Combinatorial Optimization in Computer Vision (IN2245)

Frank R. Schmidt
Csaba Domokos

Winter Semester 2015/2016

7. Linear Programming

Linear Programming

Maximal Flow Revisited

Given a network $G = (V, E, c, s, t)$, both the capacity function c and a flow f can be interpreted as vectors $c, f \in \mathbb{R}^{|E|}$. The MaxFlow can be rewritten as

$$
\begin{array}{rll}
\max & z & \\
\text{subject to} & 0 \leqslant f_e \leqslant c_e & \text{for all } e \in E \\
& [\operatorname{div} f]_i = 0 & \text{for all } i \in V - \{s, t\} \\
& [\operatorname{div} f]_s = +z & \\
& [\operatorname{div} f]_t = -z & \\
& z \geqslant 0 &
\end{array}
$$

Maximal Flow Revisited

Given a network $G = (V, E, c, s, t)$, both the capacity function c and a flow f can be interpreted as vectors $c, f \in \mathbb{R}^{|E|}$. The MaxFlow can be rewritten as

$$\begin{aligned}
\max \quad & z & \\
\text{subject to} \quad & 0 \leqslant f_e \leqslant c_e & \text{for all } e \in E \\
& [\operatorname{div} f]_i = 0 & \text{for all } i \in V - \{s, t\} \\
& [\operatorname{div} f]_s = +z & \\
& [\operatorname{div} f]_t = -z & \\
& z \geqslant 0 &
\end{aligned}$$

Here, $\operatorname{div}\colon \mathbb{R}^{|E|} \to \mathbb{R}^{|V|}$ is a linear mapping that maps information from the edges to information on the vertices.

Maximal Flow Revisited

Given a network $G = (V, E, c, s, t)$, both the capacity function c and a flow f can be interpreted as vectors $c, f \in \mathbb{R}^{|E|}$. The MaxFlow can be rewritten as

$$
\begin{array}{rcl}
\max & z & \\
\text{subject to} & 0 \leqslant f_e \leqslant c_e & \text{for all } e \in E \\
& [\operatorname{div} f]_i = 0 & \text{for all } i \in V - \{s,t\} \\
& [\operatorname{div} f]_s = +z & \\
& [\operatorname{div} f]_t = -z & \\
& z \geqslant 0 &
\end{array}
$$

Here, $\operatorname{div}\colon \mathbb{R}^{|E|} \to \mathbb{R}^{|V|}$ is a linear mapping that maps information from the edges to information on the vertices.

Note that the **objective function** and the **constraints** are all linear.

A **Linear Program (LP)** is an optimization problem of a linear function with respect to linear constraints, i.e.,

$$\begin{Bmatrix} \min \\ \max \end{Bmatrix}_{x \in \mathbb{R}^n} \langle c, x \rangle$$

$$\text{subject to} \langle a_i, x \rangle \begin{Bmatrix} \leqslant \\ = \\ \geqslant \end{Bmatrix} b_i \qquad \text{for all } i = 1, \dots, m$$

A **Linear Program (LP)** is an optimization problem of a linear function with respect to linear constraints, i.e.,

$$\begin{Bmatrix} \min \\ \max \end{Bmatrix}_{x \in \mathbb{R}^n} \langle c, x \rangle$$

$$\text{subject to } \langle a_i, x \rangle \begin{Bmatrix} \leqslant \\ = \\ \geqslant \end{Bmatrix} b_i \qquad \text{for all } i = 1, \ldots, m$$

If we use the following partial ordering on $\mathbb{R}^n$:

$$x \leqslant y \quad \Leftrightarrow \quad x_k \leqslant y_k \qquad \text{for all } k = 1, \ldots, n$$

we can simplify the notation of LPs.

Canonical and Standard Form

An LP is in **canonical form** if it is of the form

$$\max_{x \in \mathbb{R}^n} \langle c, x \rangle$$
$$\text{subject to } Ax \leqslant b$$
$$x \geqslant 0$$

for a **constraint matrix** $A \in \mathbb{R}^{m \times n}$, a **constraint vector** $b \in \mathbb{R}^m$ and a **cost vector** $c \in \mathbb{R}^n$. We have n variables and m constraints.

Canonical and Standard Form

An LP is in **canonical form** if it is of the form

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \langle c, x \rangle \\ \text{subject to } Ax \leqslant b \\ x \geqslant 0 \end{aligned}$$

for a **constraint matrix** $A \in \mathbb{R}^{m \times n}$, a **constraint vector** $b \in \mathbb{R}^m$ and a **cost vector** $c \in \mathbb{R}^n$. We have n variables and m constraints.

An LP is in **standard form** if it is of the form

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \langle c, x \rangle \\ \text{subject to } Ax = b \\ x \geqslant 0 \end{aligned}$$

LP Transformation

Every minimization problem becomes an equivalent maximization problem by replacing c with $-c$.

LP Transformation

Every minimization problem becomes an equivalent maximization problem by replacing c with $-c$.

Note that the following equivalences can transform the constraints of an LP in purely "$\leqslant$" or "$=$" constraints:

$$\langle a_i, x\rangle \geqslant b_i \quad \Leftrightarrow \quad \langle -a_i, x\rangle \leqslant -b_i$$

$$\langle a_i, x\rangle = b_i \quad \Leftrightarrow \quad \begin{aligned}\langle +a_i, x\rangle &\leqslant +b_i,\\ \langle -a_i, x\rangle &\leqslant -b_i\end{aligned}$$

$$\langle a_i, x\rangle \leqslant b_i \quad \Leftrightarrow \quad \langle a_i, x\rangle + s_i = b_i$$

The extra variable in the last equivalence is called **slack variable** $s_i \geqslant 0$.

LP Transformation

Every minimization problem becomes an equivalent maximization problem by replacing c with $-c$.

Note that the following equivalences can transform the constraints of an LP in purely "$\leqslant$" or "$=$" constraints:

$$\langle a_i, x\rangle \geqslant b_i \quad \Leftrightarrow \quad \langle -a_i, x\rangle \leqslant -b_i$$

$$\langle a_i, x\rangle = b_i \quad \Leftrightarrow \quad \begin{aligned}\langle +a_i, x\rangle &\leqslant +b_i,\\ \langle -a_i, x\rangle &\leqslant -b_i\end{aligned}$$

$$\langle a_i, x\rangle \leqslant b_i \quad \Leftrightarrow \quad \langle a_i, x\rangle + s_i = b_i$$

The extra variable in the last equivalence is called **slack variable** $s_i \geqslant 0$.

If a variable x_i is not constrained $(x_i \geqslant 0)$, one can use two constrained variables $x_i^+, x_i^- \geqslant 0$ and replace each occurence of x_i with $x_i^+ - x_i^-$.

