$$\sum_{j=1}^{n}\left(\sum_{i=1}^{m} a_{ij}y_i\right)x_j = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} a_{ij}x_j\right)y_i.$$

□

By the LP-duality theorem, $\boldsymbol{x}$ and $\boldsymbol{y}$ are both optimal solutions iff (12.3) holds with equality. Clearly, this happens iff both (12.4) and (12.5) hold with equality. Hence, we get the following result about the structure of optimal solutions:

Theorem 12.3 (Complementary slackness conditions) *Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be primal and dual feasible solutions, respectively. Then, $\boldsymbol{x}$ and $\boldsymbol{y}$ are both optimal iff all of the following conditions are satisfied:*

Primal complementary slackness conditions
For each $1 \le j \le n$: either $x_j = 0$ or $\sum_{i=1}^{m} a_{ij}y_i = c_j$; and
Dual complementary slackness conditions
For each $1 \le i \le m$: either $y_i = 0$ or $\sum_{j=1}^{n} a_{ij}x_j = b_i$.

The complementary slackness conditions play a vital role in the design of efficient algorithms, both exact and approximation; see Chapter 15 for details. (For a better appreciation of their importance, we recommend that the reader study algorithms for the weighted matching problem, see Section 12.5.)

12.2 Min–max relations and LP-duality

In order to appreciate the role of LP-duality theory in approximation algorithms, it is important to first understand its role in exact algorithms. To do so, we will review some of these ideas in the context of the max-flow min-cut theorem. In particular, we will show how this and other min–max relations follow from the LP-duality theorem. Some of the ideas on cuts and flows developed here will also be used in the study of multicommodity flow in Chapters 18, 20, and 21.

The problem of computing a maximum flow in a network is: given a directed[1] graph, $G = (V, E)$ with two distinguished nodes, *source* s and *sink* t, and positive arc capacities, $c : E \to \mathbf{R}^+$, find the maximum amount of flow that can be sent from s to t subject to

1. *capacity constraint:* for each arc e, the flow sent through e is bounded by its capacity, and

[1] The maximum flow problem in undirected graphs reduces to that in directed graphs: replace each edge (u, v) by two directed edges, $(u \to v)$ and $(v \to u)$, each of the same capacity as (u, v).

2. *flow conservation:* at each node v, other than s and t, the total flow into v should equal the total flow out of v.

An *s–t cut* is defined by a partition of the nodes into two sets X and $\overline{X}$ so that $s \in X$ and $t \in \overline{X}$, and consists of the set of arcs going from X to $\overline{X}$. The *capacity* of this cut, $c(X, \overline{X})$, is defined to be the sum of capacities of these arcs. Because of the capacity constraints on flow, the capacity of any s–t cut is an upper bound on any feasible flow. Thus, if the capacity of an s–t cut, say $(X, \overline{X})$, equals the value of a feasible flow, then $(X, \overline{X})$ must be a minimum s–t cut and the flow must be a maximum flow in G. The max-flow min-cut theorem proves that it is always possible to find a flow and an s–t cut so that equality holds.

Let us formulate the maximum flow problem as a linear program. First, introduce a fictitious arc of infinite capacity from t to s, thus converting the flow to a circulation; the objective now is to maximize the flow on this arc, denoted by f_{ts}. The advantage of making this modification is that we can now require flow conservation at s and t as well. If f_{ij} denotes the amount of flow sent through arc $(i, j) \in E$, we can formulate the maximum flow problem as follows:

$$
\begin{array}{lll}
\text{maximize} & f_{ts} & \\
\text{subject to} & f_{ij} \le c_{ij}, & (i,j) \in E \\
& \displaystyle\sum_{j:\ (j,i)\in E} f_{ji} - \sum_{j:\ (i,j)\in E} f_{ij} \le 0, & i \in V \\
& f_{ij} \ge 0, & (i,j) \in E
\end{array}
$$

The second set of inequalities say that for each node i, the total flow into i is at most the total flow out of i. Notice that if this inequality holds at each node, then in fact it must be satisfied with equality at each node, thereby implying flow conservation at each node (this is so because a deficit in flow balance at one node implies a surplus at some other node). With this trick, we get a linear program in standard form.

To obtain the dual program we introduce variables d_{ij} and p_i corresponding to the two types of inequalities in the primal. We will view these variables as distance labels on arcs and potentials on nodes, respectively. The dual program is:

$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{(i,j)\in E} c_{ij} d_{ij} & \\
\text{subject to} & d_{ij} - p_i + p_j \ge 0, & (i,j) \in E \\
& p_s - p_t \ge 1 & \\
& d_{ij} \ge 0, & (i,j) \in E
\end{array}
\tag{12.6}
$$

$$p_i \geq 0, \qquad i \in V \tag{12.7}$$

For developing an intuitive understanding of the dual program, it will be best to first transform it into an integer program that seeks 0/1 solutions to the variables:

$$\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{(i,j)\in E} c_{ij} d_{ij} & \\
\text{subject to} & d_{ij} - p_i + p_j \geq 0, & (i,j) \in E \\
& p_s - p_t \geq 1 & \\
& d_{ij} \in \{0,1\}, & (i,j) \in E \\
& p_i \in \{0,1\}, & i \in V
\end{array}$$

