Algorithmic Aspects of Communication Networks

Chapter 3
General Optimization Methods for Network Design

Basic Facts About LP

- LP (Linear Program) in general form

  **indices**
  
  \[ j = 1, 2, \ldots, n \quad \text{variables} \]
  \[ i = 1, 2, \ldots, m \quad \text{constraints} \]

  **constants**
  
  \[ a_{ij} \quad \text{coefficient for variable } j \text{ in constraint } i \]
  \[ b_i \quad \text{right-hand side of constraint } i \]
  \[ c_j \quad \text{cost coefficient of variable } j \]

  **objective**
  
  minimize \( z = \sum_j c_j x_j \)

  **constraints**
  
  \[ \sum_j a_{ij} x_j \leq b_i, \quad i = 1, 2, \ldots, m \]
Basic Facts About LP

- LP (Linear Program) in general form

Let
\[ x = (x_1, x_2, \ldots, x_n); \]
\[ c = (c_1, c_2, \ldots, c_n); \]
\[ A = [a_{ij}], i = 1, 2, \ldots, m, j = 1, 2, \ldots, n; \]
and
\[ b = (b_1, b_2, \ldots, b_m). \]

Then the LP in general form can be written as

minimize \[ z = c x \]
subject to \[ Ax \leq b \]

- LP (Linear Program) in standard form

minimize \[ z = c x \]
subject to \[ Ax = b \]
\[ x \geq 0 \]

where \[ b \geq 0 \] and \( A \) has rank \( m \leq n \)

- Remark

Any LP in general form can be transformed in an LP in standard form and vice versa.
Basic Facts About LP

- **Example**

  \[
  \begin{align*}
  \text{minimize} & \quad z = -x - 3y \\
  \text{subject to} & \quad -x + y \leq 1 \\
  & \quad x + y \leq 2 \\
  & \quad x,y \geq 0
  \end{align*}
  \]

- **In general form**

  \[
  \begin{align*}
  \text{minimize} & \quad z = (-1, -3) (x,y)^T \\
  \text{subject to} & \quad -x + y \leq 1 \\
  & \quad x + y \leq 2 \\
  & \quad -x \leq 0 \\
  & \quad -y \leq 0
  \end{align*}
  \]

- **Example, cont'd**

  \[
  \begin{align*}
  \text{minimize} & \quad z = -x - 3y \\
  \text{subject to} & \quad -x + y \leq 1 \\
  & \quad x + y \leq 2 \\
  & \quad x,y \geq 0
  \end{align*}
  \]

- **In standard form**

  \[
  \begin{align*}
  \text{minimize} & \quad z = (-1, -3) (x,y)^T \\
  \text{subject to} & \quad -x + y + v = 1, \\
  & \quad x + y + w = 2, \\
  & \quad x \geq 0, y \geq 0, v \geq 0, w \geq 0
  \end{align*}
  \]

  The ‘artificial’ variables \(v, w\) are called **slack variables**.
Basic Facts About LP

- Structure of feasible set
  - We consider an LP in standard form

  minimize \( z = c^T x \)
  subject to \( Ax = b \), \( x \geq 0 \)

  where \( b \geq 0 \) and \( A \) has rank \( m \leq n \)

  - Its feasible set \( M \) is

    \[ M = \{ x : x \geq 0 \} \cap \{ x : Ax = b \} \]

    We put \( C = \{ x : x \geq 0 \} \), and so \( M = C \cap \{ x : Ax = b \} \).
Basic Facts About LP

- **Structure of feasible set, cont’d**

  - The set $\{ \mathbf{x} : A\mathbf{x} = \mathbf{b} \}$ is an affine subspace, i.e. the set of all solutions of an inhomogeneous system of linear equations. The set $\{ \mathbf{x} : \mathbf{x} \geq \mathbf{0} \}$ is a polyhedral set, i.e. the intersection of finitely many closed half-spaces. Consequently, $M$ is a polyhedral set. Both sets are convex and closed and consequently, $M$ is convex and closed.

  - A point $\mathbf{x}$ in $M$ is an **extreme point** of $M$ if it cannot be expressed as a convex linear combination of other points in $M$. The extreme points of a polyhedral set are called its **vertices**.
Basic Facts About LP

- **Structure of feasible set, cont’d**

  - Let \( a_1, a_2, \ldots, a_n \) be the column vectors of the matrix \( A \). A solution \( x = (x_1, x_2, \ldots, x_n) \) of \( Ax = b \) is a **basic solution** if the set \( \{a_k : x_k \neq 0\} \) is linearly independent.