The canonical form of the MaxFlow problem is

$$\max_{f \in \mathbb{R}^{|E|}, z \in \mathbb{R}} \quad z$$

subject to

$$\begin{pmatrix} \mathrm{Id} & 0 \\ \mathrm{div} & (\mathbf{1}_t - \mathbf{1}_s) \\ -\mathrm{div} & -(\mathbf{1}_t - \mathbf{1}_s) \end{pmatrix} \begin{pmatrix} f \\ z \end{pmatrix} \leqslant \begin{pmatrix} c \\ 0 \\ 0 \end{pmatrix}$$

$$f, z \geqslant 0$$

Maximal Flow as LP

The canonical form of the MaxFlow problem is

$$\max_{f \in \mathbb{R}^{|E|}, z \in \mathbb{R}} \quad z$$

subject to

$$\begin{pmatrix} \mathrm{Id} & 0 \\ \mathrm{div} & (\mathbf{1}_t - \mathbf{1}_s) \\ -\mathrm{div} & -(\mathbf{1}_t - \mathbf{1}_s) \end{pmatrix} \begin{pmatrix} f \\ z \end{pmatrix} \leqslant \begin{pmatrix} c \\ 0 \\ 0 \end{pmatrix}$$

$$f, z \geqslant 0$$

The standard form of the MaxFlow problem is

$$\max_{f, r \in \mathbb{R}^{|E|}, z \in \mathbb{R}} \quad z$$

subject to

$$\begin{pmatrix} \mathrm{Id} & \mathrm{Id} & 0 \\ \mathrm{div} & 0 & (\mathbf{1}_t - \mathbf{1}_s) \end{pmatrix} \begin{pmatrix} f \\ r \\ z \end{pmatrix} = \begin{pmatrix} c \\ 0 \end{pmatrix}$$

$$f, r, z \geqslant 0$$

Basic Feasible Solutions

In the following, we assume that an LP is given in its standard form and that A is of maximal rank, i.e. $\mathrm{rank}(A) = m \leqslant n$. If $x \geqslant 0$ satisfies $Ax = b$, x is called **feasible**.

Basic Feasible Solutions

In the following, we assume that an LP is given in its standard form and that A is of maximal rank, i.e. $\operatorname{rank}(A) = m \leqslant n$. If $x \geqslant 0$ satisfies $Ax = b$, x is called **feasible**.

Given the decomposition $\{1, \dots, n\} = B + N$ with $|B| = m$, we can define A_B as the submatrix of A that contains only those columns a^i with indices $i \in B$. Since A has maximal rank, we can select B such that $A_B \in \mathbb{R}^{m \times m}$ has maximal rank and we can compute

$$x_B = A_B^{-1} b$$

Basic Feasible Solutions

In the following, we assume that an LP is given in its standard form and that A is of maximal rank, i.e. $\operatorname{rank}(A) = m \leqslant n$. If $x \geqslant 0$ satisfies $Ax = b$, x is called **feasible**.

Given the decomposition $\{1, \ldots, n\} = B + N$ with $|B| = m$, we can define A_B as the submatrix of A that contains only those columns a^i with indices $i \in B$. Since A has maximal rank, we can select B such that $A_B \in \mathbb{R}^{m \times m}$ has maximal rank and we can compute

$$x_B = A_B^{-1} b$$

x_B only defines those entries of x whose indices are in B. Filling the rest of x with zeros ($x_N = 0$), we obtain a feasible x. Feasible x that are created in this way ($x = x_B + x_N$) are called **basic feasible solutions**.

Fundamental Theorem of LP (Part 1)

Theorem 1. *If there is a feasible x, there is a basic feasible solution x'.*

Proof.

Fundamental Theorem of LP (Part 1)

Theorem 1. *If there is a feasible x, there is a basic feasible solution x'.*

Proof. Without loss of generality, let us assume $b = \sum_{i=1}^{k} x_i a^i$ with $x_i > 0$.

Fundamental Theorem of LP (Part 1)

Theorem 1. *If there is a feasible x, there is a basic feasible solution x'.*

Proof. Without loss of generality, let us assume $b = \sum_{i=1}^{k} x_i a^i$ with $x_i > 0$.

Case 1: The k a_i are linearly independent.

$k = m$ proves the theorem. Otherwise, $m - k$ of the remaining vectors form a base and x is a basic feasible solution with respect to these indices.

Fundamental Theorem of LP (Part 1)

Theorem 1. *If there is a feasible x, there is a basic feasible solution x'.*

Proof. Without loss of generality, let us assume $b = \sum_{i=1}^{k} x_i a^i$ with $x_i > 0$.

Case 1: The k a_i are linearly independent.
$k = m$ proves the theorem. Otherwise, $m - k$ of the remaining vectors form a base and x is a basic feasible solution with respect to these indices.

Case 2: The k a_i are linearly dependent.
We have $0 = \sum_{i=1}^{k} \lambda_i a^i$ with at least one $\lambda_i > 0$ and thus for all $\epsilon > 0$

$$b = \sum_{i=1}^{k} (x_i - \epsilon\lambda_i) a_i$$

Fundamental Theorem of LP (Part 1)

Theorem 1. *If there is a feasible x, there is a basic feasible solution x'.*

Proof. Without loss of generality, let us assume $b = \sum_{i=1}^{k} x_i a^i$ with $x_i > 0$.

Case 1: The k a_i are linearly independent.
$k = m$ proves the theorem. Otherwise, $m - k$ of the remaining vectors form a base and x is a basic feasible solution with respect to these indices.

Case 2: The k a_i are linearly dependent.
We have $0 = \sum_{i=1}^{k} \lambda_i a^i$ with at least one $\lambda_i > 0$ and thus for all $\epsilon > 0$

$$b = \sum_{i=1}^{k} (x_i - \epsilon\lambda_i) a_i$$

Choosing $\epsilon = \min\left\{ \frac{x_i}{\lambda_i} \middle| \lambda_i > 0 \right\}$ creates a feasible solution $x' = x - \epsilon\lambda$ that uses at most $k - 1$ positive variables. Iterating this step leads eventually to the 1st case of linear independence. □

Fundamental Theorem of LP (Part 2)

Theorem 2. *If x^* is feasible, there is an optimal basic feasible solution x'.*

Proof.

Fundamental Theorem of LP (Part 2)

Theorem 2. *If x^* is feasible, there is an optimal basic feasible solution x'.*

Proof. Without loss of generality, let us assume $b = \sum_{i=1}^{k} x_i^* a^i$ with $x_i^* > 0$.

Fundamental Theorem of LP (Part 2)

Theorem 2. *If x^* is feasible, there is an optimal basic feasible solution x'.*

Proof. Without loss of generality, let us assume $b = \sum_{i=1}^{k} x_i^* a^i$ with $x_i^* > 0$.

Case 1: The k a_i are linearly independent. (Analogously as before)

Fundamental Theorem of LP (Part 2)

Theorem 2. *If x^* is feasible, there is an optimal basic feasible solution x'.*

Proof. Without loss of generality, let us assume $b = \sum_{i=1}^{k} x_i^* a^i$ with $x_i^* > 0$.

