Chapter 5

Network Flows

A wide variety of engineering and management problems involve optimization of network flows – that is, how objects move through a network. Examples include coordination of trucks in a transportation system, routing of packets in a communication network, and sequencing of legs for air travel. Such problems often involve few indivisible objects, and this leads to a finite set of feasible solutions. For example, consider the problem of finding a minimal cost sequence of legs for air travel from Nuku'alofa to Reykjavik. Though there are many routes that will get a traveller from one place to the other, the number is finite. This may appear as a striking difference that distinguishes network flows problems from linear programs – the latter always involves a polyhedral set of feasible solutions. Surprisingly, as we will see in this chapter, network flows problems can often be formulated and solved as linear programs.

5.1 Networks

A network is characterized by a collection of nodes and directed edges, called a *directed graph*. Each edge points from one node to another. Figure 5.1 offers a visual representation of a directed graph with nodes labelled 1 through 8. We will denote an edge pointing from a node i to a node j by (i, j). In this notation, the graph of Figure 5.1 can be characterized in terms of a set of nodes $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and a set of edges $E = \{(1, 2), (1, 3), (1, 6), (2, 5), (3, 4), (4, 6), (5, 8), (6, 5), (6, 7), (7, 8)\}$.

Graphs can be used to model many real networked systems. For example, in modelling air travel, each node might represent an airport, and each edge a route taken by some flight. Note that, to solve a specific problem, one often requires more information than the topology captured by a graph. For

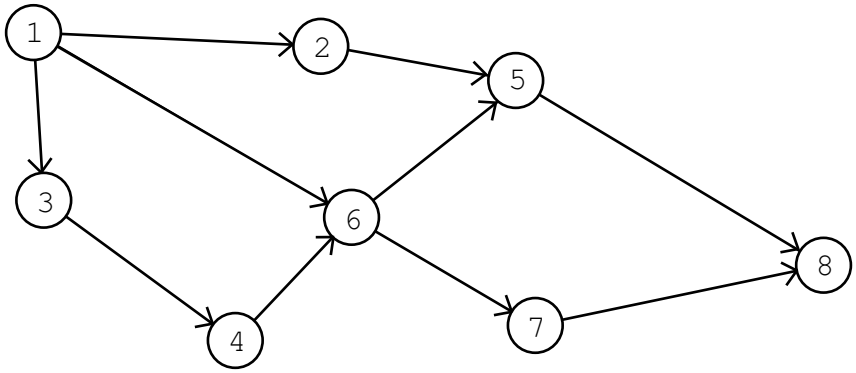

Figure 5.1: A directed graph.

example, to minimize cost of air travel, one would need to know costs of tickets for various routes.

5.2 Min-Cost-Flow Problems

Consider a directed graph with a set V of nodes and a set E of edges. In a min-cost-flow problem, each edge $(i, j) \in E$ is associated with a cost c_{ij} and a capacity constraint u_{ij}. There is one decision variable f_{ij} per edge $(i, j) \in E$. Each f_{ij} is represents a flow of objects from i to j. The cost of a flow f_{ij} is $c_{ij}f_{ij}$. Each node $j \in V \setminus \{s, d\}$ satisfies a flow constraint:

$$\sum_{\{k|(j,k)\in E\}} f_{jk} - \sum_{\{i|(i,j)\in E\}} f_{ij} = b_j,$$

where b_j denotes an amount of flow generated by node j. Note that if b_j is negative, the node consumes flow.

The min-cost-flow problem is to find flows that minimize total cost subject to capacity and flow conservation constraints. It can be written as a linear program:

$$\begin{array}{lll} \text{minimize} & \sum_{(i,j)\in E} c_{ij}f_{ij} & \\ \text{subject to} & \sum_{\{k|(j,k)\in E\}} f_{jk} - \sum_{\{i|(i,j)\in E\}} f_{ij} = b_j, & \forall j \in V \\ & 0 \le f_{ij} \le u_{ij}, & \forall (i,j) \in E. \end{array} \tag{5.1}$$

An important property of the min-cost-flow problem is that basic feasible solutions are integer-valued if capacity constraints and quantitied of flow produced at each node are integer-valued, as captured by the following theorem.

Theorem 5.2.1. *If for all $i \in b$, b_i is an integer, and for all $(i,j) \in E$, u_{ij} is an integer, then for any basic feasible solution of the linear program (5.1), each flow f_{ij} is an integer.*

Proof. Our proof builds on three facts:

1. A basic feasible solution can not be a convex combination of two other feasible solutions.

2. If f is a feasible solution and there is an incoming or outgoing edge at node i for which the flow is fractional then there must be at least one additional incoming or outgoing edge at node i for which the flow is fractional.