- **Theorem 1**: Let \( A \) is a matrix with \( m \) rows and \( n \) columns such that \( A \) has rank \( m \). Then, a point \( x \) in \( M \) is a vertex of \( M \) if and only if it is a basic solution of \( Ax = b \).

**Proof of Thm. 1**

- \( x \) is not a vertex of \( M \)

  \[ \Rightarrow \exists \ u,v \in M : \exists c \in (0,1): x = cu + (1-c)v \text{ and } u - v \neq 0 \]
  \[ \Rightarrow \quad Au = Av = b \]
  \[ \Rightarrow \quad A(u - v) = 0 \]
  \[ \Rightarrow \quad \{a_k : u_k - v_k \neq 0\} \text{ is not linearly independent} \]

  Since \( u,v \geq 0 \), \( u_k - v_k \neq 0 \) and \( c \in (0,1) \) implies \( c u_k + (1 - c) v_k = x_k \neq 0 \).

  Hence, \( \{a_k : u_k - v_k \neq 0\} \subseteq \{a_k : x_k \neq 0\} \).

  Consequently, if \( x \) is not a vertex of \( M \), then \( \{a_k : x_k \neq 0\} \) is not linearly independent, i.e. \( x \) is not a basic solution of \( Ax = b \).
Basic Facts About LP

- \( \mathbf{x} \) is not a basic solution of \( \mathbf{A} \mathbf{x} = \mathbf{b} \)

  \[ \Rightarrow \{ \mathbf{a}_k : x_k \neq 0 \} \text{ is not linearly independent} \]
  \[ \Rightarrow \exists \ \mathbf{v} \in \mathbb{R}^n : \mathbf{A} \mathbf{v} = \mathbf{0} \text{ and } v_k \neq 0 \implies x_k \neq 0 \]

  If \( \varepsilon \) is sufficiently small, then
  \[ \mathbf{x}^+ \varepsilon \mathbf{v} \geq \mathbf{0} \text{ and } \mathbf{x}^- \varepsilon \mathbf{v} \geq \mathbf{0}. \]
  Clearly, \( \mathbf{A} (\mathbf{x}^+ \varepsilon \mathbf{v}) = \mathbf{A} (\mathbf{x}^- \varepsilon \mathbf{v}) = \mathbf{b} \) and
  \[ \mathbf{x} = \frac{1}{2} (\mathbf{x}^+ \varepsilon \mathbf{v}) + \frac{1}{2} (\mathbf{x}^- \varepsilon \mathbf{v}). \]

  Consequently, \( \mathbf{x} \) is not a vertex of \( M \).

Q.E.D.

Structure of feasible set, cont’d

An immediate consequence of Thm. 1 is the following Corollary 2.

- **Corollary 2:** \( M \) has at most \( n! / ((n-m)! \cdot m!) \) vertices.

Furthermore, one can prove that \( M \) has at least one vertex.

- **Theorem 3:** \( M \) has at least one vertex.
Basic Facts About LP

- **Example:**
  
  \[
  \begin{align*}
  x + y + z & \leq 4 \\
  x & \leq 2 \\
  z & \leq 3 \\
  3y + z & \leq 6 \\
  x, y, z & \geq 0 \\
  
  x &= 2 \\
  x + y + z &= 4 \\
  3y + z &= 6 \\
  z &= 3
  \end{align*}
  \]

- **Theorem 4:** A bounded polyhedral set is the convex hull of its vertices.

- **Theorem 5:** Let \( M \) be a nonempty closed bounded polyhedral set (= a polyhedron) and \( c \) an \( n \)-dimensional vector. Then the LP (*)

  \[
  \begin{align*}
  \text{minimize} \quad z &= c \cdot x \\
  \text{subject to} \quad x &\in M
  \end{align*}
  \]

  has an optimal solution which is a vertex of \( M \).
Basic Facts About LP

- **Proof of Thm. 5:**
  - Since $M$ is a bounded closed set and the objective function $z = c \cdot x$ is continuous, there is an optimal solution $x \in M$.
  - Suppose that $x$ is not a vertex of $M$. Then $x$ is a convex linear combination of vertices $x_1, \ldots, x_r$
    
    $x = \sum_j y_j x_j$
    
    with $y_1 > 0, \ldots, y_r > 0$ and $\sum_j y_j = 1$.