Case 1: The k a_i are linearly independent. (Analogously as before)

Case 2: The k a_i are linearly dependent.

Create analogously as before $x' = x^* - \epsilon\lambda$ and we have

$$\langle c, x' \rangle = \langle c, x^* \rangle - \epsilon \langle c, \lambda \rangle$$

Fundamental Theorem of LP (Part 2)

Theorem 2. *If x^* is feasible, there is an optimal basic feasible solution x'.*

Proof. Without loss of generality, let us assume $b = \sum_{i=1}^{k} x_i^* a^i$ with $x_i^* > 0$.

Case 1: The k a_i are linearly independent. (Analogously as before)

Case 2: The k a_i are linearly dependent.

Create analogously as before $x' = x^* - \epsilon\lambda$ and we have

$$\langle c, x' \rangle = \langle c, x^* \rangle - \epsilon \langle c, \lambda \rangle$$

If $\langle c, \lambda \rangle \neq 0$, we could improve x^* for small ϵ, which contradicts the optimality of x^*.

Fundamental Theorem of LP (Part 2)

Theorem 2. *If x^* is feasible, there is an optimal basic feasible solution x'.*

Proof. Without loss of generality, let us assume $b = \sum_{i=1}^{k} x_i^* a^i$ with $x_i^* > 0$.

Case 1: The k a_i are linearly independent. (Analogously as before)

Case 2: The k a_i are linearly dependent.

Create analogously as before $x' = x^* - \epsilon\lambda$ and we have

$$\langle c, x' \rangle = \langle c, x^* \rangle - \epsilon \langle c, \lambda \rangle$$

If $\langle c, \lambda \rangle \neq 0$, we could improve x^* for small ϵ, which contradicts the optimality of x^*.

Thus, $\langle c, \lambda \rangle = 0$, which proves the optimality of x'. x' is a feasible solution that uses at most $k-1$ positive variables. Iterating this step eventually leads to the case of linear independence and thus, proves the theorem. $\square$

Simplex Method

Idea of the Simplex Method

We saw that it is enough to restrict ourselves to basic feasible solutions.

Idea of the Simplex Method

We saw that it is enough to restrict ourselves to basic feasible solutions.

Since a basic feasible solution only depends on the choice $B \subset \{1, \ldots, n\}$, we have no more than $\binom{n}{m}$ basic feasible solutions.

Idea of the Simplex Method

We saw that it is enough to restrict ourselves to basic feasible solutions.

Since a basic feasible solution only depends on the choice $B \subset \{1, \ldots, n\}$, we have no more than $\binom{n}{m}$ basic feasible solutions.

The **Simplex method** works as following

1. Find a basic feasible solution x

Idea of the Simplex Method

We saw that it is enough to restrict ourselves to basic feasible solutions.

Since a basic feasible solution only depends on the choice $B \subset \{1, \ldots, n\}$, we have no more than $\binom{n}{m}$ basic feasible solutions.

The **Simplex method** works as following

1. Find a basic feasible solution x
2. If $\langle c, x \rangle$ is not optimal, find a better basic feasible solution

Idea of the Simplex Method

We saw that it is enough to restrict ourselves to basic feasible solutions.

Since a basic feasible solution only depends on the choice $B \subset \{1, \dots, n\}$, we have no more than $\binom{n}{m}$ basic feasible solutions.

The **Simplex method** works as following

1. Find a basic feasible solution x
2. If $\langle c, x \rangle$ is not optimal, find a better basic feasible solution
3. Iterate until convergence.

Idea of the Simplex Method

We saw that it is enough to restrict ourselves to basic feasible solutions.

Since a basic feasible solution only depends on the choice $B \subset \{1, \dots, n\}$, we have no more than $\binom{n}{m}$ basic feasible solutions.

The **Simplex method** works as following

1. Find a basic feasible solution x
2. If $\langle c, x \rangle$ is not optimal, find a better basic feasible solution
3. Iterate until convergence.

The main challenge is to perform Step 2 as efficiently as possible.

Idea of the Simplex Method

We saw that it is enough to restrict ourselves to basic feasible solutions.

Since a basic feasible solution only depends on the choice $B \subset \{1, \ldots, n\}$, we have no more than $\binom{n}{m}$ basic feasible solutions.

The **Simplex method** works as following

1. Find a basic feasible solution x
2. If $\langle c, x \rangle$ is not optimal, find a better basic feasible solution
3. Iterate until convergence.

The main challenge is to perform Step 2 as efficiently as possible.

There are certain LPs for which Step 1 is difficult. For the problems we will consider, this step will be very easy.

Pivot Operation

Let us assume that we have a basic feasible solution $x \in \mathbb{R}^n$ with its basic set B. This means $b = \sum_{i \in B} x_i a_i$.

Pivot Operation

Let us assume that we have a basic feasible solution $x \in \mathbb{R}^n$ with its basic set B. This means $b = \sum_{i \in B} x_i a_i$.

Since the $(a_i)_{i \in B}$ form a base of $\mathbb{R}^m$, we also have

$$a_j = \sum_{i \in B} y_{ij} a_i \qquad \text{for all } j \notin B$$

for certain $y_{ij} \in \mathbb{R}$.

Pivot Operation

Let us assume that we have a basic feasible solution $x \in \mathbb{R}^n$ with its basic set B. This means $b = \sum_{i \in B} x_i a_i$.

Since the $(a_i)_{i \in B}$ form a base of $\mathbb{R}^m$, we also have

$$a_j = \sum_{i \in B} y_{ij} a_i \qquad \text{for all } j \notin B$$

for certain $y_{ij} \in \mathbb{R}$.

For a small $\epsilon \geqslant 0$, the point $x_\epsilon = \epsilon a_j + \sum_{i \in B}(x_i - \epsilon y_{ij}) a_i$ is feasible, but not a basic feasible solution.

Pivot Operation

Let us assume that we have a basic feasible solution $x \in \mathbb{R}^n$ with its basic set B. This means $b = \sum_{i \in B} x_i a_i$.

Since the $(a_i)_{i \in B}$ form a base of $\mathbb{R}^m$, we also have

$$a_j = \sum_{i \in B} y_{ij} a_i \qquad \text{for all } j \notin B$$

for certain $y_{ij} \in \mathbb{R}$.

For a small $\epsilon \geqslant 0$, the point $x_\epsilon = \epsilon a_j + \sum_{i \in B}(x_i - \epsilon y_{ij}) a_i$ is feasible, but not a basic feasible solution.

For $\epsilon = \min_k \left\{ \frac{x_k}{y_{kj}} \middle| y_{kj} > 0 \right\}$, x_ϵ becomes a basic feasible solution w.r.t. the basic set $B - \{j\} + \{k\}$ where k is the minimizing index that defines ϵ.