3. If there is fractional flow along an edge, the capacity constraints for that edge cannot be binding.

The first fact was established in Chapter 3. The second follows from the flow conservation constraint. The third is a consequence of integer-valued capacity constraints.

Assume for contradiction that there is a basic feasible solution f such that for some edge (i,j), the flow f_{ij} takes on a fractional value. It follows from Fact 2 above that there must be a cycle of edges in the graph, each of which has fractional flow. Then, by Fact 3, a small flow of ϵ can be added to or subtracted from the cycle, while retaining feasibility either way. Clearly, f is the convex combination of these two alternatives. Therefore, by Fact 1, f is not a basic feasible solution. □

Since basic feasible solutions are integer-valued, when there is an optimal solution, there will be one that is integer-valued. This enables use of linear programming algorithms to solve min-cost-flow problems even when integer-valued solutions are required. We discuss some examples in the following subsections.

5.2.1 Shortest Path Problems

Consider a graph with nodes V and a set of edges E. Think of each node as a city and each edge as a highway that can be used to send a truck from one city to another. Suppose we are given the travel time c_{ij} associated with each highway, and we want to get the truck from node o (the origin) to node d (the destination) in minimum time. Suppose that for each $(i,j) \in E$, we take f_{ij} to be a binary variable to which we would assign a value of 1 if edge (i,j) is to be part of the route and 0 otherwise. Then, the route with shortest travel time can be found by solving the a min-cost-flow problem with $b_o = 1$,

$b_d = -1$, $b_i = 0$ for $i \notin \{o, d\}$, $u_{ij} = 1$ for each $(i, j) \in E$, and an additional constraint that $f_{ij} \in \{0, 1\}$ for each $(i, j) \in E$.

The additional set of constraints that require flow variables to be binary are introduced because it is not possible to send a fraction of the truck along a highway. With the additional constraints, the problem is not a linear program. Fortunately, we can drop these additional constraints and by doing so obtain a min-cost-flow problem – a linear program for which basic feasible solutions are integer-valued. Because basic feasible solutions are integer-valued, if there exists an optimal solution, which is the case if there is a path from node o to node d, then there will be an integer-valued optimal solution. This means a binary-valued solution since each flow is constrained to be between 0 and 1.

5.2.2 Transportation Problems

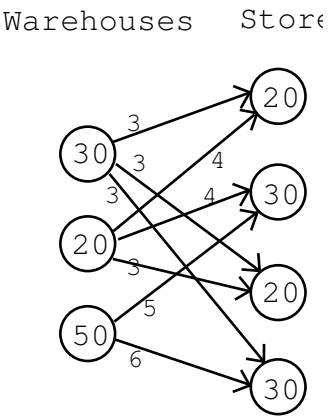

Figure 5.2: A transportation problem.

Consider a situation illustrated in Figure 5.2 where we have inventory at warehouses to be transported to retail stores. Each circle in the left column represents a warehouse, while each circle on the right column represents a store. Each warehouse holds a certain supply of inventory, given by the number written in its corresponding circle. Each store demands a certain amount of inventory, given by the number written in *its* circle. Each line represents a route through which inventory can be transported from a warehouse to a store and is marked with a number that indicates per unit inventory transportation cost. The units are indivisible. The goal is to satisfy demands while minimizing transportation costs.

The problem illustrated in Figure 5.2 is an instance of the transportation problem. More generally, the transportation problem involves:

- n suppliers with inventory amounts $S_1, \ldots, S_m$,
- m demand sites with inventory demands $D_1, \ldots, D_n$,
- a set T of routes from suppliers to demand sites,
- costs c_{ij}, for each $(i, j) \in T$ for transporting one unit from supplier i to demand site j.

Transportation problems can be formulated as min-cost-flow problems with one node per supplier and one node per demand site. Edges correspond to elements of T. There are not capacity constraints (i.e., each u_{ij} is set to ∞). The flow quantities produced by suppliers are given by $S_1, \ldots, S_m$ and the quantities required at demand sites are given by $D_1, \ldots, D_n$. Since this is a min-cost-flow problem, basic feasible solutions assign integer values to variables. Hence, this formulation addresses the requirement that units of inventory are indivisible.

5.2.3 Assignment Problems

The assignment problem involves allocating a number of jobs to workers in some optimal manner. A simple example should be sufficient to demonstrate the central ideas.