- **Proof of Thm. 5, cont’d**
  - Consequently, $cx = \sum_j y_j c x_j$.
  - Since $x$ is an optimal solution, $c x_j \geq c \cdot x$ for all $j = 1, \ldots, r$.
  - It follows $cx = c x_j$ for all $j = 1, \ldots, r$, i.e. all vertices $x_1, \ldots, x_r$ are also optimal solutions.

Q.E.D.
Basic Facts About LP

A similar but more elaborate argument can be used to prove the following Theorem 6.

- **Theorem 6:** If $M$ is unbounded, then (*) has either no optimal solution at all, or it has an optimal solution which is a vertex of $M$.

The Simplex Algorithm

**The Simplex Algorithm**

We consider the following

- **LP (Linear Program) in standard form**

  minimize $z = c \ x$

  subject to $Ax = b$

  $x \geq 0$

  where $b \geq 0$ and $A$ has rank $m \leq n$

- Let $M$ be the **feasible set** i.e.

  $M = \{x : x \geq 0\} \cap \{x : Ax = b\}$
The Simplex Algorithm

The Simplex Algorithm consists of two parts, called Phase 1 and Phase 2.

- The input of Phase 2 is a feasible basic solution.
- Phase 2 stops when an optimal feasible basic solution has been found, or it has been detected that the objective function is unbounded from below on $M$.
- Phase 1 is needed only if no feasible basic solution is known.
- Phase 1 stops when a feasible basic solution has been found, or it has been detected that $M$ is empty.
- Phase 1 consists in applying Phase 2 to a modified LOP.

The Simplex Algorithm: Phase 2

- Let $x$ be a feasible basic solution with $P = \{ k : x_k \neq 0 \}$ and $S(x) = \{ a_k : x_k \neq 0 \}$.

- Let $B(x)$ be a basis of columns of $A$ containing $S(x)$ and $Z = \{ k : a_k \in B(x) \}$ and $L = \{ 1, \ldots, n \} - Z$.

- If $P \neq Z$, $x$ is called degenerate. Otherwise, $x$ is non-degenerate.
The Simplex Algorithm

- For each \( j \) let
  \[
  a_j = \sum_{k \in \mathbb{Z}} t_{kj} a_k
  \]

- Because \( B(\mathbf{x}) \) is a basis all \( t_{kj} \) are uniquely determined by \( j \).

- If \( k \in \mathbb{Z} \), then \( t_{k,k} = 1 \) and \( t_{j,k} = 0 \) for \( j \neq k \).

- For \( j \in \mathbb{Z} \) let:
  \[
  u_j = \sum_{k \in \mathbb{Z}} t_{kj} c_k \quad \text{and} \quad d_j = u_j - c_j.
  \]

- Let \( \mathbf{y} \) an arbitrary feasible solution. Then
  \[
  \sum_{k \in \mathbb{Z}} x_k a_k = b = \sum_{j=1 \ldots n} y_j a_j = \sum_{j=1 \ldots n} y_j \left( \sum_{k \in \mathbb{Z}} t_{kj} a_k \right) = \sum_{k \in \mathbb{Z}} \left( \sum_{j=1 \ldots n} y_j t_{kj} \right) a_k.
  \]

- Since \( B(\mathbf{x}) \) is a basis it follows that
  \[
  x_k = \sum_{j=1 \ldots n} y_j t_{kj} = y_k + \sum_{j \in \mathbb{Z}} y_j t_{kj}, \quad k \in \mathbb{Z}.
  \]

- Consequently,
  \[
  y_k = x_k - \sum_{j \in \mathbb{Z}} y_j t_{kj}, \quad k \in \mathbb{Z}.
  \]
The Simplex Algorithm

- This implies

\[
\mathbf{c}y = \sum_{k \in Z} c_k y_k + \sum_{j \in L} c_j y_j = \sum_{k \in Z} c_k (x_k - \sum_{j \in L} y_j t_{k,j}) + \sum_{j \in L} c_j y_j = \sum_{k \in Z} c_k x_k - \sum_{j \in L} (\sum_{k \in Z} t_{k,j} c_k - c_j) y_j = \sum_{k \in Z} c_k x_k - \sum_{j \in L} d_j y_j = c\mathbf{x} - \sum_{j \in L} d_j y_j .
\]

- Case 1: For all \( j \in L : 0 \geq d_j \). Then \( c\mathbf{y} \geq c\mathbf{x} \) for all feasible solutions \( \mathbf{y} \), i.e. \( \mathbf{x} \) is optimal, and the algorithm stops.

