Discrete Optimization for Image Analysis

Björn Andres
(Frank R. Schmidt)

2018-07-05

Simplex Method

Linear Programming

Maximal Flow Revisited

Given a network $G = (V, \mathcal{E}, c, s, t)$, both the edge capacity c and and a flow f can be written as vectors $c, f \in \mathbb{R}^{\mathcal{E}}$. The computation of the maximal flow can then be rewritten as

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

Here, $\operatorname{div} \colon \mathbb{R}^{|\mathcal{E}|} \to \mathbb{R}^{|V|}$ maps information from the edges to information on the vertices:

$$
[\operatorname{div} f]_i = \sum_{(i,j) \in \mathcal{E}} f_{(i,j)} - \sum_{(j,i) \in \mathcal{E}} f_{(j,i)}
$$

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.,

$$\left\{\begin{matrix}\min\\ \max\end{matrix}\right\}_{x \in \mathbb{R}^n} \quad \langle c, x \rangle$$

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

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

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

we can simplify the notation of LPs.

This will lead to two different representations:

- the **canonical form** that essentially uses $\leqslant$-constraints
- the **standard form** that essentially uses $=$-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

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

for a **constraint matrix** $A \in \mathbb{R}^{m \times n}$, a **constraint vector** $b \in \mathbb{R}^m, b \geqslant 0$ and a **cost vector** $c \in \mathbb{R}^n$.

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 one of the b_i in the equality constraint is negative, we can multiply the whole equation with -1 in order to obtain a non-negative value on the right hand side.

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 $x_i^+ - x_i^-$.

As a consequence, each LP can be easily transformed into either its canonical or its standard form.

Maximal Flow as LP

The canonical form of the MaxFlow problem is

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

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

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

subject to

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

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

The canonical form introduced **more constraints** (due to the original equality constraints).
The standard form introduced **more variables** (due to the original inequality constraints).

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**.

Since A is not square-shaped, we cannot compute $A^{-1}b$, but we can use a $m \times m$-submatrix and solve the linear equation for this submatrix.

To this end, we decompose the set $\{1, \ldots, n\}$ into two subsets:

$$\{1, \ldots, n\} = B + N \qquad \text{with } |B| = m.$$

Now 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 an x that satisfies $Ax = b$. Feasible x of that kind ($x_B \geqslant 0$) 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. 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. Note that the x_i for the $m - k$ indices will be 0.

Case 2: The k a_i are linearly dependent.
We have $0 = \sum_{i=1}^{k} \lambda_i a_i$ with at least one $\lambda_i > 0$. (Otherwise multiply the equation with -1)
For all ϵ we obtain now

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

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 1^{st} 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. 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. (Analogous to last theorem)

Case 2: The k a_i are linearly dependent.

Create like in the last theorem $x' = x^* - \epsilon\lambda$ and we have

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

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

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

Thus, $\langle c, \lambda \rangle = 0$ and ϵ as in the last theorem.
This proves the optimality of x' which 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.

Simplex Method

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.

We will later see that Step 1 can be reduced to solving an LP to which we know a basic feasible solution.
Thus, Step 2 is the important step.

Step 1 is sometimes referred to as **Phase I** of the simplex method.
The remaining steps are called **Phase II**.

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 $y_{*j} = A_B^{-1} a_j$. We will now choose a specific $j \notin B$.

For a small $\epsilon \geqslant 0$, we have $b = \epsilon a_j + \sum_{i \in B} (x_i - \epsilon y_{ij}) a_i$, which creates a feasible, but not basic feasible solution x_ϵ [$(x_\epsilon)_j = \epsilon$ and $(x_\epsilon)_i = x_i - \epsilon y_{ij}$].