Example 5.2.1. *The coach of a swim team needs to assign swimmers to a 200-yard medley relay team to compete in a tournament. The problem facing him is that his best swimmers are good in more than one stroke, so it is not clear which swimmer to assign to which stroke. The 5 fastest swimmers and the best times (in seconds) they have achieved with each of the strokes (for 50 yards) are given below.*

Stroke	*Carl*	*Chris*	*David*	*Tony*	*Ken*
Backstroke	*37.7*	*32.9*	*33.8*	*37.0*	*35.4*
Breaststroke	*43.4*	*33.1*	*42.2*	*34.7*	*41.8*
Butterfly	*33.3*	*28.5*	*38.9*	*30.4*	*33.6*
Freestyle	*29.2*	*26.4*	*29.6*	*28.5*	*31.1*

The problem is to try to minimize the sum of the best times for the people competing in the race.

The assignment problem is a special case of the transportation problem. To convert the above example to a transportation problem think of each swimmer as a supplier with one unit of inventory and each stroke as a site

demanding one unit of inventory. There is an edge from each swimmer to each stroke in which he can participate. Assign a cost to each edge equal to the time it takes the swimmer to do the stroke. The only problem that arises is that the number of swimmers is greater than the number of strokes. One way to deal with this is to introduce a dummy stroke (the "not taking part" stroke), and treat that as an additional site with one unit of demand. The time for each swimmer assigned to the dummy stroke would be zero.

5.3 Max-Flow Problems

Consider a graph with a set of vertices V, a set of edges E, and two distinguished nodes o and d. Each edge has an associated capacity u_{ij} but no associated cost. We will assume that there is no edge from d to o. In a max-flow problem, the goal is to maximize the total flow from o to d. In our formulation, nodes do not produce or consume flow, but rather, we introduce an auxiliary edge (d, o) with no capacity limit and aim to maximize flow along this edge. By doing so, we indirectly maximize flow from o to d via the edges in E. We define an augmented set of edges $E' = E \cup \{(d, o)\}$. The max-flow problem is given by

$$\begin{array}{lll} \text{maximize} & f_{do} & \\ \text{subject to} & \sum_{\{k|(j,k)\in E'\}} f_{jk} - \sum_{\{i|(i,j)\in E'\}} f_{ij} = 0, & \forall j \in V \\ & 0 \leq f_{ij} \leq u_{ij}, & \forall (i,j) \in E. \end{array} \tag{5.2}$$

Note that this can be thought of as a special case of the min-cost-flow problem, where all edges have zero cost, except for (d, o), which is assigned a cost of $c_{do} = -1$.

5.3.1 Min-Cut Problems

An interesting related problem is the min-cut problem. A *cut* is a partition of the nodes V into two disjoint sets V_o and V_d such that $o \in V_s$, $d \in V_d$, and $V_o \cup V_d = V$. The objective in the min-cut problem is to find a cut that such that the capacity for flows from V_s to V_d is minimized. One way to represent a choice of partition involves assigning a binary variable to each node. In particular, for each node $i \in V$, let $p_i = 0$ if $i \in V_o$ and $p_i = 1$ if $i \in V_d$. Note that for nodes o and d, we always have $p_o = 0$ and $p_d = 1$. Further, for each $(i, j) \in E$, let $q_{ij} = 1$ if there is an edge directed from i to j, and let $q_{ij} = 0$, otherwise. Note that $q_{ij} = \max(p_j - p_i, 0)$. The min-cut problem

can be written as

$$\begin{array}{lll} \text{minimize} & \sum_{(i,j)\in E} u_{ij} q_{ij} & \\ \text{subject to} & q_{ij} = \max(p_j - p_i, 0), & \forall (i,j) \in E, \\ & p_d - p_s = 1, & \\ & p_i \in \{0,1\}, & \forall i \in V. \end{array} \tag{5.3}$$

The decision variables being optimized here include each p_i for $i \in V$ and each q_{ij} for $(i,j) \in E$. Since the problem involves minimizing a weighted combination of q_{ij}'s, with positive weights, and each q_{ij} is constrained to $\{0,1\}$, the constraint that $q_{ij} = \max(p_j - p_i, 0)$ can be converted to $q_{ij} \geq p_j - p_i$, without changing the optimal solution. Hence, an equivalent optimization problem is:

$$\begin{array}{lll} \text{minimize} & \sum_{(i,j)\in E} u_{ij} q_{ij} & \\ \text{subject to} & q_{ij} \geq p_j - p_i, & \forall (i,j) \in E, \\ & p_d - p_s = 1, & \\ & q_{ij} \geq 0, & \forall (i,j) \in E, \\ & p_i \in \{0,1\}, & \forall i \in V. \end{array} \tag{5.4}$$

It is easy to see that the optimal objective value for the min-cut problem is greater than or equal to the optimal objective value for the max-flow problem. This is because for any cut, flow that gets from o to d must pass from V_o to V_d. Hence, the maximal flow is limited by the capacity for flow from V_o to V_d.