- Case 2: There is an \( j \in L : 0 < d_j \), and for all \( k \in Z : 0 \geq t_{k,j} \).

Then \( \mathbf{y} \) with

- \( y_k = x_k - \varepsilon t_{k,j} \) for \( k \in Z \)
- \( y_j = \varepsilon \)
- \( y_i = 0 \) for \( i \in L \) with \( i \neq j \)

is a feasible solution and \( c\mathbf{y} = c\mathbf{x} - d_j \varepsilon \). Since \( 0 < d_j \), \( c\mathbf{y} \to -\infty \) for \( \varepsilon \to \infty \), i.e. the objective function is unbounded from below on \( M \). The algorithm stops.
Case 3: There is an \( s \in L : 0 < d_s \), and a \( k \in \mathbb{Z} : t_{k,s} > 0 \).
- Let \( \varepsilon = \min\{ x_k / t_{k,s} : k \in \mathbb{Z} \text{ and } t_{k,s} > 0 \} \). Note that \( \varepsilon \geq 0 \).

Suppose that \( \varepsilon = x_r / t_{r,s} \).

Let \( y \) with:
\[
  y_k = x_k - \varepsilon t_{k,j} \quad \text{for } k \in \mathbb{Z} \\
  y_s = \varepsilon \\
  y_j = 0 \quad \text{for } j \in L \text{ with } j \neq s
\]

\( y \) is a feasible basic solution and \( cy = cx - d_s \varepsilon \).

If \( \varepsilon > 0 \), then \( cy > cx \), i.e. the value of the objective function has decreased. The algorithm starts over with \( x = y \).

If \( \varepsilon = 0 \), then \( cy = cx \), i.e. the value of the objective function has not changed. The algorithm starts over with \( x = y \). (Special care is needed to avoid "cycling").
The Simplex Algorithm: Phase 1

- **Case A:** If the original LOP is in standard form, and there is no feasible basic solution known, an auxiliary LOP can be used to find one.

- The auxiliary LOP

  minimize \( z = y \)

  subject to \( y + Ax = b, \]
  \( x \geq 0, y \geq 0 \)

  where (as for the original LOP) \( b \geq 0 \) and \( A \) has rank \( m \leq n \)

The Simplex Algorithm

- Clearly, \( x=0 \) and \( y=b \) is a feasible basic solution of the auxiliary LOP.
- The Simplex Algorithm, Phase 2 is applied to obtain an optimal basic solution \( (x',y') \) of the auxiliary LOP.
- If \( y'=0 \), then \( x' \) is a feasible basic solution of the original LOP.
- If \( y' \neq 0 \), then the original LOP has no feasible basic solution.
Case B: The original LOP is given the form

\[
\begin{align*}
\text{minimize} & \quad z = c x \\
\text{subject to} & \quad Ax \leq b \\
& \quad x \geq 0 \\
\text{where} & \quad b \geq 0 \quad \text{and} \quad A \text{ has rank } m \leq n
\end{align*}
\]

We introduce slack variables \( y = (y_1, \ldots, y_m) \) and obtain

\[
\begin{align*}
\text{minimize} & \quad z = c x \\
\text{subject to} & \quad y + Ax = b \\
& \quad x \geq 0, \ y \geq 0
\end{align*}
\]

The new LOP is equivalent to the original one, and \( x=0, y=b \) is a feasible basic solution of the new LOP.

The Complexity of the Simplex Algorithm

- There are examples where the Simplex Algorithm starting with a specified feasible basic solution will pass through all vertices, and the number of vertices is exponential in the number \( n \) of variables.
- Yet, in practice the Simplex Algorithm is a very efficient approach. In many relevant cases, its running time is proportional to \( m+n \).
- There are polynomial time algorithms for LOPs. (Khachian, Karmarkar)
- There are polynomial time variations of the Simplex Algorithm for special cases, e.g. single-commodity network flow problems.
Branch and Bound – the basic algorithm

- The idea of branch and bound methods
  - Consider a (hard to solve) optimization problem
    \[
    \text{minimize } f(x) \quad \text{subject to } x \in M
    \]  
    (1)
  - Associate a relaxed optimization problem
    \[
    \text{minimize } g(x) \quad \text{subject to } x \in R
    \]  
    (2)
    such that
    - \( R \supseteq M \),
    - if \( x \in R \), then \( g(x) = f(x) \), and
    - (2) can be solved efficiently.

