Max-Flow and Min-Cut Problems

Alberto Santini

Fall 2024

Many examples in these lecture notes are adapted from popular books:

- Alexander Schrijver (1998). *Theory of linear and integer programming.* Wiley. ISBN: 0-471-98232-6.
- Vašek Chvátal (1983). *Linear Programming.* W.H. Freeman and Company. ISBN: 0-716-71195-8.
- Laurence Wolsey (2020). *Integer Programming.* 2nd Edition. Wiley. ISBN: 978-1-119-60653-6.
- Silvano Martello and Paolo Toth (1990). *Knapsack Problems: algorithms and computer implementations.* Wiley. ISBN: 978-0-471-92420-3.

1 Total unimodularity

Consider an integer programme (IP) and let x^* be an optimal solution and z^* the optimal objective value. Also, consider the IP's continuous relaxation and let x^*_{cont} be an optimal solution and z^*_{cont} the optimal objective value. We have remarked in previous lecture notes that z^* and z^*_{cont} can have very different values, i.e., z^*_{cont} does not necessarily provide accurate information on the unknown value of z^*. Analogously, solution x^*_{cont} can differ significantly from the unknown x^*.

In this section, however, we introduce a property of IPs which, if it holds, ensures that $x^*_{\text{cont}} = x^*$ and, therefore, $z^*_{\text{cont}} = z^*$. In other words, this property implies that the optimal solution of the continuous relaxation of an IP is integer. To introduce the property, we need some preliminary definitions.

Definition 1. Consider a matrix $B \in \mathbb{Z}^{m \times m}$. We say that B is **unimodular** if $\det(B) \in \{-1, +1\}$.

Remark that the definition of unimodularity only refers to square integer-valued matrixes. We will limit ourselves to integer-valued matrices for the rest of this section. In the following definition, however, we extend the notion of unimodularity to rectangular matrices.

Definition 2. A matrix $A \in \mathbb{Z}^{m \times n}$ is called **totally unimodular** (TUM) if all square submatrices B of A are either unimodular or have $\det(B) = 0$.

With the above definitions in mind, we introduce the main result on unimodularity.

Theorem 1. *Consider the following IP:*

$$\min \quad \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_{ij} = b_i \qquad \forall i \in \{1, \dots, n\} \tag{2}$$

$$x_j \geq 0 \text{ and integer} \qquad \forall j \in \{1, \dots, n\} \tag{3}$$

with $c = (c_1, \dots, c_n) \in \mathbb{R}^n$, $A = (a_{ij})_{\substack{i=1,\dots,m \\ j=1,\dots,n}} \in \mathbb{Z}^{m \times n}$, *and* $b = (b_1, \dots, b_m) \in \mathbb{Z}^m$. *Remark that this formulation only contains equality constraints and that both the constraint matrix coefficients and the right-hand-side coefficients are integer numbers.*

Then, if A is TUM, the vertices of the polytope defining the feasible region of the continuous relaxation are integers. In other words, the continuous relaxation of (1)–(3) *admits an optimal integer solution, which is, consequently, optimal for the original IP.*

Theorem 1 gives a sufficient condition for an IP to be "easy". Whenever an IP satisfies the hypotheses of the theorem, we need not solve it as an IP (e.g., via the branch-and-bound algorithm) but can instead solve it as a Linear Programme (LP), with the guarantee that we will nonetheless obtain an integer solution. The disadvantage of Theorem 1 is that verifying that an IP satisfies the TUM hypothesis is not trivial: we would have to prove that any square submatrix B of A is such that $\det(B) \in \{-1, 0, 1\}$. The following theorem, then, provides an alternative characterisation of TUM matrices. The hypotheses of the theorem might seem as difficult to prove as the TUM definition. However, as we will see in the rest of these notes, verifying them for some practically relevant IPs can be easy.

Theorem 2. *A matrix $A \in \mathbb{Z}^{m \times n}$ is TUM if all the following conditions hold:*

1. *All entries of A are either -1, 0, or $+1$.*
2. *At most two entries in each column of A are non-zero.*
3. *It is possible to partition the rows of A into two groups indexed respectively by $I_1, I_2 \subseteq \{1, \dots, m\}$ such that*
   - *If two non-zero entries in a column have the same sign, one belongs to a row indexed from I_1, and the other belongs to a row indexed from I_2.*
   - *If two non-zero entries in a column have opposite signs, they either both belong to rows indexed from I_1 or from I_2.*

Before using this theorem, we state one last result, which is an immediate consequence of the definition of a TUM matrix.

Theorem 3. *Let $A \in \mathbb{Z}^{m \times n}$ be TUM. Then, the matrix obtained by vertically juxtaposing A and the identity matrix $I \in \mathbb{Z}^{n \times n}$ is also TUM. Furthermore, the results of Theorem 1 are still valid for the new matrix, even if the constraints associated with the identity matrix are inequalities.*

In the following sections, we will introduce two new optimisation problems and show how TUM results apply to them. We will also introduce the concept of Linear Programming duality, which, for the interested student, is further developed in the additional lecture notes available on the website.

