12 The Maximum Flow Problem

Consider a flow network (V, E) with a single source 1, a single sink n, and finite capacities $\overline{m}_{ij} = C_{ij}$ for all $(i, j) \in E$. We will also assume for convenience that $\underline{m}_{ij} = 0$ for all $(i, j) \in E$. The *maximum flow problem* then asks for the maximum amount of flow that can be sent from vertex 1 to vertex n, i.e., the goal is to

$$\begin{aligned} &\text{maximize} \quad \delta \\ &\text{subject to} \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} \delta & \text{if } i = 1 \\ -\delta & \text{if } i = n \\ 0 & \text{otherwise} \end{cases} \\ &\qquad\qquad 0 \leqslant x_{ij} \leqslant C_{ij} \quad \text{for all } (i, j) \in E. \end{aligned} \tag{12.1}$$

This problem is in fact a special case of the minimum-cost flow problem. To see this, set $c_{ij} = 0$ for all $(i, j) \in E$, and add an edge $(n, 1)$ with infinite capacity and cost $c_{n1} = -1$. Since the new edge $(n, 1)$ has infinite capacity, any feasible flow of the original network is also feasible for the new network. Cost is clearly minimized by maximizing the flow across the edge $(n, 1)$, which by the flow conservation constraints for vertices 1 and n maximizes flow through the original network.

12.1 The Max-Flow Min-Cut Theorem

Consider a flow network $G = (V, E)$ with capacities C_{ij} for $(i, j) \in E$. A *cut* of G is a partition of V into two sets, and the capacity of a cut is defined as the sum of capacities of all edges across the partition. Formally, for $S \subseteq V$, the capacity of the cut $(S, V \setminus S)$ is given by

$$C(S) = \sum_{(i,j)\in S\times(V\setminus S)} C_{ij}.$$

Assume that x is a feasible flow vector that sends δ units of flow from vertex 1 to vertex n. It is easy to see that δ is bounded from above by the capacity of any cut S with $1 \in S$ and $n \in V \setminus S$. Indeed, for $X, Y \subseteq V$, let

$$f_x(X, Y) = \sum_{(i,j)\in E\cap(X\times Y)} x_{ij}.$$

Then, for any $S \subseteq V$ with $1 \in S$ and $n \in V \setminus S$,

$$\begin{aligned}\delta &= \sum_{i \in S} \left(\sum_{j:(i,j) \in E} x_{ij} - \sum_{j:(j,i) \in E} x_{ji} \right) \\ &= f_x(S, V) - f_x(V, S) \\ &= f_x(S, S) + f_x(S, V \setminus S) - f_x(V \setminus S, S) - f_x(S, S) \\ &= f_x(S, V \setminus S) - f_x(V \setminus S, S) \\ &\leqslant f_x(S, V \setminus S) \leqslant C(S).\end{aligned} \tag{12.2}$$

The following result states that this upper bound in fact tight, i.e., that there exists a flow of size equal to the minimum capacity of a cut that separates vertex 1 from vertex n.

THEOREM 12.1 (Max-flow min-cut theorem). *Let δ be the optimal solution of* (12.1) *for a network* (V, E) *with capacities* C_{ij} *for all* $(i, j) \in E$. *Then,*

$$\delta = \min\{\, C(S) : S \subseteq V, 1 \in S, n \in V \setminus S \,\}.$$

Proof. It remains to be shown that there exists a cut that separates vertex 1 from vertex n and has capacity equal to δ. Consider a feasible flow vector x. A path $v_0, v_1, \ldots, v_k$ is called an *augmenting path* for x if $x_{v_{i-1}v_i} < C_{v_{i-1}v_i}$ or $x_{v_i v_{i-1}} > 0$ for every $i = 1, \ldots, k$. If there exists an augmenting path from vertex 1 to vertex n, then we can push flow along the path, by increasing the flow on every forward edge and decreasing the flow on every backward edge along the path by the same amount, such that all constraints remain satisfied and the amount of flow from 1 to n increases.

