)$.

And to get an $\varepsilon$-approximate supporting hyperplane is $O(T(\rho^*\delta^*)^2\ln\rho^*\delta^*)$.
Let $T$ be the worst-case complexity of computing a pivot for a point in $K$, or $K'$. The total number of arithmetic operations in Triangle Algorithm I to get an $\varepsilon$-approximate solution when $\delta_* = 0$, or a witness pair is

$$O\left(\frac{T}{\varepsilon^2}\right).$$
Let $T$ be the worst-case complexity of computing a pivot for a point in $K$, or $K'$. The total number of arithmetic operations in Triangle Algorithm I to get an $\varepsilon$-approximate solution when $\delta_*=0$, or a witness pair is

$$O\left(\frac{T}{\varepsilon^2}\right).$$

The total number of arithmetic operations in Triangle Algorithm II to get an $\varepsilon$-approximate solution to $\delta_*$ is
Let $T$ be the worst-case complexity of computing a pivot for a point in $K$, or $K'$. The total number of arithmetic operations in Triangle Algorithm I to get an $\varepsilon$-approximate solution when $\delta_* = 0$, or a witness pair is

$$O\left(\frac{T}{\varepsilon^2}\right).$$

The total number of arithmetic operations in Triangle Algorithm II to get an $\varepsilon$-approximate solution to $\delta_*$ is

$$O\left(\frac{T(\rho'_*)}{\delta_*^{\frac{1}{2}}}\right)^2.$$

And to get and $\varepsilon$-approximate supporting hyperplane is
Summary of Triangle Algorithms I and II

Let $T$ be the worst-case complexity of computing a pivot for a point in $K$, or $K'$. The total number of arithmetic operations in Triangle Algorithm I to get an $\varepsilon$-approximate solution when $\delta_* = 0$, or a witness pair is

$$O\left( T \frac{1}{\varepsilon^2} \right).$$

The total number of arithmetic operations in Triangle Algorithm II to get an $\varepsilon$-approximate solution to $\delta_*$ is

$$O\left( T \left( \frac{\rho_*}{\delta_*} \right)^2 \right).$$

And to get and $\varepsilon$-approximate supporting hyperplane is

$$O\left( T \left( \frac{\rho_*}{\delta_*} \frac{1}{\varepsilon} \right)^2 \ln \frac{\rho_*}{\delta_*} \right).$$
Special Applications and Extensions

When $K = \text{conv}(V)$, $V = \{v_1, \ldots, v_n\}$, $K' = \text{conv}(V')$, $V' = \{v'_1, \ldots, v'_n\}$. In particular, when one set is a single point.

This includes applications such as SVM. In this case $T = O(m(n + n'))$,

with preprocessing $T = O(m + \max\{n, n'\})$.

CS Masters Thesis, Mayank Gupta, 2015-2016, extensive computation and comparison with sequential minimal optimization (SMO). The results are very good! Article to be released in near future.

Applications in non-convex optimization.

Applications in combinatorial and graph problems.

Applications in conic programming.
When $K = \text{conv}(V)$, $V = \{v_1, \ldots, v_n\}$, $K' = \text{conv}(V')$, $V' = \{v'_1, \ldots, v'_{n'}\}$). In particular, when one set is a single point. This includes applications such as SVM. In this case

\[ T = O(m(n + n')) \]

with preprocessing \[ T = O(m + \max\{n, n'\}) \].

CS Masters Thesis, Mayank Gupta, 2015-1016, extensive computation and comparison with sequential minimal optimization (SMO). The results are very good! Article to be released in near future.
When $K = \text{conv}(V)$, $V = \{v_1, \ldots, v_n\}$, $K' = \text{conv}(V')$, $V' = \{v'_1, \ldots, v'_{n'}\}$). In particular, when one set is a single point. This includes applications such as SVM. In this case

$$T = O(m(n + n')),$$ 

with preprocessing

$$T = O(m + \max\{n, n'\}).$$

CS Masters Thesis, Mayank Gupta, 2015-1016, extensive computation and comparison with sequential minimal optimization (SMO). The results are very good! Article to be released in near future.

- Applications in non-convex optimization.
- Applications in combinatorial and graph problems.
- Applications in conic programming.
Approximation of An NP-Complete Problem

Decision Problem: Given a symmetric $n \times n$ matrix $Q$, does there exist $x \in S = \{x: e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?

This problem is NP-complete.

Let $Z = \{x: x^T Q x = 0: x^T x \leq 1\}$.

Let $K = \text{conv}(Z)$.

Let $K' = S = \{x: e^T x = 1, x \geq 0\}$.