Choosing a Good Pivot Operation

The pivot operation changes B by replacing one element with an element that is not in B. Now, we want to address, which element we should remove in order to improve the cost function with respect to the basic feasible solution that is associated with B.

Choosing a Good Pivot Operation

The pivot operation changes B by replacing one element with an element that is not in B. Now, we want to address, which element we should remove in order to improve the cost function with respect to the basic feasible solution that is associated with B.

Given a basic set B we can rewrite the constraint $Ay = b$ as

$$\begin{pmatrix}\mathrm{Id} & A_B^{-1}A_N\end{pmatrix}\begin{pmatrix}y_B \\ y_N\end{pmatrix} = A_B^{-1}b$$

and the cost function becomes for $y_N \neq 0$

$$\begin{aligned}\langle c, y\rangle = &\langle y_N, c_N\rangle + \left\langle A_B^{-1}\left(b - A_N y_N\right), c_B\right\rangle \\ &\langle x_B, c_B\rangle + \left\langle c_N - A_N^\top A_B^{-\top} c_B, y_N\right\rangle\end{aligned}$$

Choosing a Good Pivot Operation

The pivot operation changes B by replacing one element with an element that is not in B. Now, we want to address, which element we should remove in order to improve the cost function with respect to the basic feasible solution that is associated with B.

Given a basic set B we can rewrite the constraint $Ay = b$ as

$$\begin{pmatrix}\mathrm{Id} & A_B^{-1}A_N\end{pmatrix}\begin{pmatrix}y_B \\ y_N\end{pmatrix} = A_B^{-1}b$$

and the cost function becomes for $y_N \neq 0$

$$\begin{aligned}\langle c, y\rangle = &\langle y_N, c_N\rangle + \left\langle A_B^{-1}\left(b - A_N y_N\right), c_B\right\rangle \\ &\langle x_B, c_B\rangle + \left\langle c_N - A_N^\top A_B^{-\top} c_B, y_N\right\rangle\end{aligned}$$

We can improve the solution iff $c_N - A_N^\top A_B^{-\top} c_B$ has positive entries.

Simplex Algorithm

The theory that we studied so far explored everything we need to know in order to solve an LP.

Simplex Algorithm

The theory that we studied so far explored everything we need to know in order to solve an LP.

Given a basic feasible solution defined by B, we know whether it is optimal or not. If it is not optimal, we know how to change B in order to get an improved solution. In addition, we know how x_B will change if we change B.

Simplex Algorithm

The theory that we studied so far explored everything we need to know in order to solve an LP.

Given a basic feasible solution defined by B, we know whether it is optimal or not. If it is not optimal, we know how to change B in order to get an improved solution. In addition, we know how x_B will change if we change B.

The actual **Simplex Algorithm** that we discuss now combines this knowledge in order to reduce the computational complexity. After all, we do not want to recompute A_B^{-1} in every iteration.

Simplex Algorithm

The theory that we studied so far explored everything we need to know in order to solve an LP.

Given a basic feasible solution defined by B, we know whether it is optimal or not. If it is not optimal, we know how to change B in order to get an improved solution. In addition, we know how x_B will change if we change B.

The actual **Simplex Algorithm** that we discuss now combines this knowledge in order to reduce the computational complexity. After all, we do not want to recompute A_B^{-1} in every iteration.

To this end, we will store an LP that is equivalent to the original LP. This representation is called the **Simplex Tableau**.

Simplex Tableau

Given a basic feasible solution x_B and its basic set B, the simplex tableau is a $(m+1) \times (n+1)$ matrix of the following form

$$\left(\begin{array}{cc|c} 0 & c_N^\top - c_B^\top A_B^{-1} A_N & -\langle c_B, x_B \rangle \\ \hline \mathrm{Id} & A_B^{-1} A_N & A_B^{-1} b \end{array}\right)$$

Simplex Tableau

Given a basic feasible solution x_B and its basic set B, the simplex tableau is a $(m+1) \times (n+1)$ matrix of the following form

$$\left(\begin{array}{cc|c} 0 & c_N^\top - c_B^\top A_B^{-1} A_N & -\langle c_B, x_B \rangle \\ \hline \mathrm{Id} & A_B^{-1} A_N & A_B^{-1} b \end{array}\right)$$

If the first row has a positive entry (at position j), we can improve the solution by adding j to B. Select $i \in \operatorname{argmin} \left\{ \frac{(A_B^{-1} b)_i}{(A_B^{-1} a^j)_i} \middle| (A_B^{-1} a^j)_i > 0 \right\}$ and pivot the j^{th} column of the tableau, i.e, perform Gaussian elimination until the j^{th} column is the $(i+1)^{\text{th}}$ unit vector.

Simplex Tableau

Given a basic feasible solution x_B and its basic set B, the simplex tableau is a $(m+1) \times (n+1)$ matrix of the following form

$$\left(\begin{array}{cc|c} 0 & c_N^\top - c_B^\top A_B^{-1} A_N & -\langle c_B, x_B \rangle \\ \hline \mathrm{Id} & A_B^{-1} A_N & A_B^{-1} b \end{array}\right)$$

If the first row has a positive entry (at position j), we can improve the solution by adding j to B. Select $i \in \operatorname{argmin} \left\{ \frac{(A_B^{-1} b)_i}{(A_B^{-1} a^j)_i} \,\middle|\, (A_B^{-1} a^j)_i > 0 \right\}$ and pivot the j^{th} column of the tableau, i.e, perform Gaussian elimination until the j^{th} column is the $(i+1)^{\text{th}}$ unit vector.

This operation only changes the i^{th} column among the first m columns. In particular, one can show that we obtain a tableau with respect to $B - i + j$.

Simplex Tableau

The pivoting operation can be summarized in the following form:
At each step, there exists a vector $v \in \mathbb{R}^m$ such that the tableau is representable as the following product

$$\left(\begin{array}{cc} 1 & -v^\top \\ \hline 0 & A_B^{-1} \end{array}\right) \cdot \left(\begin{array}{cc} c^\top & 0 \\ \hline A & b \end{array}\right) = \left(\begin{array}{cc} 1 & -v^\top \\ \hline 0 & A_B^{-1} \end{array}\right) \cdot \left(\begin{array}{ccc} c_B^\top & c_N^\top & 0 \\ \hline A_B & A_N & b \end{array}\right)$$

$$= \left(\begin{array}{cc|c} 0 & c_N^\top - c_B^\top A_B^{-1} A_N & -\langle c_B, x_B \rangle \\ \hline \mathrm{Id} & A_B^{-1} A_N & A_B^{-1} b \end{array}\right)$$

Simplex Tableau

The pivoting operation can be summarized in the following form:
At each step, there exists a vector $v \in \mathbb{R}^m$ such that the tableau is representable as the following product

$$
\left(\begin{array}{cc} 1 & -v^\top \\ \hline 0 & A_B^{-1} \end{array}\right) \cdot \left(\begin{array}{cc} c^\top & 0 \\ \hline A & b \end{array}\right) = \left(\begin{array}{cc} 1 & -v^\top \\ \hline 0 & A_B^{-1} \end{array}\right) \cdot \left(\begin{array}{ccc} c_B^\top & c_N^\top & 0 \\ \hline A_B & A_N & b \end{array}\right)
$$

$$
= \left(\begin{array}{cc|c} 0 & c_N^\top - c_B^\top A_B^{-1} A_N & -\langle c_B, x_B \rangle \\ \hline \mathrm{Id} & A_B^{-1} A_N & A_B^{-1} b \end{array}\right)
$$

Note that the first equality is only true after reordering the columns.