2 The max-flow problem

For a dramatic introduction to the max-flow problem, consider a large country at war fighting a desperate battle for survival. To resist and, perhaps, even prevail, the country must ensure that appropriate resources converge at the front, where the main battle is taking place. They must ensure that soldiers, weapons, foodstuffs, and fuel, among others, arrive at the crucial point where the country's destiny is being decided. For example, the country produces heavy

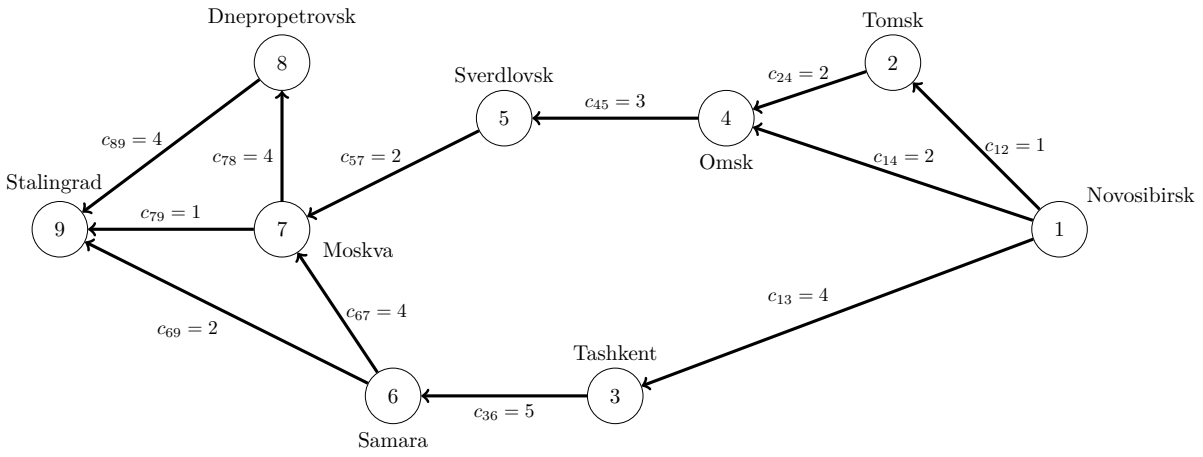

Figure 1: Example of a supply chain network used to move goods from node 1 to node 9.

weapons in a mineral-rich region and must use the railway network to efficiently ship them to the front. There are multiple paths that they can use, passing through various cities linked by the railway. The available number of trains, cranes, and other supply chain factors limit the maximum weekly shipments on each rail segment. The objective is to ensure that the largest possible number of shipments flow from the production site to the battlefront.

To model this problem, the planning staff uses a directed graph $G = (V, A)$ in which the vertex set represents the cities and each arc represents a railway segment. The goods originate at a **source** node $s \in V$ and must reach a **sink** node $t \in V \setminus \{s\}$. It is often assumed that all arcs incident to the source are outgoing, i.e., they are of type (s, i) from some $i \in V \setminus \{s\}$ and there are no arcs of type (i, s). Analogously, all arcs incident to the sink are incoming, i.e., of type (i, t) for some $i \in V \setminus \{t\}$ and there are no arcs of type (t, i). For a given arc $(i, j) \in A$, we denote its **capacity** with $c_{ij} \in \mathbb{N}$. This number represents the maximum flow of goods along the arcs. In our example, it is the maximum number of weekly weapons shipments on a rail link. Finally, we also assume that there is no arc (s, t) because this arc can be trivially used to send c_{st} units of flow.

The problem of determining the maximum flow from source to sink in G is known as the **Max-flow Problem**. Figure 1 shows an example graph with source 1 and sink 9. In this example, the total capacity of the arcs outgoing from the source and incoming to the sink is 7. However, the maximum flow from source to sink is 6 because of bottlenecks in the rest of the network. Figure 2 shows a possible optimal solution to the max-flow problem in the example. In the figure, values x_{ij} represent the amount of shipments flowing along arcs (i, j).

To model the Max-flow Problem, we use variables $x_{ij} \in \mathbb{N}$ for each arc $(i, j) \in A$. These variables will hold the quantity of goods flowing along each arc. To ease the notation, we consider the following sets:

$$\delta^+(i) = \{j \in V \ : \ (i, j) \in A\} \qquad \forall i \in V$$
$$\delta^-(i) = \{j \in V \ : \ (j, i) \in A\} \qquad \forall i \in V.$$

An IP for this problem reads as follows:

$$\max \quad \sum_{j \in \delta^+(s)} x_{sj} \tag{4a}$$

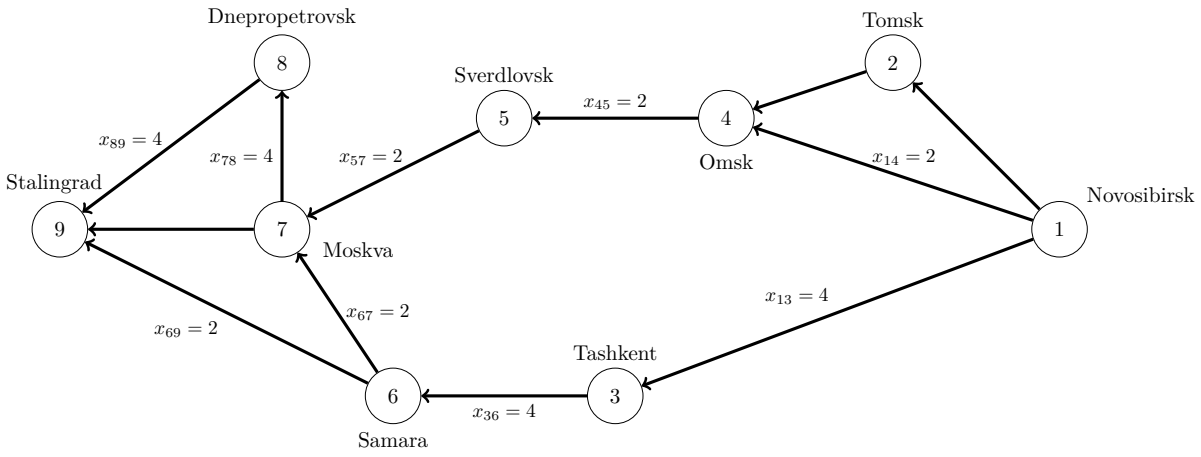

Figure 2: An optimal solution for the max-flow problem of Figure 1.

$$\text{subject to} \quad \sum_{j \in \delta^-(i)} x_{ji} = \sum_{j \in \delta^+(i)} x_{ij} \qquad \forall i \in V \setminus \{s, t\} \tag{4b}$$

$$x_{ij} \leq c_{ij} \qquad \forall (i, j) \in A \tag{4c}$$

$$x_{ij} \geq 0 \text{ and integer} \qquad \forall (i, j) \in A. \tag{4d}$$

The objective function (4a) maximises the flow outgoing from the source. Because all flow going out of the source must reach the sink, this corresponds to the total flow sent from the source to the sink. Alternatively, the objective function could also maximise the flow received at the sink, i.e.,

$$\max \quad \sum_{i \in \delta^-(t)} x_{it}.$$

Constraints (4b) are flow conservation constraints and ensure that the flow sent by all intermediate nodes is the same as the flow they receive. This constraint ensures that no flow disappears at an intermediate node or is created there. The source and the sink are excluded from the quantifiers of these constraints because they do not obey flow balance: flow only leaves the source and only reaches the sink. Finally, constraints (4c) ensure that each arc's capacity is respected.

3 Max-flow and unimodularity

Consider model (4a)–(4d). Due to Theorem 3, if the coefficient matrix of constraint (4b) is TUM, the entire model satisfies the hypotheses of Theorem 1 because the coefficients of constraint (4c) form the identity matrix. To show that the coefficients of (4b) form a TUM matrix, we apply Theorem 2. In particular, this matrix has one row per vertex $i \in V$ different from s and t and one column per arc. Bringing all variables on the left-hand side, we can rewrite the constraints as follows:

$$\sum_{j \in \delta^-(i)} x_{ji} - \sum_{j \in \delta^+(i)} x_{ij} = 0 \quad \forall i \in V \setminus \{s, t\}.$$

The coefficients, therefore, take values -1, 0, or $+1$. Furthermore, in the columns relative to arcs of type (s, i), there is a single non-zero coefficient with value $+1$ at row i. In the columns relative to arcs of type (i, t), there is a single non-zero coefficient with value -1 at row i. All other columns have exactly two non-zero coefficients: a -1 at row i and a $+1$ at row j. Because there

is no column with two positive or two negative coefficients, it is sufficient to choose $I_1 = V \setminus \{s, t\}$ (i.e., all row indices) and $I_2 = \emptyset$ to satisfy the last hypothesis of Theorem 2.

As a consequence of the above reasoning, solving the continuous relaxation of a max-flow problem is guaranteed to provide an integer solution that is also optimal for the original integer problem.

4 The min-cut problem

In this section, we present another problem on a graph. Like the max-flow problem, the **Min-Cut Problem** originated during wartime military operations. Imagine you are an enemy of the country presented in Section 2. You want to disrupt the flow of goods from the origin to the sink of the network in Figure 1. The values of c_{ij} now represent the cost of destroying link (i, j). Your objective is to completely stop the flow from the origin to the sink at the lowest possible cost. To do so, you must destroy links (i.e., arcs in the graph) until the maximum flow from source to sink is zero and the graph is completely disconnected. To present this problem formally, let us introduce some notation.

Definition 3. Given a directed graph $G = (V, A)$, a source vertex $s \in V$ and a sink $t \in V \setminus \{s\}$, we define an ***s-t* partition** a pair of sets $S, T \subset V$ such that (i) $S \cap T = \emptyset$, (ii) $S \cup T = V$, (iii) $s \in S$, and (iv) $t \in T$.

Definition 4. Given an s-t partition (S, T) of a directed graph $G = (V, A)$, we define its **crossing arcs** as

$$A_{S,T} = \{(i, j) \in A \ : \ i \in S, \ j \in T\}.$$

If we denote with $c_{ij} \in \mathbb{N}$ the cost associated with each arc $(i, j) \in A$, the **cost** of the partition is

$$c_{S,T} = \sum_{(i,j) \in A_{S,T}} c_{ij}.$$

The min-cut problem asks to find an s-t partition (S, T) with minimum possible cost $c_{S,T}$. To write an IP for the min-cut problem, we consider two sets of variables.