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

In other words, for every $j \notin B$ we find at least one $i \in B$ such that $\overline{B} = B - \{i\} + \{j\}$ provides us with a basic feasible solution, i.e.,

$$x_B = A_B^{-1} b \geqslant 0 \qquad x_{\overline{B}} = A_{\overline{B}}^{-1} b \geqslant 0$$

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 with its $x = (x_B, x_N)$, we have $x_N = 0$ and $x_B = A_B^{-1} b$.

Any other solution y with $Ay = b$ can also be written as $y = (y_B, y_N)$ with $y_N \geqslant 0$, but $y_N \neq 0$ and we have

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

The cost function becomes

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

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.

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**.

It contains

- an altered **cost vector**
- the accumulated **costs so far**
- an altered **description of the constraints**

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

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

The left hand side is the actual tableau. The right hand side shows the tableau if we assume that the first m indices belong to B and the remaining $n-m$ belong to N. To achieve this we need to permute the columns.

The **first row** contains the changed cost vector and the negative accumulated costs. Along the B-entries we always have zeros.

The **lower tableau** has all the information to obtain the basic feasible solution x_B as well as the indices i and j to alter B (if necessary).

Note that the right column of the lower tableau encodes the non-zero entries of the current basic feasible solution.

Simplex Tableau - Constraints Update

Let us assume we have the following simplex tableau

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

If the first row has a positive entry (at position $j \in N$), we can improve the solution by adding j to B. The $i \in B$ that we have to remove from B is selected as $i \in \operatorname{argmin} \left\{ \frac{(A_B^{-1} b)_i}{(A_B^{-1} a^j)_i} \middle| (A_B^{-1} a^j)_i > 0 \right\}$.

Now pivot the j^{th} column of the lower tableau, i.e, perform Gaussian elimination until the j^{th} column of the constraint matrix is the i^{th} unit vector.

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

Note that the computation of the new basic feasible operation only took m row operations. This is much faster than m^2 row operation we would have needed to solve the linear equation from scratch.

Simplex Tableau - Cost Update

The simplex tableau is

$$\left(\begin{array}{c|c} c^\top - c_B^\top A_B^{-1} A & * \\ \hline * & * \end{array}\right)$$

with the first n entries of the first row being the transposed of the vector $c - (A_B^{-1}A)^\top c_B$. How can we update this if c is not available anymore?

Due to the pivot operation, we do not change the range of the matrix and we know that the update of the first row can be done by using a linear combination of the constraint rows. The only way to achieve this is to expand the pivoting operation to the first row.

By doing so, we subtract the product of the positice entry of $c_N^\top - c_B^\top A_B^{-1} A_N$ and the new value $(x_{\overline{B}})_i$ from the upper right corner:

$$-\langle x_B, c_B \rangle - (c_N^\top - c_B^\top A_B^{-1} A_N)_i \cdot (x_{\overline{B}})_i = -\langle x_{\overline{B}}, c_{\overline{B}} \rangle$$

Thus by pivoting, we also updated $-\langle x_B, c_B \rangle$ correctly.

Simplex Tableau - Matrix Multiplication

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

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

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

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

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

Note that if the simplex method terminates we have

$$c_N^\top - v^\top A_N = c_N^\top - c_B^\top A_B^{-1} A_N \leqslant 0$$

Finding an Initial Feasible Solution

We saw how to solve the problem

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

if an initial basic feasible solution is given. How can we find such a solution?

To this end we solve

$$\begin{aligned} \min_{x \in \mathbb{R}^n, y \in \mathbb{R}^m, x, y \geqslant 0} \quad & \langle 1, y \rangle \\ \text{subject to} \quad & Ax + y = b \end{aligned}$$

with the initial solution $(x, y) = (0, b)$. If the minimum is 0, x^* is an initial feasible solution for the original problem. Otherwise, an initial feasible solution does not exist, *i.e.* $Ax = b \wedge x \geqslant 0$ defines the empty set.

Practically, we can do this by adding the cost function of Phase I to our tableau until convergence and continue then with the altered cost function. The pivot operation of the first phase will automatically change the cost function as needed.