Simplex Tableau

The pivoting operation can be summarized in the following form:
At each step, there exists a vector $v \in \mathbb{R}^m$ such that the tableau is representable as the following product

$$\left(\begin{array}{cc} 1 & -v^\top \\ \hline 0 & A_B^{-1} \end{array}\right) \cdot \left(\begin{array}{cc} c^\top & 0 \\ \hline A & b \end{array}\right) = \left(\begin{array}{cc} 1 & -v^\top \\ \hline 0 & A_B^{-1} \end{array}\right) \cdot \left(\begin{array}{ccc} c_B^\top & c_N^\top & 0 \\ \hline A_B & A_N & b \end{array}\right)$$

$$= \left(\begin{array}{cc|c} 0 & c_N^\top - c_B^\top A_B^{-1} A_N & -\langle c_B, x_B \rangle \\ \hline \mathrm{Id} & A_B^{-1} A_N & A_B^{-1} b \end{array}\right)$$

Note that the first equality is only true after reordering the columns.

It is easy to check that $v^\top = c_B^\top A_B^{-1}$.

Running Time

Each iteration takes $\mathcal{O}(mn)$ steps and there are at most $\binom{n}{m}$ basic feasible solutions. Therefore, the running time is finite, but may be exponential.

Running Time

Each iteration takes $\mathcal{O}(mn)$ steps and there are at most $\binom{n}{m}$ basic feasible solutions. Therefore, the running time is finite, but may be exponential.

There are methods that can solve the problem in polynomial time, but are numerical less stable than the simplex method. In practice, the simplex is quite fast and does not visit every basic feasible solution.

Running Time

Each iteration takes $\mathcal{O}(mn)$ steps and there are at most $\binom{n}{m}$ basic feasible solutions. Therefore, the running time is finite, but may be exponential.

There are methods that can solve the problem in polynomial time, but are numerical less stable than the simplex method. In practice, the simplex is quite fast and does not visit every basic feasible solution.

Nonetheless, there is an LP for which the simplex method might visit every of its 2^n basic feasible solutions. For $0 < \epsilon < \frac{1}{2}$ this is such an example

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \quad & x_n \\ \text{subject to} \quad & 0 \leqslant x_1 \leqslant 1 \\ & \epsilon x_i \leqslant x_{i+1} \leqslant 1 - \epsilon x_i \qquad \text{for all } i = 1, \ldots, n-1 \end{aligned}$$

Integer LP

If we want to minimize pseudo-Boolean functions, we want to add the constraint $x_i \in \mathbb{B}$ for each variable. This is equivalen to the following two constraints

$$0 \leqslant x_i \leqslant 1 \qquad\qquad x_i \in \mathbb{Z}$$

The first constraint fits well into the LP framework.

Integer LP

If we want to minimize pseudo-Boolean functions, we want to add the constraint $x_i \in \mathbb{B}$ for each variable. This is equivalen to the following two constraints

$$0 \leqslant x_i \leqslant 1 \qquad\qquad x_i \in \mathbb{Z}$$

The first constraint fits well into the LP framework.

An **Integer Linear Program (ILP)** is an LP for which we add an integer constraint to the variables. Minimizing an ILP is NP hard.

Integer LP

If we want to minimize pseudo-Boolean functions, we want to add the constraint $x_i \in \mathbb{B}$ for each variable. This is equivalen to the following two constraints

$$0 \leqslant x_i \leqslant 1 \qquad\qquad x_i \in \mathbb{Z}$$

The first constraint fits well into the LP framework.

An **Integer Linear Program (ILP)** is an LP for which we add an integer constraint to the variables. Minimizing an ILP is NP hard.

Nonetheless, some well behaving ILPs are solvable using the LP framework

Integer LP

If we want to minimize pseudo-Boolean functions, we want to add the constraint $x_i \in \mathbb{B}$ for each variable. This is equivalen to the following two constraints

$$0 \leqslant x_i \leqslant 1 \qquad\qquad x_i \in \mathbb{Z}$$

The first constraint fits well into the LP framework.

An **Integer Linear Program (ILP)** is an LP for which we add an integer constraint to the variables. Minimizing an ILP is NP hard.

Nonetheless, some well behaving ILPs are solvable using the LP framework

- Dropping the integer constraints on the variable leads to an LP

Integer LP

If we want to minimize pseudo-Boolean functions, we want to add the constraint $x_i \in \mathbb{B}$ for each variable. This is equivalen to the following two constraints

$$0 \leqslant x_i \leqslant 1 \qquad\qquad x_i \in \mathbb{Z}$$

The first constraint fits well into the LP framework.

An **Integer Linear Program (ILP)** is an LP for which we add an integer constraint to the variables. Minimizing an ILP is NP hard.

Nonetheless, some well behaving ILPs are solvable using the LP framework

- Dropping the integer constraints on the variable leads to an LP
- This LP might have an optimizer $x \in \mathbb{R}^n$

Integer LP

If we want to minimize pseudo-Boolean functions, we want to add the constraint $x_i \in \mathbb{B}$ for each variable. This is equivalen to the following two constraints

$$0 \leqslant x_i \leqslant 1 \qquad\qquad x_i \in \mathbb{Z}$$

The first constraint fits well into the LP framework.

An **Integer Linear Program (ILP)** is an LP for which we add an integer constraint to the variables. Minimizing an ILP is NP hard.

Nonetheless, some well behaving ILPs are solvable using the LP framework

- Dropping the integer constraints on the variable leads to an LP
- This LP might have an optimizer $x \in \mathbb{R}^n$
- If $x \in \mathbb{Z}^n$, the ILP is solved by x.

Solving ILP via LP

Let us assume that we have the following maximization problem

$$\max_{x \in \mathbb{Z}^n} \langle c, x \rangle$$
$$\text{subject to } Ax = b$$
$$x \geqslant 0$$

with $b \in \mathbb{Z}^n$ and $A \in \mathbb{Z}^{m \times n}$.

Solving ILP via LP

Let us assume that we have the following maximization problem

$$\max_{x \in \mathbb{Z}^n} \langle c, x \rangle$$
$$\text{subject to } Ax = b$$
$$x \geqslant 0$$

with $b \in \mathbb{Z}^n$ and $A \in \mathbb{Z}^{m \times n}$.