- $y_{ij} \in \{0, 1\}$ for each arc $(i, j) \in A$. Variable y_{ij} will take the value 1 if and only if arc (i, j) is a crossing arc (i.e., $i \in S$ and $j \in T$).
- $z_i \in \{0, 1\}$ for each vertex $i \in V \setminus \{s, t\}$. Variable z_i will take the value 1 if and only if $i \in S$; conversely, if $i \in T$, $z_i = 0$. Remark that we need not define variables z for indices s and t because, by definition, $s \in S$ and $t \in T$.

An IP for the min-cut problem reads as follows.

$$\begin{aligned}
\min \quad & \sum_{(i,j) \in A} c_{ij} y_{ij} && && \text{(5a)} \\
\text{subject to} \quad & y_{ij} \geq z_i - z_j && \forall (i, j) \in A, \ i \neq s, \ j \neq t && \text{(5b)} \\
& y_{sj} \geq 1 - z_j && \forall j \in \delta^+(s) && \text{(5c)} \\
& y_{it} \geq z_i && \forall i \in \delta^-(t) && \text{(5d)} \\
& y_{ij} \in \{0, 1\} && \forall (i, j) \in A && \text{(5e)} \\
& z_i \in \{0, 1\} && \forall i \in V \setminus \{s, t\}. && \text{(5f)}
\end{aligned}$$

The objective function (5a) minimises the cost of the crossing arcs. Inequality (5b) links the y and the z variables, forcing $y_{ij} = 1$ whenever $z_i = 1$ and $z_j = 0$. In other words, it ensures that

Value of z_i	Value of z_j	Constraint	Remarks
1	0	$y_{ij} \geq 1 - 0 = 1$	It forces $y_{ij} = 1$.
1	1	$y_{ij} \geq 1 - 1 = 0$	Trivially true ($y_{ij} \in \{0, 1\}$).
0	0	$y_{ij} \geq 0 - 0 = 0$	Trivially true ($y_{ij} \in \{0, 1\}$).
0	1	$y_{ij} \geq 0 - 1 = -1$	Trivially true ($y_{ij} \in \{0, 1\}$).

Table 1: Validity check for constraint (5b).

(i, j) is considered a crossing arc whenever $i \in S$ and $j \in T$. Before explaining how it achieves this objective, let us remark that we do not need an analogous constraint that forces $y_{ij} = 0$ whenever (i, j) is *not* a crossing arc. The reason is that y_{ij} is penalised in the objective function and, therefore, y_{ij} will take the value 0 as long as no constraint prevents it. We must check two things to verify that constraint (5b) works correctly. First, it forces $y_{ij} = 1$ when $z_i = 1$ and $z_j = 0$. Second, it is moot (i.e., it does not enforce anything) in all other cases. The reason is that this constraint is always present in the model and must be valid in all cases. No matter the values of z, the constraint must be valid; in the case of $z_i = 1$ and $z_j = 0$, it must moreover enforce $y_{ij} = 1$. To check the above properties, in Table 1, we rewrite the constraint for the four possible assignments of values to the z variables.

Because z_i is not defined when $i \in \{s, t\}$, we must ensure that $i \neq s$ and $j \neq t$ in the quantifier of (5b). Constraints (5c) and (5d) take care of the cases involving s and t, respectively.

Consider a vertex $j \in \delta^+(s)$. If $j \in S$, arc (s, j) is not a crossing arc because s and j are in S. If $j \in T$, arc (s, j) is a crossing arc by definition. Constraint (5c) makes sure that this property holds. When $j \in T$, $z_j = 0$ and the constraint becomes $y_{sj} \geq 1 - 0 = 1$, i.e., it forces $y_{sj} = 1$. When $j \in S$, $z_j = 1$ and the constraint is moot: $y_{sj} \geq 1 - 1 = 0$.

Analogously, consider a vertex $i \in \delta^-(t)$. If $i \in T$, arc (i, t) is not a crossing arc because i and t are in T. If $i \in S$, arc (i, t) is a crossing arc. Constraint (5d) enforces this property. When $i \in S$, $z_i = 1$ and the constraint becomes $y_{it} \geq 1$ forcing $y_{it} = 1$. When $i \in T$, $z_i = 0$ and the constraint is moot: $y_{it} \geq 0$.

After presenting the max-flow and the min-cut problems, we take a short break and give an introduction to linear programming duality. Interested students will find more information on this fascinating topic in the extra lecture notes on the website.

5 Duality

Consider an LP in standard form, which we call the **primal LP**:

$$\max \quad \sum_{j=1}^{n} c_j x_j \tag{6a}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad \forall i \in \{1, \ldots, m\} \tag{6b}$$

$$x_j \geq 0 \qquad \forall j \in \{1, \ldots, n\}. \tag{6c}$$

The following LP is the **dual LP** of (6a)–(6c).

$$\min \quad \sum_{i=1}^{m} b_i y_i \tag{7a}$$

$$\text{subject to} \quad \sum_{i=1}^{m} a_{ij} y_i \geq c_j \qquad \forall j \in \{1, \dots, n\} \tag{7b}$$

$$y_i \geq 0 \qquad \forall i \in \{1, \dots, m\}. \tag{7c}$$

The dual is obtained from the primal in the following way.