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 method is often 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

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

Dual LP

The Dual LP

Given a **primal LP** in canonical form

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

its **dual LP** is

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

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 3. *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 $y \in \mathbb{R}^m$ 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. Since the standard form assumes equalities with positive entries on the right hand side, we have to multiply some rows with -1. Given a matrix A or a vector b, we denote the result of this multiplication with A_σ resp. b_σ.
At the last step, the simplex tableau looks like this:

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

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

Unbounded Problems

Theoretically, it is possible that we formulate an LP such that the maximum is ∞. This happens if the set defined by the constraints is not compact, but unbounded instead.

By the duality theorem we know that it is enough to find a feasible for the dual problem in order to prove that the problem is bounded and can be optimized by a basic feasible solution.

There are methods that make use of the duality theorem by simultaneously solving the primal and the dual problem. Since the two problems constrain each other, one can even terminate such methods if the gap between the primal solution and the dual solution is small enough.

Methods that use primal and dual variables simultaneously are called **primal-dual methods**.

Graph Cut as LP

Maximal Flow and its Dual

The canonical form of the MaxFlow problem is

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

subject to

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

$$f, z \geqslant 0$$

Its dual problem is

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

subject to

$$\begin{pmatrix} \mathrm{Id} & \mathrm{div}^\top & -\mathrm{div}^\top \\ 0 & (\mathbf{1}_t - \mathbf{1}_s)^\top & -(\mathbf{1}_t - \mathbf{1}_s)^\top \end{pmatrix} \begin{pmatrix} y \\ \ell^- \\ \ell^+ \end{pmatrix} \geqslant \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$y, \ell^-, \ell^+ \geqslant 0$$

Transposing Divergence

Note that the transposed of div is a linear mapping that maps a vector $g \in \mathbb{R}^{|V|}$ to a vector $\operatorname{div}^\top g \in \mathbb{R}^{|\mathcal{E}|}$. Using the unit vector $\mathbf{1}_{(i,j)} \in \mathbb{R}^{|\mathcal{E}|}$, we obtain

$$\begin{aligned}
[\operatorname{div}^\top g]_{(i,j)} &= \left\langle \operatorname{div}^\top g, \mathbf{1}_{(i,j)} \right\rangle = \left\langle g, \operatorname{div} \mathbf{1}_{(i,j)} \right\rangle \\
&= \sum_{u \in V} g_u [\operatorname{div} \mathbf{1}_{(i,j)}]_u \\
&= \sum_{u \in V} g_u \left[\sum_{(u,v) \in \mathcal{E}} [\mathbf{1}_{(i,j)}]_{(u,v)} - \sum_{(v,u) \in \mathcal{E}} [\mathbf{1}_{(i,j)}]_{(v,u)} \right] \\
&= g_i - g_j \qquad (\text{iff } (i,j) \in \mathcal{E})
\end{aligned}$$

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

Exploring MaxFlow's Dual

The dual of MaxFlow becomes therefore

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

subject to

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

$$y \geqslant 0$$

Or in more explicit form

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

subject to

$$y = \max(0, -\mathrm{Grad}(\ell))$$

$$\ell(s) \geqslant \ell(t) + 1$$

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.
- One can show (beyond the scope of this lecture) that the optimum of ℓ only contains integers, *i.e.*, $\ell(i) \in \{0, 1\}$.

Hence, the dual of the MaxFlow is the MinCut:

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

Ford-Fulkerson vs. Simplex

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 or via Ford-Fulkerson creates always feasible flows.

Ford-Fulkerson updates the flow along a path, while the simplex method may update the flow on all edges during the Pivot-operation.

Checking whether the problem is solved is stored explicitly in the simplex tableau. Ford-Fulkerson needs to travers the network in order to detect optimality.

On a CPU, Ford-Fulkerson's method is faster if we use dynamic trees.
The simplex method can be easily implemented on a parallel computation architecture.

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.