The min-cut problem as stated above is not a linear program because the variables are constrained to be binary. As we will now explain, the following linear program also solves the min-cut problem:

$$\begin{array}{lll} \text{minimize} & \sum_{(i,j)\in E} u_{ij} q_{ij} & \\ \text{subject to} & q_{ij} \geq p_j - p_i, & \forall (i,j) \in E, \\ & p_d - p_s = 1, & \\ & q_{ij} \geq 0, & \forall (i,j) \in E. \end{array} \tag{5.5}$$

This linear program is the dual of the max-flow problem (5.2). We will not go through the mechanics of converting the max-flow problem to its dual here – that is straightforward to do. However, we will argue that, as a consequence, at each optimal basic feasible solution, each q_{ij} is binary valued.

At an optimal basic feasible solution, each q_{ij} can be viewed as the sensitivity of max-flow to the capacity constraint u_{ij}. At an optimal basic feasible solution to the dual, q_{ij} is equal to either the rate at which flow increases as a consequence increasing u_{ij} or the rate at which flow decreases as a consequence of decreasing u_{ij}. Because all capacities are integer-valued, if increasing u_{ij} slightly can increase flow, the rate of increase in flow must

equal the rate of increase in capacity. The situation with decreasing u_{ij} is analogous. Hence, if the sensitivity is nonzero, it is equal to one.

Note that the p_is may not be binary-valued; for example, adding a constant to every p_i maintains feasibility and does not alter the objective value. So given binary-valued q_{ij}s, how do we recover the cut? Well, at an optimal basic feasible solution, $q_{ij} = 1$ if and only if the edge (i, j) goes from V_o to V_d. Based on this, one can easily recover the cut.

5.3.2 Matching Problems

Consider making matches among n potential grooms and n potential brides. We have data on which couples will or will not be happy together, and the goal is to create as many happy couples as possible. One might represent the compatibilities in terms of a graph as depicted in Figure 5.3. The problem of finding a matching that maximizes the number of happy couples is known as the *matching problem.*

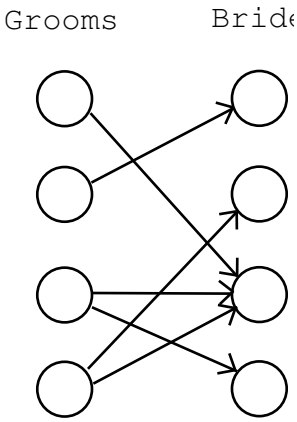

Figure 5.3: A matching problem.

Clearly, this problem can be treated as a max-flow problem by adding origin and destination nodes as shown in Figure 5.4. The capacity of each edge is one. By solving the linear program associated with this max-flow problem, we arrive at a solution that maximizes the number of happy couples. If we are lucky this will turn out to be n. But sometimes, it may not. Interestingly, there is a simple rule for determining whether or not there will be n happy couples without solving the linear program.

Theorem 5.3.1. (Hall's Marriage Lemma) *There exists a matching such that every bride and every groom is happy if and only if for any set of grooms,*

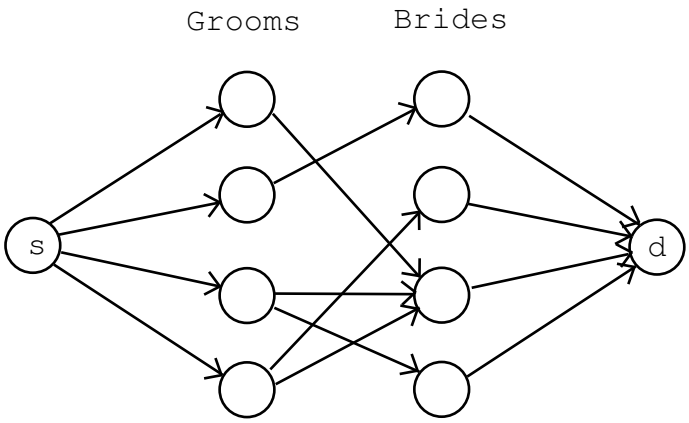

Figure 5.4: Conversion to a max-flow problem.

the set of brides that would form a happy couple with at least one of these grooms is at least as large as the set of grooms.