Now assume that x is optimal, and let

$$S = \{1\} \cup \{\, i \in V : \text{there exists an augmenting path for } x \text{ from 1 to } i \,\}.$$

By optimality of x, $n \in V \setminus S$. Moreover,

$$\delta = f_x(S, V \setminus S) - f_x(V \setminus S, S) = f_x(S, V \setminus S) = C(S).$$

The first equality holds by (12.2). The second equality holds because $x_{ij} = 0$ for every $(i, j) \in E \cap ((V \setminus S) \times S)$. The third equality holds because $x_{ij} = C_{ij}$ for every $(i, j) \in E \cap (S \times (V \setminus S))$. □

12.2 The Ford-Fulkerson Algorithm

The *Ford-Fulkerson algorithm* attempts to find a maximum flow by repeatedly pushing flow along an augmenting path, until such a path can no longer be found:

1. Start with a feasible flow vector x.
2. If there is no augmenting path for x from 1 to n, stop.

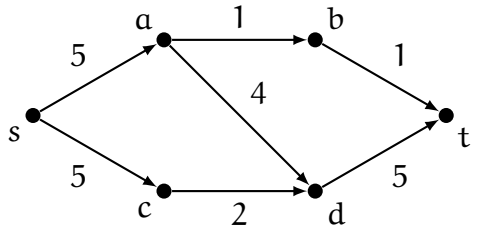

Figure 12.1: An instance of the maximum flow problem

3. Otherwise pick some augmenting path from 1 to n, and push a maximum amount of flow along this path without violating any constraints. Then go to Step 2.

Consider for example the flow network in Figure 12.1. Pushing one unit of flow along the path s, a, b, t, four units along the path s, a, d, t, and one more unit along the path s, c, d, t yields a maximum flow, and the fact that this flow is optimal is witnessed by the cut $(\{s, a, b, c, d\}, \{t\})$, which has capacity 6.

If all capacities are integral and if we start from an integral flow vector, e.g., the flow vector x such that $x_{ij} = 0$ for all $(i, j) \in E$, then the Ford-Fulkerson algorithm maintains integrality and increases the overall amount of flow by at least one unit in each iteration. The algorithm is therefore guaranteed to find a maximum flow after a finite number of iterations. Clearly, the latter also holds when all capacities are rational.

12.3 The Bipartite Matching Problem

A *matching* of a graph (V, E) is a set of edges that do not share any vertices, i.e., a set $M \subseteq E$ such for all $(s, t), (u, v) \in M$, $s \neq u$ and $s \neq v$. Matching M is called perfect if it covers every vertex, i.e., if $|M| = |V|/2$. A graph is k-regular if every vertex has degree k. Using flows it is easy to show that every k-regular bipartite graph, for $k \geqslant 1$, has a perfect matching. For this, consider a k-regular bipartite graph $(L \uplus R, E)$, orient all edges from L to R, and add two new vertices s and t and new edges (s, i) and (j, t) for every $i \in L$ and $j \in R$. Finally set the capacity of every new edge to 1, and that of every original edge to infinity. We can now send $|L|$ units of flow from s to t by setting the flow to 1 for every new edge and to $1/k$ for every original edge. The Ford-Fulkerson algorithm is therefore guaranteed to find an integral solution with at least the same value, and it is easy to see that such a solution corresponds to a perfect matching.

This result is a special case of a well-known characterization of the bipartite graphs that have a perfect matching. It should not come as a surprise that this characterization can be obtained from the max-flow min-cut theorem as well.

THEOREM 12.2 (Hall's Theorem). *A bipartite graph* $G = (L \uplus R, E)$ *with* $|L| = |R|$ *has a perfect matching if and only if* $|N(X)| \geqslant |X|$ *for every* $X \subseteq L$*, where* $N(X) = \{j \in R : i \in X, (i, j) \in E\}$.