Let $(\boldsymbol{d}^*, \boldsymbol{p}^*)$ be an optimal solution to this integer program. The only way to satisfy the inequality $p_s^* - p_t^* \geq 1$ with a 0/1 substitution is to set $p_s^* = 1$ and $p_t^* = 0$. This solution naturally defines an s–t cut $(X, \overline{X})$, where X is the set of potential 1 nodes, and $\overline{X}$ the set of potential 0 nodes. Consider an arc (i, j) with $i \in X$ and $j \in \overline{X}$. Since $p_i^* = 1$ and $p_j^* = 0$, by the first constraint, $d_{ij}^* \geq 1$. But since we have a 0/1 solution, $d_{ij}^* = 1$. The distance label for each of the remaining arcs can be set to either 0 or 1 without violating the first constraint; however, in order to minimize the objective function value, it must be set to 0. The objective function value must thus be equal to the capacity of the cut $(X, \overline{X})$, and $(X, \overline{X})$ must be a minimum s–t cut.

Thus, the previous integer program is a formulation of the minimum s–t cut problem! What about the dual program? The dual program can be viewed as a relaxation of the integer program where the integrality constraint on the variables is dropped. This leads to the constraints $1 \geq d_{ij} \geq 0$ for $(i, j) \in E$ and $1 \geq p_i \geq 0$ for $i \in V$. Next, we notice that the upper bound constraints on the variables are redundant; their omission cannot give a better solution. Dropping these constraints gives the dual program in the form given above. We will say that this program is the *LP-relaxation* of the integer program.

Consider an s–t cut C. Set C has the property that any path from s to t in G contains at least one edge of C. Using this observation, we can interpret any feasible solution to the dual program as a *fractional s–t cut*: the distance labels it assigns to arcs satisfy the property that on any path from s to t the distance labels add up to at least 1. To see this, consider an s–t path $(s = v_0, v_1, \ldots, v_k = t)$. Now, the sum of the potential differences on the endpoints of arcs on this path is

$$\sum_{i=0}^{k-1} (p_i - p_{i+1}) = p_s - p_t.$$

By the first constraint, the sum of the distance labels on the arcs must add up to at least $p_s - p_t$, which is ≥ 1. Let us define the *capacity* of this fractional s–t cut to be the dual objective function value achieved by it.

In principle, the best fractional s–t cut could have lower capacity than the best integral cut. Surprisingly enough, this does not happen. Consider the polyhedron defining the set of feasible solutions to the dual program. Let us call a feasible solution an *extreme point solution* if it is a vertex of this polyhedron, i.e., it cannot be expressed as a convex combination of two feasible solutions. From linear programming theory we know that for any objective function, i.e., assignment of capacities to the arcs of G, there is an extreme point solution that is optimal (for this discussion let us assume that for the given objective function, an optimal solution exists). Now, it can be proven that each extreme point solution of the polyhedron is integral, with each coordinate being 0 or 1 (see Exercise 12.6). Thus, the dual program always has an integral optimal solution.

By the LP-duality theorem maximum flow in G must equal capacity of a minimum fractional s–t cut. But since the latter equals the capacity of a minimum s–t cut, we get the max-flow min-cut theorem.

The max-flow min-cut theorem is therefore a special case of the LP-duality theorem; it holds because the dual polyhedron has integral vertices. In fact, most min–max relations in combinatorial optimization hold for a similar reason.

Finally, let us illustrate the usefulness of complementary slackness conditions by utilizing them to derive additional properties of optimal solutions to the flow and cut programs. Let $\boldsymbol{f}^*$ be an optimum solution to the primal LP (i.e., a maximum s–t flow). Also, let $(\boldsymbol{d}^*, \boldsymbol{p}^*)$ be an integral optimum solution to the dual LP, and let $(X, \overline{X})$ be the cut defined by $(\boldsymbol{d}^*, \boldsymbol{p}^*)$. Consider an arc (i, j) such that $i \in X$ and $j \in \overline{X}$. We have proven above that $d^*_{ij} = 1$. Since $d^*_{ij} \neq 0$, by the dual complementary slackness condition, $f^*_{ij} = c_{ij}$. Next, consider an arc (k, l) such that $k \in \overline{X}$ and $l \in X$. Since $p^*_k - p^*_l = -1$, and $d^*_{kl} \in \{0, 1\}$, the constraint $d^*_{kl} - p^*_k + p^*_l \geq 0$ must be satisfied as a strict inequality. By the primal complementary slackness condition, $f^*_{kl} = 0$. Thus, we have proven that arcs going from X to $\overline{X}$ are saturated by $\boldsymbol{f}^*$ and the reverse arcs carry no flow. (Observe that it was not essential to invoke complementary slackness conditions to prove these facts; they also follow from the fact that flow across cut $(X, \overline{X})$ equals its capacity.)

12.3 Two fundamental algorithm design techniques

We can now explain why linear programming is so useful in approximation algorithms. Many combinatorial optimization problems can be stated as integer programs. Once this is done, the linear relaxation of this program provides a natural way of lower bounding the cost of the optimal solution. As stated in Chapter 1, this is typically a key step in the design of an approximation