- Recall:
  An optimal solution for (1) is an element \( x' \) of \( M \) such that \( f(x) \geq f(x') \) for all \( x \) in \( M \) (likewise for (2)).

- Claim 1:
  (a) If \( y' \) is an optimal solution for (2), then \( f(x) \geq f(y') \) for all \( x \) in \( M \).
  (b) If \( y' \) is an optimal solution for (2) and \( y' \in M \), then \( y' \) is an optimal solution for (1) too.

- Claim 2:
  Let \( M = M_1 \cup M_2 \) be a partition of \( M \), and let \( x'_k \) be an optimal solution for the OP
  \[
  \text{minimize } f(x) \quad \text{subject to } x \in M_k
  \]  
  (1\(_k\)) \((k=1,2)\). Then \( \text{argmin}_x \{ f(x'_1), f(x'_2) \} \) is an optimal solution for (1).
Branch and Bound – the basic algorithm

- Let $R'$ be a nonempty subset of $R$, and consider the optimization problem

\[ \text{minimize } g(y) \quad \text{subject to } y \in R' \quad (R'). \]

- If $(R')$ has a solution, then $\text{SOLVE}(R')$ returns a pair $(y', g(y'))$ consisting of an optimal solution $y'$ of $(R')$ and the corresponding value $g(y')$, otherwise $\text{SOLVE}(R')$ returns $(n, n)$.

- If $\text{SOLVE}(R')$ yields an optimal solution $y' \in M$, then $\text{BRANCH}(R', y')$ returns two disjoint subsets $R'_1, R'_2$ of $R'$ such that

\[ M \cap R' = (M \cap R'_1) \cup (M \cap R'_2). \]

The idea of the algorithm

- **Initialization**: $L \leftarrow \{R\}; \quad \text{best} \leftarrow \infty$;
- **While** $L \neq \emptyset$ **do**

  **begin**

  - choose $B \in L$;
  - $(y', g(y')) \leftarrow \text{SOLVE}(B)$;
  - **if** $y' \in M$ and $g(y') < \text{best}$ **then**
    **begin**
    - $\text{best} \leftarrow g(y')$;
    - $y_{\text{best}} \leftarrow y'$;
    - remove $B$ from $L$;
    **end**;
  - **if** $g(y') \geq \text{best}$ **then** remove $B$ from $L$; % bounding
  **else**
    **begin**
    - $(B_1, B_2) \leftarrow \text{BRANCH}(B, y')$;
    - $L \leftarrow L \cup \{B_1, B_2\}$;
    **end**;
  **end**
Branch and Bound – the basic algorithm

A more concrete recursive realization of Branch and Bound

- Initialization $A \leftarrow M$, $B \leftarrow R$, $best \leftarrow \infty$

- procedure $BB(A, B, f, g)$
  
  begin
  
  $(y', g(y')) \leftarrow SOLVE(B)$
  
  if $y' \in A$ then
  
  if $g(y') < best$ then
  
  begin
  
  best $\leftarrow g(y')$; return $(y', g(y'))$
  
  else
  
  if $g(y') \geq best$ then return % bounding %
  
  else
  
  begin
  
  $(B_1, B_2) \leftarrow BRANCH(B, y')$
  
  $BB(A, B_1, f, g)$;
  
  $BB(A, B_2, f, g)$;
  
  end;
  
  end

Remarks:

- Branch and Bound yields an (exact) optimal solution provided there is a constant $K$ (depending on the input) such that all subproblems obtained after at most $K$ repetitions of BRANCH either have no optimal solution or their optimal solutions are in $M$.