Using the algorithmic separating hyperplane theorem in the corresponding Triangle Algorithm, we can give a fully polynomial-time approximation scheme to either separate $S$ from $\text{conv}(Z)$, hence proving that either $Z \cap S$ is empty, or to give an approximate point in $\text{conv}(Z) \cap S$.

In particular, in the later case when $Z$ is convex, the algorithm gives an approximate zero of $Q$ in $S$. 

(Rutgers) Triangle Algorithm July 26, 2016 56 / 57
**Decision Problem:** Given a symmetric $n \times n$ matrix $Q$, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?
**Decision Problem:** Given a symmetric $n \times n$ matrix $Q$, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?

This problem is NP-complete.
**Decision Problem:** Given a symmetric $n \times n$ matrix $Q$, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?

This problem is NP-complete.

Let $Z = \{x : x^T Q x = 0 : x^T x \leq 1\}$. 

Let $K = \text{conv} (Z)$. 

Let $K' = S = \{x : e^T x = 1, x \geq 0\}$.

Using the algorithmic separating hyperplane theorem in the corresponding Triangle Algorithm, we can give a fully polynomial-time approximation scheme to either separate $S$ from $\text{conv} (Z)$, hence proving that either $Z \cap S$ is empty, or to give an approximate point in $\text{conv} (Z) \cap S$.

In particular, in the later case when $Z$ is convex, the algorithm gives an approximate zero of $Q$ in $S$. 

(Rutgers) Triangle Algorithm July 26, 2016 56 / 57
Decision Problem: Given a symmetric $n \times n$ matrix $Q$, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?

This problem is NP-complete.

Let $Z = \{x : x^T Q x = 0 : x^T x \leq 1\}$.

Let $K = \text{conv}(Z)$. 

**Decision Problem:** Given a symmetric $n \times n$ matrix $Q$, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?

This problem is NP-complete.

Let $Z = \{x : x^T Q x = 0 : x^T x \leq 1\}$.

Let $K = \text{conv}(Z)$.

Let $K' = S = \{x : e^T x = 1, x \geq 0\}$.
**Decision Problem:** Given a symmetric $n \times n$ matrix $Q$, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?

This problem is NP-complete.

Let $Z = \{x : x^T Q x = 0 : x^T x \leq 1\}$.

Let $K = \text{conv}(Z)$.

Let $K' = S = \{x : e^T x = 1, x \geq 0\}$.

Using the algorithmic separating hyperplane theorem in the corresponding Triangle Algorithm, we can give a fully polynomial-time approximation scheme to either separate $S$ from $\text{conv}(Z)$, hence proving that either $Z \cap S$ is empty,
Decision Problem: Given a symmetric $n \times n$ matrix $Q$, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?

This problem is NP-complete.

Let $Z = \{x : x^T Q x = 0 : x^T x \leq 1\}$.

Let $K = \text{conv}(Z)$.

Let $K' = S = \{x : e^T x = 1, x \geq 0\}$.

Using the algorithmic separating hyperplane theorem in the corresponding Triangle Algorithm, we can give a fully polynomial-time approximation scheme to either separate $S$ from $\text{conv}(Z)$, hence proving that either $Z \cap S$ is empty,

or to give an approximate point in $\text{conv}(Z) \cap S$. 

(Rutgers)
Triangle Algorithm
July 26, 2016 56 / 57
**Decision Problem:** Given a symmetric $n \times n$ matrix $Q$, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?

This problem is NP-complete.

Let $Z = \{x : x^T Q x = 0 : x^T x \leq 1\}$.

Let $K = \text{conv}(Z)$.

Let $K' = S = \{x : e^T x = 1, x \geq 0\}$.

Using the algorithmic separating hyperplane theorem in the corresponding Triangle Algorithm, we can give a fully polynomial-time approximation scheme to either separate $S$ from $\text{conv}(Z)$, hence proving that either $Z \cap S$ is empty, or to give an approximate point in $\text{conv}(Z) \cap S$.

In particular, in the later case when $Z$ is convex, the algorithm gives an approximate zero of $Q$ in $S$. 
Related Articles and Forthcoming Work

- Experiments with the Triangle Algorithm for Linear Systems, 23rd Fall Workshop on Computational Geometry, City College of New York, Oct 25, 2013, with Thomas Gibson (2-page abstract).
- Experimental Study of the Convex Hull Decision Problem via a New Geometric Algorithm, 23rd Fall Workshop on Computational Geometry, City College of New York, Oct 25, 2013, with Meng Li. (2-page abstract).
- An Approximation to an NP-Complete Problem via The Triangle Algorithm, forthcoming.

Finally, there remain many open problems.