Let us assume that the LP that ignores the integer constraint will find a solution $x = x_B + x_N$ with

$$x_B = A_B^{-1} b \qquad\qquad x_N = 0.$$

Since $b \in \mathbb{Z}^m$, the ILP would be solved if $A_B^{-1} \in \mathbb{Z}^{m \times n}$. While this is not true in general, we can classify those matrices that give rise to ineger solutions.

Totally Unimodular Matrix

A matrix $A \in \mathbb{R}^{m \times n}$ is called **totally unimodular** if the determinant of any quadratic submatrix is either -1, 0 or $+1$.

Totally Unimodular Matrix

A matrix $A \in \mathbb{R}^{m \times n}$ is called **totally unimodular** if the determinant of any quadratic submatrix is either -1, 0 or $+1$.

Lemma 1. *If $A \in \mathbb{Z}^{m \times n}$ is totally unimodular, we have $A_B^{-1} \in \mathbb{Z}^{m \times n}$ for any basic set $B \subset \{1, \ldots, n\}$.*

Totally Unimodular Matrix

A matrix $A \in \mathbb{R}^{m \times n}$ is called **totally unimodular** if the determinant of any quadratic submatrix is either -1, 0 or $+1$.

Lemma 1. *If $A \in \mathbb{Z}^{m \times n}$ is totally unimodular, we have $A_B^{-1} \in \mathbb{Z}^{m \times n}$ for any basic set $B \subset \{1, \dots, n\}$.*

Proof. Let $M \in \mathbb{R}^{m \times m}$ be an arbitrary invertible matrix. Then we denote by $m^{\#}_{i,j} = \det(M_{j,i})$ the determinant of the submatrix of M after removing the j^{th} row and the i^{th} column of M. This creates a new matrix $M^{\#}$ and for the product of these matrices we have $MM^{\#} = \det(M) \cdot \mathrm{Id}$.

Totally Unimodular Matrix

A matrix $A \in \mathbb{R}^{m \times n}$ is called **totally unimodular** if the determinant of any quadratic submatrix is either -1, 0 or $+1$.

Lemma 1. *If $A \in \mathbb{Z}^{m \times n}$ is totally unimodular, we have $A_B^{-1} \in \mathbb{Z}^{m \times n}$ for any basic set $B \subset \{1, \ldots, n\}$.*

Proof. Let $M \in \mathbb{R}^{m \times m}$ be an arbitrary invertible matrix. Then we denote by $m_{i,j}^{\#} = \det(M_{j,i})$ the determinant of the submatrix of M after removing the j^{th} row and the i^{th} column of M. This creates a new matrix $M^{\#}$ and for the product of these matrices we have $MM^{\#} = \det(M) \cdot \mathrm{Id}$.
In other words, $A_B^{-1} = \pm A_B^{\#}$, which proves the lemma. □

Classification of Total Unimodularity

If $A \in \mathbb{R}^{m \times n}$ is totally unimodular, we have $A \in \{-1, 0, 1\}^{m \times n}$.

Classification of Total Unimodularity

If $A \in \mathbb{R}^{m\times n}$ is totally unimodular, we have $A \in \{-1,0,1\}^{m\times n}$.

Given $A \in \{-1,0,1\}^{m\times n}$, there is a very useful classification by Ghouila-Houri of totally unimodular matrices:

Theorem 3. *Iff for every selection of rows $a^i, i \in R$, there is a seperation $R = R^+ + R^-$ such that $\sum_{i\in R^+} a^i - \sum_{i\in R^-} a^i \in \{-1,0,1\}^n$, the matrix A is totally unimodular.*

Classification of Total Unimodularity

If $A \in \mathbb{R}^{m \times n}$ is totally unimodular, we have $A \in \{-1, 0, 1\}^{m \times n}$.

Given $A \in \{-1, 0, 1\}^{m \times n}$, there is a very useful classification by Ghouila-Houri of totally unimodular matrices:

Theorem 3. *Iff for every selection of rows $a^i, i \in R$, there is a seperation $R = R^+ + R^-$ such that $\sum_{i \in R^+} a^i - \sum_{i \in R^-} a^i \in \{-1, 0, 1\}^n$, the matrix A is totally unimodular.*

The matrix $\operatorname{div} \colon \mathbb{R}^{|E|} \to \mathbb{R}^{|E|}$ that we used for the maximal flow description is totally unimodular.

Classification of Total Unimodularity

If $A \in \mathbb{R}^{m \times n}$ is totally unimodular, we have $A \in \{-1, 0, 1\}^{m \times n}$.

Given $A \in \{-1, 0, 1\}^{m \times n}$, there is a very useful classification by Ghouila-Houri of totally unimodular matrices:

Theorem 3. *Iff for every selection of rows $a^i, i \in R$, there is a seperation $R = R^+ + R^-$ such that $\sum_{i \in R^+} a^i - \sum_{i \in R^-} a^i \in \{-1, 0, 1\}^n$, the matrix A is totally unimodular.*

The matrix $\operatorname{div}\colon \mathbb{R}^{|E|} \to \mathbb{R}^{|E|}$ that we used for the maximal flow description is totally unimodular.

If a matrix A is totally unimodular, $\begin{pmatrix} A & \mathrm{Id} \end{pmatrix}$ is totally unimodular.

Classification of Total Unimodularity

If $A \in \mathbb{R}^{m \times n}$ is totally unimodular, we have $A \in \{-1, 0, 1\}^{m \times n}$.

Given $A \in \{-1, 0, 1\}^{m \times n}$, there is a very useful classification by Ghouila-Houri of totally unimodular matrices:

Theorem 3. *Iff for every selection of rows $a^i, i \in R$, there is a seperation $R = R^+ + R^-$ such that $\sum_{i \in R^+} a^i - \sum_{i \in R^-} a^i \in \{-1, 0, 1\}^n$, the matrix A is totally unimodular.*

The matrix $\operatorname{div}\colon \mathbb{R}^{|E|} \to \mathbb{R}^{|E|}$ that we used for the maximal flow description is totally unimodular.

If a matrix A is totally unimodular, $\begin{pmatrix} A & \mathrm{Id} \end{pmatrix}$ is totally unimodular.

If a matrix A is totally unimodular, $A^\top$ is totally unimodular.