1. The primal has n variables and m constraints; the dual has m variables and n constraints.
2. The primal is a maximisation problem; the dual is a minimisation one.
3. The primal objective function coefficients become the dual constraints' right-hand sides.
4. Conversely, the primal constraints' right-hand sides become the dual objective function coefficients.
5. The dual constraint matrix is the transpose of the primal one.
6. Both problems feature non-negative continuous variables.

The motivation to devise the dual LP and the reason why it takes this form are beyond the scope of this lecture but are available in the LP Duality extra lecture notes. In the following, we list some fundamental properties of primal-dual LPs that highlight the importance of the dual.

Property 1. The dual of the dual is the primal. If we wrote (7a)–(7c) in standard form, took its dual, and brought it back in standard form, we would obtain (6a)–(6c).

Property 2 (Weak duality)**.** Assume that the primal problem is feasible and bounded. The objective value of any dual feasible solution is an upper bound (also known as a *dual bound*) for the objective value of the optimal primal solution.

Proof. For any primal feasible solution $(x_1, \dots, x_n)$ and any dual feasible solution $(y_1, \dots, y_m)$,

$$\sum_{j=1}^{n} c_j x_j \overset{\text{by (7b)}}{\leq} \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \overset{\text{by (6b)}}{\leq} \sum_{i=1}^{m} b_i y_i. \tag{8}$$

Therefore, the objective value of a dual feasible solution is not smaller than that of a primal feasible solution and, in particular, of a primal optimal solution. □

Property 3 (Strong duality)**.** If the primal problem has an optimal solution $(x_1^*, \dots, x_n^*)$, then the dual also has an optimal solution, say $(y_1^*, \dots, y_m^*)$. Moreover, the two optimal objective values coincide, i.e.,

$$\sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^*. \tag{9}$$

Property 4. The duality theorem states that if the primal has an optimal solution, so does the dual. However, a stronger if-and-only-if relation holds. It is also true that if the dual has an optimal solution, then the primal has one with the same objective value. This result is a direct consequence of Property 1 and Property 3.

Property 5. If the primal problem is unbounded, the dual must be infeasible. If the dual admitted a solution $(y_1, \dots, y_m)$, by eq. (8) quantity $\sum_{i=1}^{m} b_i y_i$ would place a bound on the objective value of the primal, contradicting the hypothesis that the primal is unbounded. Because the dual of the dual is the primal, we can swap words "primal" and "dual" in the above reasoning and also conclude that if the dual is unbounded, then the primal must be infeasible.

While we established above the "unbounded ⇒ infeasible" primal-dual implication, the reverse is untrue. If the primal is infeasible, then we know for sure that the dual cannot admit an optimal solution: If this were not the case, we would obtain an optimal solution for the primal using Property 3 and the fact that the dual of the dual is the primal. However, both options are possible for the dual: it can be either infeasible or unbounded.

Exercise 1. Consider the following LP:

$$\begin{aligned} \max \quad & 2x_1 - x_2 \\ \text{subject to} \quad & x_1 - x_2 \leq 1 \\ & -x_1 + x_2 \leq -2 \\ & x_1,\ x_2 \geq 0. \end{aligned}$$

Does this LP admit an optimal solution? If not, is it infeasible or unbounded? What about its dual?

5.1 Writing down the dual: practical aspects

We have, until now, considered LPs in standard form. Although any LP can be converted into an equivalent LP in standard form, it would be convenient to write down the dual of a generic LP without explicitly performing the conversion.

We start by writing down a generic form for linear programmes. In this form, we consider two types of variables: $x_1, \ldots, x_n$ are non-negative variables, i.e., $x_j \geq 0\ \forall j \in \{1, \ldots, n\}$; $u_1, \ldots, u_l$ are free variables, i.e., $u_j \in \mathbb{R}\ \forall j \in \{1, \ldots, l\}$. We also consider two types of constraints: "$\leq$"-inequalities and equalities. With these assumptions, a generic LP can be written as

$$\max \quad \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{l} p_j u_j \tag{10a}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j + \sum_{j=1}^{l} d_{ij} u_j \leq b_i \qquad \forall i \in \{1, \ldots, m\} \tag{10b}$$

$$\sum_{j=1}^{n} e_{ij} x_j + \sum_{j=1}^{l} f_{ij} u_j = h_i \qquad \forall i \in \{1, \ldots, q\} \tag{10c}$$

$$x_j \geq 0 \qquad \forall j \in \{1, \ldots, n\} \tag{10d}$$

$$u_j \in \mathbb{R} \qquad \forall j \in \{1, \ldots, l\}. \tag{10e}$$

In the above formulation, the c_j's, p_j's, a_{ij}'s, d_{ij}'s, b_i's, e_{ij}'s, f_{ij}'s, and h_i's are real parameters. The advantage of formulation (10a)–(10e) is that it does not require any additional artificial variable and can express any LP using the x's and the u's.