- Branch and Bound is not (necessarily) efficient.
Branch and Bound for MIP

- Consider the MIP (1)

\[
\text{minimize} \quad f(x) = c \cdot x \\
\text{subject to} \quad A x = b, \quad x \geq 0, \text{ and } x_j \text{ is integer for all } k \in I \\
\text{where} \quad x = (x_1, \ldots, x_n), \quad b \geq 0, \quad A \text{ has rank } m \leq n, \text{ and } I \subseteq \{1, \ldots, n\}
\]

- Choose as relaxed problem the LOP (2) (called LP-relaxation)

\[
\text{minimize} \quad f(x) = c \cdot x \\
\text{subject to} \quad A x = b, \quad x \geq 0 \\
\text{where} \quad x = (x_1, \ldots, x_n), \quad b \geq 0, \text{ and } A \text{ has rank } m \leq n
\]

- SOLVE can be any method to solve LOPs (e.g. the simplex algorithm).

- If \((y^*, g(y^*))\) is an optimal solution where \(y^*_k\) is not an integer, then 
  BRANCH\((B, y^*)\) ‘adds’ new inequalities to \(B\), i.e.

\[
\text{BRANCH}(B, y^*) = (B_1, B_2) \text{ where} \\
B_1 = \{ y \in B \mid y_k \leq [y^*_k] \}, \\
B_2 = \{ y \in B \mid y_k \geq \{y^*_k\} \}.
\]

(Notation: \([y^*_k]\) is the smallest integer not smaller than \(y^*_k\) and \(\{y^*_k\}\) is the greatest integer not greater than \(y^*_k\).)
Cutting planes for MIP

☐ Consider the MIP (1)

\[
\begin{align*}
\text{minimize} & \quad f(x) = c \cdot x \\
\text{subject to} & \quad x \in M = \{x \mid Ax = b, x \geq 0, \text{ and } x_j \text{ is integer for all } k \in I\} \\
\text{where} & \quad x = (x_1, \ldots, x_n) \text{ and } I \subseteq \{1, \ldots, n\},
\end{align*}
\]

and its LP-relaxation (2)

\[
\begin{align*}
\text{minimize} & \quad f(x) = c \cdot x \\
\text{subject to} & \quad x \in R = \{x \mid Ax = b, x \geq 0\}.
\end{align*}
\]

☐ A valid cut for (1) is an equality \( d \cdot x \geq q \) such that

\[
\begin{align*}
\{x \mid Ax = b, x \geq 0, d \cdot x \geq q\} & \neq R \\
\{x \mid Ax = b, x \geq 0, \text{ and } x_j \text{ is integer for all } k \in I, d \cdot x \geq q\} & = M.
\end{align*}
\]

☐ Outline of a cutting plane algorithm

\[
\begin{align*}
\text{begin} \\
B & \leftarrow R; \\
(y', g(y')) & \leftarrow \text{SOLVE}(B); \\
\text{while } y' \notin M & \text{ do} \\
\text{begin} \\
\text{compute a valid cut } d \cdot x \geq q \text{ for } (B); \\
B & \leftarrow \{x \in B \mid d \cdot x \geq q\}; \\
(y', g(y')) & \leftarrow \text{SOLVE}(B); \\
\text{end}; \\
\text{end}
\end{align*}
\]
Cutting planes for MIP

- There are two kinds of cuts:
  - problem specific ones (e.g. for the knapsack problem), and
  - general purpose ones (Gomory cuts).

- Gomory cuts
  - Let \( x = (x_1, \ldots, x_n) \) be a basic solution of the LP-relaxation such that \( x_i \) is not an integer.
  - Then \( x_i \) is a basic variable and from the simplex tableau we know that
    \[
    x_i + \sum a_{ij} x_j = b_i \tag{a}
    \]
  - Equation (a) implies
    \[
    x_i + \sum [a_{ij}] x_j - [b_i] = b_i - [b_i] - \sum (a_{ij} - [a_{ij}]) x_j \tag{b}
    \]

- For any point \( x \in M \) the right hand side of (b) is less than 1, and the left hand side is an integer.
- Consequently,
  \[
  b_i - [b_i] - \sum (a_{ij} - [a_{ij}]) x_j \leq 0 \tag{c}
  \]
  for any point \( x \in M \).

- Furthermore, for the original basic solution \( x = (x_1, \ldots, x_n) \) is
  \[
  b_i - [b_i] - \sum (a_{ij} - [a_{ij}]) x_j = b_i - [b_i]
  \]
  (because all non basic variables =0) not an integer, and therefore
  \[
  b_i - [b_i] - \sum (a_{ij} - [a_{ij}]) x_j = b_i - [b_i] > 0,
  \]
  i.e. (c) excludes \( x \).

- Hence (c) is a valid cut. The inequality (c) is called a Gomory cut.