Dual LP

The Dual LP

Given a **primal LP** in canonical form

$$\text{(P)} \qquad \begin{aligned} &\max_{x \in \mathbb{R}^n} \langle c, x \rangle \\ &\text{subject to } Ax \leqslant b \\ &\qquad x \geqslant 0 \end{aligned}$$

its **dual LP** is

$$\text{(D)} \qquad \begin{aligned} &\min_{x \in \mathbb{R}^m} \langle b, y \rangle \\ &\text{subject to } A^\top y \geqslant c \\ &\qquad y \geqslant 0 \end{aligned}$$

The Dual LP

Given a **primal LP** in canonical form

$$
\text{(P)} \qquad \begin{aligned} &\max_{x \in \mathbb{R}^n} \langle c, x \rangle \\ \text{subject to } &Ax \leqslant b \\ &x \geqslant 0 \end{aligned}
$$

its **dual LP** is

$$
\text{(D)} \qquad \begin{aligned} &\min_{x \in \mathbb{R}^m} \langle b, y \rangle \\ \text{subject to } &A^\top y \geqslant c \\ &y \geqslant 0 \end{aligned}
$$

The problem is called dual due to the usage of the dual matrix $A^\top$ and the dual variable $y \in \mathbb{R}^m$.

Duality Theorem

Theorem 4. *Given the primal and dual LP as above, we have* $\langle c, x\rangle \leqslant \langle b, y\rangle$ *for feasible* $x \in \mathbb{R}^n$ *of (P) and feasible* $y \in \mathbb{R}^m$ *of (D). Moreover, we have equality for the optimizers* x^* *and* y^* *of the primal resp. dual problem.*

Proof.

Duality Theorem

Theorem 4. *Given the primal and dual LP as above, we have* $\langle c, x\rangle \leqslant \langle b, y\rangle$ *for feasible* $x \in \mathbb{R}^n$ *of (P) and feasible* $y \in \mathbb{R}^m$ *of (D). Moreover, we have equality for the optimizers* x^* *and* y^* *of the primal resp. dual problem.*

Proof. Let $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ be feasible, i.e., $Ax \leqslant b$ and $c \leqslant A^\top y$. Then

$$\langle c, x\rangle \leqslant \left\langle A^\top y, x\right\rangle = \langle y, Ax\rangle \leqslant \langle y, b\rangle$$

Duality Theorem

Theorem 4. *Given the primal and dual LP as above, we have $\langle c, x\rangle \leqslant \langle b, y\rangle$ for feasible $x \in \mathbb{R}^n$ of (P) and feasible $y \in \mathbb{R}^m$ of (D). Moreover, we have equality for the optimizers x^* and y^* of the primal resp. dual problem.*

Proof. Let $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ be feasible, i.e., $Ax \leqslant b$ and $c \leqslant A^\top y$. Then

$$\langle c, x\rangle \leqslant \left\langle A^\top y, x\right\rangle = \langle y, Ax\rangle \leqslant \langle y, b\rangle$$

Let us now assume that $x^* \in \mathbb{R}^n$ is an optimizers of (P) that is obtained via the simplex method. At the last step, the simplex tableau looks like this:

$$\begin{pmatrix} 1 & -v^\top \\ 0 & M \end{pmatrix} \cdot \begin{pmatrix} c^\top & 0 & 0 \\ A & \mathrm{Id} & b \end{pmatrix} = \begin{pmatrix} (c - A^\top v)^\top & -v^\top & -\langle v, b\rangle \\ * & * & * \end{pmatrix}$$

Duality Theorem

Theorem 4. *Given the primal and dual LP as above, we have $\langle c, x\rangle \leqslant \langle b, y\rangle$ for feasible $x \in \mathbb{R}^n$ of (P) and feasible $y \in \mathbb{R}^m$ of (D). Moreover, we have equality for the optimizers x^* and y^* of the primal resp. dual problem.*

Proof. Let $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ be feasible, i.e., $Ax \leqslant b$ and $c \leqslant A^\top y$. Then

$$\langle c, x\rangle \leqslant \left\langle A^\top y, x\right\rangle = \langle y, Ax\rangle \leqslant \langle y, b\rangle$$

Let us now assume that $x^* \in \mathbb{R}^n$ is an optimizers of (P) that is obtained via the simplex method. At the last step, the simplex tableau looks like this:

$$\begin{pmatrix} 1 & -v^\top \\ 0 & M \end{pmatrix} \cdot \begin{pmatrix} c^\top & 0 & 0 \\ A & \mathrm{Id} & b \end{pmatrix} = \begin{pmatrix} (c - A^\top v)^\top & -v^\top & -\langle v, b\rangle \\ * & * & * \end{pmatrix}$$

Since x^* is the minimizer we know that $\langle v, b\rangle = \langle x^*, c\rangle$, $c \leqslant A^\top v$ and $v \geqslant 0$. Thus v is a feasible dual variable that has the same cost as x^*. $\square$

Graph Cut as LP

The canonical form of the MaxFlow problem is

$$
\begin{aligned}
\max_{f \in \mathbb{R}^{|E|}, z \in \mathbb{R}} \quad & z \\
\text{subject to} \quad & \begin{pmatrix} \mathrm{Id} & 0 \\ \mathrm{div} & (\mathbf{1}_t - \mathbf{1}_s) \\ -\mathrm{div} & -(\mathbf{1}_t - \mathbf{1}_s) \end{pmatrix} \begin{pmatrix} f \\ z \end{pmatrix} \leqslant \begin{pmatrix} c \\ 0 \\ 0 \end{pmatrix} \\
& f, z \geqslant 0
\end{aligned}
$$

Maximal Flow and its Dual

The canonical form of the MaxFlow problem is

$$\begin{aligned} \max_{f\in\mathbb{R}^{|E|},z\in\mathbb{R}} \quad & z \\ \text{subject to} \quad & \begin{pmatrix} \mathrm{Id} & 0 \\ \mathrm{div} & (\mathbf{1}_t - \mathbf{1}_s) \\ -\mathrm{div} & -(\mathbf{1}_t - \mathbf{1}_s) \end{pmatrix} \begin{pmatrix} f \\ z \end{pmatrix} \leqslant \begin{pmatrix} c \\ 0 \\ 0 \end{pmatrix} \\ & f, z \geqslant 0 \end{aligned}$$

Its dual problem is

$$\begin{aligned} \min_{y\in\mathbb{R}^{|E|},\ell^-,\ell^+\in\mathbb{R}^{|V|}} \quad & \langle c, y \rangle \\ \text{subject to} \quad & \begin{pmatrix} \mathrm{Id} & \mathrm{div}^\top & -\mathrm{div}^\top \\ 0 & (\mathbf{1}_t - \mathbf{1}_s)^\top & -(\mathbf{1}_t - \mathbf{1}_s)^\top \end{pmatrix} \begin{pmatrix} y \\ \ell^- \\ \ell^+ \end{pmatrix} \geqslant \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ & y, \ell^-, \ell^+ \geqslant 0 \end{aligned}$$