One can model minimisation problems by flipping the sign in the objective coefficients. Similarly, "$\geq$"-inequalities require flipping the signs on their left- and right-hand sides. Finally, note that using only non-negative and free variables is not restrictive because all other variable bounds can be modelled as inequalities of type (10b). For example, bound $x_j \geq 5$ could be modelled as $-x_j \leq -5$.

We consider two sets of dual variables to write the dual LP of (10a)–(10e). First, we introduce one non-negative variable y_i for each "$\leq$"-inequality, i.e., $y_i \geq 0\ \forall i \in \{1, \ldots, m\}$. Next, we

introduce one free variable w_i for each equality, i.e., $w_i \in \mathbb{R}\ \forall i \in \{1, \dots, q\}$. The dual then reads as follows:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m} b_i y_i + \sum_{i=1}^{q} h_i w_i && && \text{(11a)}\\
\text{subject to} \quad & \sum_{i=1}^{m} a_{ij} y_i + \sum_{i=1}^{q} e_{ij} w_i \geq c_j && \forall j \in \{1, \dots, n\} && \text{(11b)}\\
& \sum_{i=1}^{m} d_{ij} y_i + \sum_{i=1}^{q} f_{ij} w_i = p_j && \forall j \in \{1, \dots, l\} && \text{(11c)}\\
& y_i \geq 0 && \forall i \in \{1, \dots, m\} && \text{(11d)}\\
& w_i \in \mathbb{R} && \forall i \in \{1, \dots, q\}. && \text{(11e)}
\end{aligned}$$

As we can see, each non-negative primal variable corresponds to a "$\geq$"-inequality in the dual, and each free primal variable corresponds to an equality in the dual.

Exercise 2. Write down the dual of the following linear programme:

$$\begin{aligned}
\max \quad & 42x_2 - 30x_3 && \text{(12a)}\\
\text{subject to} \quad & x_1 - x_2 + x_3 - x_4 = 0 && \text{(12b)}\\
& x_1 + x_3 - x_4 \leq 5 && \text{(12c)}\\
& 5x_2 + x_3 - 5x_4 = -1 && \text{(12d)}\\
& x_1 \geq 0 && \text{(12e)}\\
& x_3 \in [0, 20] && \text{(12f)}\\
& x_2, x_4 \text{ free.} && \text{(12g)}
\end{aligned}$$

5.2 Writing down the dual: an algorithmic approach

Sometimes, we deal with more complex LPs that can involve maximisation, minimisation, non-negative, non-positive and unrestricted variables, $\leq$, $\geq$ and $=$-constraints, groups of variables with different names and groups of constraints with complex quantifiers. In this case, even the procedure described in Section 5.1 can be cumbersome. We can then use a more algorithmic approach to derive the dual of any LP. This approach is described in the "How to Take the Dual of a Linear Program" lecture notes by Sébastien Lahaie. The following section will use it to prove an interesting result about the max-flow and the min-cut problems.

6 Max-flow and min-cut duality

Consider the continuous relaxation of the max-flow problem:

$$\begin{aligned}
\max \quad & \sum_{j \in \delta^+(s)} x_{sj} && && \text{(13a)}\\
\text{subject to} \quad & \sum_{j \in \delta^-(i)} x_{ji} = \sum_{j \in \delta^+(i)} x_{ij} && \forall i \in V \setminus \{s, t\} && \text{(13b)}\\
& x_{ij} \leq c_{ij} && \forall (i, j) \in A && \text{(13c)}\\
& x_{ij} \geq 0 && \forall (i, j) \in A. && \text{(13d)}
\end{aligned}$$

Let us take its dual following the procedure in the "How to Take the Dual of a Linear Program" notes.

Step 1. We rewrite (13a)–(13d) in minimisation form:

$$\min \quad \sum_{j \in \delta^+(s)} -x_{sj} \tag{14a}$$

$$\text{subject to} \quad \sum_{j \in \delta^-(i)} x_{ji} = \sum_{j \in \delta^+(i)} x_{ij} \qquad \forall i \in V \setminus \{s, t\} \tag{14b}$$

$$x_{ij} \leq c_{ij} \qquad \forall (i, j) \in A \tag{14c}$$

$$x_{ij} \geq 0 \qquad \forall (i, j) \in A. \tag{14d}$$

Step 2. We rewrite the constraints so that the right-hand sides are zero.

$$\min \quad \sum_{j \in \delta^+(s)} -x_{sj} \tag{15a}$$

$$\text{subject to} \quad \sum_{j \in \delta^-(i)} x_{ji} - \sum_{j \in \delta^+(i)} x_{ij} = 0 \qquad \forall i \in V \setminus \{s, t\} \tag{15b}$$

$$x_{ij} - c_{ij} \leq 0 \qquad \forall (i, j) \in A \tag{15c}$$

$$x_{ij} \geq 0 \qquad \forall (i, j) \in A. \tag{15d}$$

Step 3. We introduce unrestricted dual variables $z_i \in \mathbb{R}$ for $i \in V \setminus \{s, t\}$ associated with (15b) and non-negative dual variables $y_{ij} \geq 0$ for $(i, j) \in A$ associated with (15c).