Proof. It is obvious that for any set A of grooms, there has to be a set of brides of at least equal size in order for there to be any hope of a perfect matching. What is less clear is that there must be a perfect matching if this condition holds for every set of grooms. This follows from the fact that there is a perfect matching if the maximal flow/minimal cut equals n (the number of grooms/brides). It is easy to see that if the conditions on grooms and brides holds, there cannot be any cut of value less than n, and the result follows. □

5.4 Exercises

Question 1 - Shortest Path and LP

You are going to visit a national park, and have never been there before. You are using a map to try and make the distance travelled as short as possible. There are 5 intermediate towns, A, B, C, D, and E, you may go through on your way to the park, and the distances between the various locations are given below.

	Miles between Adjacent Towns					
Town	A	B	C	D	E	Destination
Origin	40	60	50	-	-	-
A		10	-	70	-	-
B			20	55	40	-
C				-	50	-
D					10	60
E						80

In the table above, a dash means that there is no direct road between the two locations.

1. Draw the graph corresponding to the above problem.

2. Give a linear program which will find the shortest path from Origin to Destination.

3. Solve the problem in Excel.

4. Suppose the numbers in the table now represented the time taken (rather than the distance). What would be the minimum time taken to travel from Origin to Destination?

5. Suppose the number in the table now represented capacities for water pipes connecting the various locations (in units of gal/sec). What is the maximum amount of water that can be pumped from Origin to Destination?

Question 2 - Networks and Swimming

The coach of a swim team needs to assign swimmers to a 200-yard medley relay team to compete in a tournament. The problem facing him is that his best swimmers are good in more than one stroke, so it is not clear which swimmer to assign to which stroke. The 5 fastest swimmers and the best times (in seconds) they have achieved with each of the strokes (for 50 yards) are given below.

Stroke	Carl	Chris	David	Tony	Ken
Backstroke	37.7	32.9	33.8	37.0	35.4
Breaststroke	43.4	33.1	42.2	34.7	41.8
Butterfly	33.3	28.5	38.9	30.4	33.6
Freestyle	29.2	26.4	29.6	28.5	31.1

The problem is to try to minimize the sum of the best times for the people competing in the race.

1. Draw a network that describes this problem. Note that one person will not be assigned.

2. Formulate the problem as a linear program. Solve it on Excel.

3. Write the dual of the problem.

4. If the coach can concentrate on improving one persons time on one stroke by 0.5 seconds. On whom should the coach spend his time (note, there may be more than one answer).

Question 3 - Graduation Party Matching

Suppose you are hosting a graduation party in your dorm. n women and n men live in your dorm (including yourself) and the party will only include the residents of the dorm. Tired of the usual process of finding a partner, you as a group decide to use a centralized matching mechanism for setting up couples for the party (each couple consists of one man and one woman). Moreover you decide this is a good chance to use for the last time linear programming before graduating.

The matching mechanism is described in what follows:

- Let $i = 1, ..., n$ be an index of the men and let $j = 1, ..., n$ be an index of the women, so that each woman (and each man) is given a label between 1 and n.

- Each woman and each man declares her/his preferences as a ranking. Let $w^j \in \Re^n$ be woman j's ranking. w^j_k is the ranking of the kth man for woman j. For example, if there are 4 men, a possible ranking for woman 1 would be $w^1 = (3, 4, 2, 1)$, meaning that man 1 is ranked in the third place, man 2 is ranked in the fourth place (he is the least preferred), man 3 is ranked second and man 4 is ranked first (he is the most preferred). Similarly, let $m^i \in \Re^n$ be man's i ranking of women.

- Each woman (man) must be paired with exactly one man (woman). The centralized matching allocates the n woman-man pairs in order to minimize the sum of the products of the rankings of the allocated pairs. For example, if woman 3 is paired with man 4, then the product of rankings is $c_{3,4} = w^3_4 \cdot m^4_3$. We want to minimize the sum of these products over the allocated pairs. We call the outcome(s) of this algorithm the "Product Matching(s)" (the solution is not necessarily unique).

Answer the following:

1. Formulate a Linear Programming problem that determines the "Product Matching(s)".

2. Is the solution of the LP an integer solution? Why or why not?

3. Suppose there are 6 women and 6 men. The rankings are given in the attached excel file, *rankings.xls*. Find the "Product Matching" using Excel.

4. We say that a matching is "stable" if there is no couple that would like to deviate from the matching to become partners. For example, suppose the matching assigns couples 1-2 and 3-4, the first number being the woman and the second being the man. Suppose woman 1 ranks man 2 sixth and man 4 fifth; and man 4 ranks woman 1 fifth and woman 3 sixth. Then woman 1 and man 4 would like to deviate from the matching. They are matched to their sixth preference, but if they deviate and become partners they will be matched to their fifth. Is the Product Matching from the previous example stable? Why or why not?