Transposing Divergence

Note that the transposed of div is a linear mapping that maps a function $g\colon V \to \mathbb{R}$ to a function $\operatorname{div}^\top g\colon E \to \mathbb{R}$. In particular, we have

$$
\begin{aligned}
\operatorname{div}^\top g(i,j) &= \left\langle \delta_{i,j}, \operatorname{div}^\top g \right\rangle = \left\langle \operatorname{div} \delta_{i,j}, g \right\rangle \\
&= \sum_{u \in V} g(u) [\operatorname{div} \delta_{i,j}](u) \\
&= \sum_{u \in V} g(u) \left[\sum_{(u,v) \in E} \delta_{i,j}(u,v) - \sum_{(v,u) \in E} \delta_{i,j}(v,u) \right] \\
&= g(i) - g(j)
\end{aligned}
$$

Transposing Divergence

Note that the transposed of div is a linear mapping that maps a function $g\colon V \to \mathbb{R}$ to a function $\operatorname{div}^\top g\colon E \to \mathbb{R}$. In particular, we have

$$
\begin{aligned}
\operatorname{div}^\top g(i,j) &= \langle \delta_{i,j}, \operatorname{div}^\top g \rangle = \langle \operatorname{div} \delta_{i,j}, g \rangle \\
&= \sum_{u \in V} g(u) [\operatorname{div} \delta_{i,j}](u) \\
&= \sum_{u \in V} g(u) \left[\sum_{(u,v) \in E} \delta_{i,j}(u,v) - \sum_{(v,u) \in E} \delta_{i,j}(v,u) \right] \\
&= g(i) - g(j)
\end{aligned}
$$

In other words, we have $\operatorname{div}^\top = -\operatorname{Grad}$.

Integer Solution of MaxFlow's Dual

The dual of MaxFlow becomes therefore

$$\min_{y\in\mathbb{R}^{|E|},\ell\in\mathbb{R}^{|V|}} \quad \langle c, y\rangle$$

$$\text{subject to} \quad \begin{pmatrix} \mathrm{Id} & \mathrm{Grad} \\ 0 & -(\mathbf{1}_t - \mathbf{1}_s)^\top \end{pmatrix} \begin{pmatrix} y \\ \ell \end{pmatrix} \geqslant \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$y \geqslant 0$$

Integer Solution of MaxFlow's Dual

The dual of MaxFlow becomes therefore

$$\min_{y\in\mathbb{R}^{|E|},\ell\in\mathbb{R}^{|V|}} \quad \langle c, y\rangle$$

$$\text{subject to} \quad \begin{pmatrix} \mathrm{Id} & \mathrm{Grad} \\ 0 & -(\mathbf{1}_t - \mathbf{1}_s)^\top \end{pmatrix} \begin{pmatrix} y \\ \ell \end{pmatrix} \geqslant \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$y \geqslant 0$$

Since the constraint matrix is totally unimodular, we obtain

$$\min_{y\in\mathbb{Z}^{|E|},\ell\in\mathbb{Z}^{|V|}} \quad \langle c, y\rangle$$

$$\text{subject to} \quad \begin{aligned} y &\geqslant -\mathrm{Grad}(\ell) \\ \ell(s) &\geqslant \ell(t) + 1 \\ y &\geqslant 0 \end{aligned}$$

Rewriting MaxFlow's Dual

We make the following observations:

- Changing ℓ by a constant value does not change y. Hence $\ell(t) := 0$.

Rewriting MaxFlow's Dual

We make the following observations:

- Changing ℓ by a constant value does not change y. Hence $\ell(t) := 0$.
- For $\ell(s) > 1$, the cost w.r.t. $\left(\frac{\ell}{\ell(s)}, \frac{y}{\ell(s)}\right)$ is lower. Hence $\ell(s) := 1$.

Rewriting MaxFlow's Dual

We make the following observations:

- Changing ℓ by a constant value does not change y. Hence $\ell(t) := 0$.
- For $\ell(s) > 1$, the cost w.r.t. $\left(\frac{\ell}{\ell(s)}, \frac{y}{\ell(s)}\right)$ is lower. Hence $\ell(s) := 1$.
- If there exists a node $i \in V$ with $\ell(i) \notin \{0, 1\}$, we can decrease the cost by clipping ℓ at 0 resp. 1.

Hence, the dual of the MaxFlow is the MinCut:

$$\begin{array}{ll} \min\limits_{\ell \in \mathbb{B}^{|V|}} & \langle c, \max(0, -\operatorname{Grad} \ell) \rangle \\ \text{subject to} & \ell(s) = 1 \quad \ell(t) = 0 \end{array}$$

Ford-Fulkerson Revisited

The MaxFlow-MinCut theorem can be seen as special case of the duality theorem for LPs.

Ford-Fulkerson Revisited

The MaxFlow-MinCut theorem can be seen as special case of the duality theorem for LPs.

We saw that quadratic submodular pseudo-Boolean functions can be cast as a graph cut problem with positive edge weights. MaxFlow, the dual problem of MinCut, exploits this non-negativity.

Ford-Fulkerson Revisited

The MaxFlow-MinCut theorem can be seen as special case of the duality theorem for LPs.

We saw that quadratic submodular pseudo-Boolean functions can be cast as a graph cut problem with positive edge weights. MaxFlow, the dual problem of MinCut, exploits this non-negativity.

The Ford-Fulkerson algorithm starts with a basic feasible solution, namely the zero-flow.

Ford-Fulkerson Revisited

The MaxFlow-MinCut theorem can be seen as special case of the duality theorem for LPs.

We saw that quadratic submodular pseudo-Boolean functions can be cast as a graph cut problem with positive edge weights. MaxFlow, the dual problem of MinCut, exploits this non-negativity.

The Ford-Fulkerson algorithm starts with a basic feasible solution, namely the zero-flow.

Solving MaxFlow with the simplex method would lead to a setup where in each step at least $1 + |E| - |V|$ variables are 0. This is not necessarily the case for the Ford-Fulkerson method.

Ford-Fulkerson Revisited

The MaxFlow-MinCut theorem can be seen as special case of the duality theorem for LPs.

We saw that quadratic submodular pseudo-Boolean functions can be cast as a graph cut problem with positive edge weights. MaxFlow, the dual problem of MinCut, exploits this non-negativity.

The Ford-Fulkerson algorithm starts with a basic feasible solution, namely the zero-flow.

Solving MaxFlow with the simplex method would lead to a setup where in each step at least $1 + |E| - |V|$ variables are 0. This is not necessarily the case for the Ford-Fulkerson method.

Next lecture we will reformulate the general quadratic pseudo-Boolean optimization problem in order to minimize some non-submodular energies.

Literature

Linear Program

- Kantorovich, *"A New Method of Solving Some Classes of Extremal Problems"*, 1940, Dokl. Akad. Sci USSR (28), 211–214.
- Schrijver, *Combinatorial Optimization*, Chapter 5.

Simplex Method

- Dantzig, *Maximization of a Linear Function of Variables subject to Linear Inequalities*, 1947.
- Bland, *New Finite Pivoting Rules for the Simplex Method*, 1977, Mathematics of OR (2), 103–107.