Step 4. We rewrite the LP as a "max-min" problem:

$$\max_{y,z} \min_{x} \quad \sum_{j \in \delta^+(s)} -x_{sj} \tag{16a}$$

$$+ \sum_{i \in V \setminus \{s,t\}} z_i \left(\sum_{j \in \delta^-(i)} x_{ji} - \sum_{j \in \delta^+(i)} x_{ij} \right) \tag{16b}$$

$$+ \sum_{(i,j) \in A} y_{ij} \big(x_{ij} - c_{ij} \big). \tag{16c}$$

Step 5. We group the terms multiplying each primal variable x_{ij}. To this end, it is convenient to consider variables of type x_{sj}, x_{it}, and x_{ij} (with $i \neq s$ and $j \neq t$) separately. The reason is that in (16b), arcs of type (s, j) can only appear in the first sum and arcs of type (i, t) can only appear in the third sum, while other arcs appear in both sums: once in the first sum as outgoing arcs, and once in the second sum as incoming arcs. Furthermore, arcs of type (s, j) also appear in (16a) and all arcs appear in (16c).

$$\max_{y,z} \max_{x} \quad \sum_{j \in \delta^+(s)} \big(\underbrace{-1}_{(16a)} + \underbrace{z_j}_{(16b)\ 1^{\text{st}}\ \text{sum}} + \underbrace{y_{sj}}_{(16c)} \big) x_{sj} + \tag{17a}$$

$$\sum_{\substack{(i,j) \in A \\ i \neq s,\ j \neq t}} \big(\underbrace{z_j}_{(16b)\ 1^{\text{st}}\ \text{sum}} - \underbrace{z_i}_{(16b)\ 2^{\text{nd}}\ \text{sum}} + \underbrace{y_{ij}}_{(16c)} \big) x_{ij} + \tag{17b}$$

$$\sum_{i \in \delta^-(t)} \big(\underbrace{-z_i}_{(16b)\ 2^{\text{nd}}\ \text{sum}} + \underbrace{y_{it}}_{(16c)} \big) x_{it} + \tag{17c}$$

$$\sum_{(i,j) \in A} \underbrace{-c_{ij} y_{ij}}_{(16c)}. \tag{17d}$$

Step 6. We write the dual in maximisation form.

$$\max \quad \sum_{(i,j)\in A} -c_{ij}y_{ij} \tag{18a}$$

$$\text{subject to} \quad -1 + z_j + y_{sj} \qquad \forall j \in \delta^+(s) \tag{18b}$$

$$z_j - z_i + y_{ij} \geq 0 \qquad \forall (i,j) \in A,\ i \neq s,\ j \neq t \tag{18c}$$

$$-z_i + y_{it} \geq 0 \qquad \forall i \in \delta^-(t) \tag{18d}$$

$$z_i \in \mathbb{R} \qquad \forall i \in V \setminus \{s,t\} \tag{18e}$$

$$y_{ij} \geq 0 \qquad \forall (i,j) \in A. \tag{18f}$$

Step 7. We put the dual in minimisation form and rearrange the constraints.

$$\min \quad \sum_{(i,j)\in A} c_{ij}y_{ij} \tag{19a}$$

$$\text{subject to} \quad y_{ij} \geq z_i - z_j \qquad \forall (i,j) \in A,\ i \neq s,\ j \neq t \tag{19b}$$

$$y_{sj} \geq 1 - z_j \qquad \forall j \in \delta^+(s) \tag{19c}$$

$$y_{it} \geq z_i \qquad \forall i \in \delta^-(t) \tag{19d}$$

$$z_i \in \mathbb{R} \qquad \forall i \in V \setminus \{s,t\} \tag{19e}$$

$$y_{ij} \geq 0 \qquad \forall (i,j) \in A. \tag{19f}$$

The LP (19a)–(19f) is *almost* the continuous relaxation of the min-cut problem formulation (5a)–(5f). The only difference is that (19e) defines the z_i's as free rather than non-negative variables. Then, if we can show that relaxing $z_i \geq 0$ to $z_i \in \mathbb{R}$ does not change the optimal solution of the continuous relaxation of the min-cut problem, we will have proven that the continuous relaxations of max-flow and min-cut form a primal-dual pair.

Indeed, we will prove a stronger statement: in any optimal solution of (19a)–(19f), the z_i's will take values in $[0,1]$. Therefore, we can replace constraints $z_i \in \mathbb{R}$ with $z_i \geq 0$.

First, we simplify the notation in the following way. Recall that variables z_i are defined for $i \in V \setminus \{s,t\}$. Therefore, symbols z_s and z_t are currently undefined. Let us, then, set $z_s = 1$ and $z_t = 0$. Remark that z_s and z_t are not variables; they are constants that, with a slight abuse of notation, use the same letter z as the variables. Using this notation, however, allows us to simplify the min-cut constraints because we can write (5b)–(5d) and, analogously, (19b)–(19d), using the single constraint

$$y_{ij} \geq z_i - z_j \quad \forall (i,j) \in A. \tag{20}$$

When (i,j) is of the form (s,j), z_i becomes $z_s = 1$ and we recover constraint (5c) (or (19c)). When (i,j) is of the form (i,t), z_j becomes $z_t = 0$ and we recover constraint (5d) (or (19d)).

We are ready to prove a useful property of the optimal solutions of (19a)–(19f).

Property 6. Let $(y,z) \in \mathbb{R}_0^{|A|,+} \times \mathbb{R}^{|V|}$ be a solution of (19a)–(19f) in which vector z has been extended to include $z_s = 1$ and $z_t = 0$. If (y,z) is optimal, then $y_{ij} = \max\{z_i - z_j, 0\}$. In other words, for each arc $(i,j) \in A$, either $y_{ij} = 0$ or (20) is satisfied with equality.

Proof. Given a solution (y,z), suppose that $y_{ij} > z_i - z_j$ and $y_{ij} > 0$ for an arc $(i,j) \in A$. We must show that such a solution cannot be optimal. Indeed, consider solution (y^*, z^*) equal to (y,j), except that component y^*_{ij} is defined as

$$y^*_{ij} = \max\{z^*_i - z^*_j, 0\}.$$

By hypothesis, $y^*_{ij} < y_{ij}$ and, therefore, the objective value of (y^*, z^*) is strictly lower than the objective value of (y, z). Because (y^*, z^*) satisfies constraint (20) by construction, we can conclude that it is a feasible solution with a strictly better objective value than (y, z) and, thus, (y, z) cannot be optimal. □

We are now able to show that, even if we let $z_i \in \mathbb{R}$ in (19a)–(19f), optimal solutions will have all $z_i \in [0, 1]$.

Property 7. Any optimal solution (y, z) of (19a)–(19f) satisfies the following property:

$$0 \leq z_i \leq 1 \quad \forall i \in V. \tag{21}$$

Proof. Suppose that a solution (y, z) violates (21) for at least one $i \in V \setminus \{s, t\}$. We build a new solution (y^*, z^*) that satisfies (21) by construction, and we show that (y^*, z^*)'s objective value is strictly better than (y, z)'s, i.e., (y, z) cannot be optimal. We build (y^*, z^*) by clipping the value of each z_i in the interval $[0, 1]$:

$$z^*_i = \min\big\{\max\{z_i, 0\}, 1\big\}.$$

Then, we assign the following values to the y^* variables:

$$y^*_{ij} = \max\{z^*_i - z^*_j, 0\}.$$

We show that the objective value of (y^*, z^*) is strictly better than the objective value of (y, z). One of the three conditions must hold for each arc $(i, j) \in A$.

1. $\boldsymbol{z_i \leq z_j}$. In this case, by definition, $z^*_i \leq z^*_j$ and, therefore, $y^*_{ij} = 0$. At the same time, $y_{ij} = 0$ because of Property 6. Therefore, arc (i, j) does not contribute anything to the objective value neither in (y, z) nor in (y^*, z^*).
2. $\boldsymbol{z_i > z_j}$ **and** $\boldsymbol{z_i, z_j \in [0, 1]}$. In this case, by definition, $z^*_i = z_i$ and $z^*_j = z_j$. Therefore, $y^*_{ij} = z^*_i - z^*_j = z_i - z_j = y_{ij}$ (the last equality is due to Property 6). It follows that the contribution of arc (i, j) to the objective value is the same in (y, z) and (y^*, z^*).
3. $\boldsymbol{z_i > z_j}$ **and** $\boldsymbol{\big(z_i \notin [0, 1]}$ **or** $\boldsymbol{z_j \notin [0, 1]\big)}$. In this case, we are clipping at least one among z_i and z_j and, therefore, strictly shortening the distance $z_i - z_j$. In other words, $z^*_i - z^*_j < z_i - z_j$ and, thus, $y^*_{ij} < y_{ij}$. Therefore, arc (i, j) contributes strictly less to the objective value of (y^*, z^*) than to the objective value of (y, z).

Since (y, z) violates (21), case 3. must hold for at least one arc and, thus, $c^\top y > c^\top y^*$, i.e., (y, z) is not optimal. □

Property 7 allows us to replace $z_i \in \mathbb{R}$ with $z_i \geq 0$ in (19a)–(19f). Indeed, we could even impose the stronger condition $z_i \in [0, 1]$ and, since Property 6 ensures that $y_{ij} = \min\{z_i - z_j, 0\}$ in all optimal solutions, we could also impose $y_{ij} \in [0, 1]$.

Consequently, the continuous relaxations of max-flow and min-cut form a primal-dual pair. Moreover, we have shown in Section 3 that the constraint matrix of the max-flow problem is TUM and, therefore, any solution of the continuous relaxation of the max-flow problem is integer. From the definition of dual, the constraint matrix of the min-cut problem is also TUM because it is the transpose of the constraint matrix of the max-flow problem. Therefore, any solution of the continuous relaxation of the min-cut problem will also be integer.

Finally, by strong duality, the optimal objective values of the max-flow and the min-cut problems coincide. The maximum flow from source to sink equals the cost of a minimum source-sink cut.