7.11 Project Selection

Large (and small) companies are constantly faced with a balancing act between projects that can yield revenue, and the expenses needed for activities that can support these projects. Suppose, for example, that the telecommunications giant CluNet is assessing the pros and cons of a project to offer some new type of high-speed access service to residential customers. Marketing research shows that the service will yield a good amount of revenue, but it must be weighed against some costly preliminary projects that would be needed in order to make this service possible: increasing the fiber-optic capacity in the core of their network, and buying a newer generation of high-speed routers.

What makes these types of decisions particularly tricky is that they interact in complex ways: in isolation, the revenue from the high-speed access service might not be enough to justify modernizing the routers; *however*, once the company has modernized the routers, they'll also be in a position to pursue a lucrative additional project with their corporate customers; and maybe this additional project will tip the balance. And these interactions chain together: the corporate project actually would require another expense, but this in turn would enable two other lucrative projects—and so forth. In the end, the question is: Which projects should be pursued, and which should be passed up? It's a basic issue of balancing costs incurred with profitable opportunities that are made possible.

The Problem

Here's a very general framework for modeling a set of decisions such as this. There is an underlying set P of *projects*, and each project $i \in P$ has an associated *revenue* p_i, which can either be positive or negative. (In other words, each of the lucrative opportunities and costly infrastructure-building steps in our example above will be referred to as a separate project.) Certain projects are prerequisites for other projects, and we model this by an underlying directed acyclic graph $G = (P, E)$. The nodes of G are the projects, and there is an edge (i, j) to indicate that project i can only be selected if project j is selected as well. Note that a project i can have many prerequisites, and there can be many projects that have project j as one of their prerequisites. A set of projects $A \subseteq P$ is *feasible* if the prerequisite of every project in A also belongs to A: for each $i \in A$, and each edge $(i, j) \in E$, we also have $j \in A$. We will refer to requirements of this form as *precedence constraints*. The profit of a set of projects is defined to be

$$\texttt{profit}(A) = \sum_{i \in A} p_i.$$

The *Project Selection Problem* is to select a feasible set of projects with maximum profit.

This problem also became a hot topic of study in the mining literature, starting in the early 1960s; here it was called the *Open-Pit Mining Problem*.[3] Open-pit mining is a surface mining operation in which blocks of earth are extracted from the surface to retrieve the ore contained in them. Before the mining operation begins, the entire area is divided into a set P of *blocks*, and the net value p_i of each block is estimated: This is the value of the ore minus the processing costs, for this block considered in isolation. Some of these net values will be positive, others negative. The full set of blocks has precedence constraints that essentially prevent blocks from being extracted before others on top of them are extracted. The Open-Pit Mining Problem is to determine the most profitable set of blocks to extract, subject to the precedence constraints. This problem falls into the framework of project selection—each block corresponds to a separate project.

Designing the Algorithm

Here we will show that the Project Selection Problem can be solved by reducing it to a minimum-cut computation on an extended graph G', defined analogously to the graph we used in Section 7.10 for image segmentation. The idea is to construct G' from G in such a way that the source side of a minimum cut in G' will correspond to an optimal set of projects to select.

To form the graph G', we add a new source s and a new sink t to the graph G as shown in Figure 7.20. For each node $i \in P$ with $p_i > 0$, we add an edge (s, i) with capacity p_i. For each node $i \in P$ with $p_i < 0$, we add an edge (i, t) with capacity $-p_i$. We will set the capacities on the edges in G later. However, we can already see that the capacity of the cut $(\{s\}, P \cup \{t\})$ is $C = \sum_{i \in P: p_i > 0} p_i$, so the maximum-flow value in this network is at most C.

We want to ensure that if (A', B') is a minimum cut in this graph, then $A = A' - \{s\}$ obeys the precedence constraints; that is, if the node $i \in A$ has an edge $(i, j) \in E$, then we must have $j \in A$. The conceptually cleanest way to ensure this is to give each of the edges in G capacity of ∞. We haven't previously formalized what an infinite capacity would mean, but there is no problem in doing this: it is simply an edge for which the capacity condition imposes no upper bound at all. The algorithms of the previous sections, as well as the Max-Flow Min-Cut Theorem, carry over to handle infinite capacities. However, we can also avoid bringing in the notion of infinite capacities by

[3] In contrast to the field of data mining, which has motivated several of the problems we considered earlier, we're talking here about actual mining, where you dig things out of the ground.

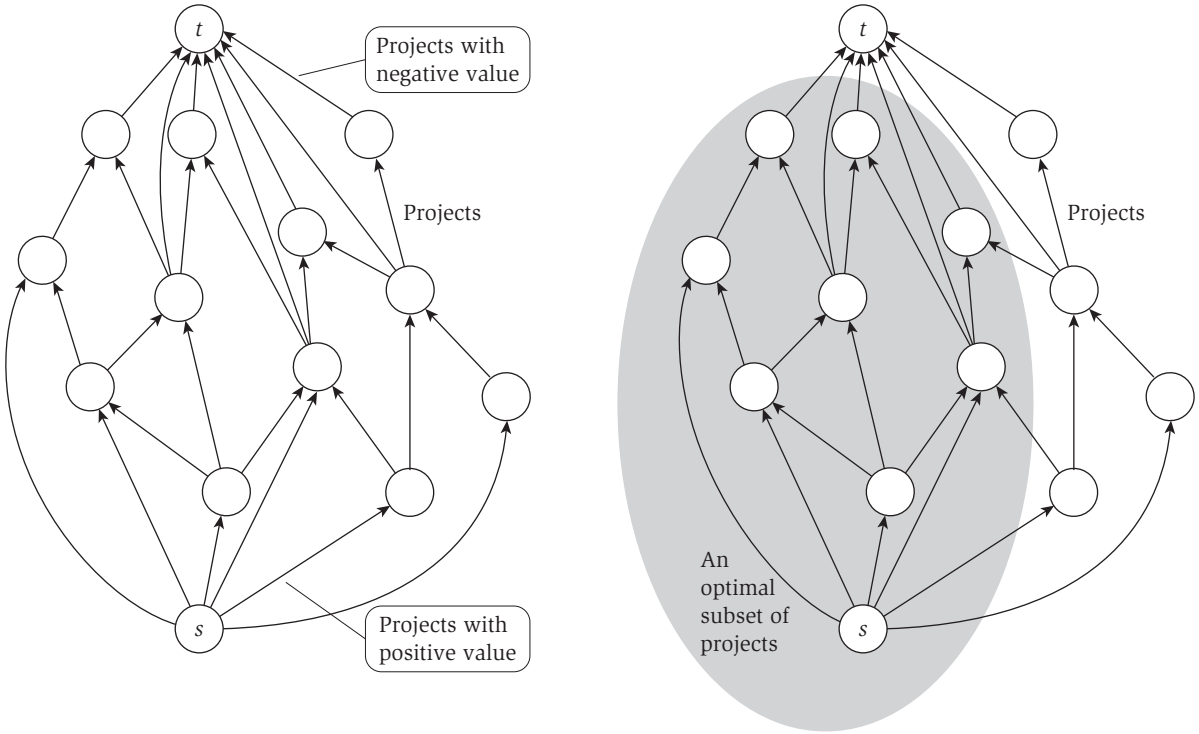

Figure 7.20 The flow graph used to solve the Project Selection Problem. A possible minimum-capacity cut is shown on the right.

simply assigning each of these edges a capacity that is "effectively infinite." In our context, giving each of these edges a capacity of $C + 1$ would accomplish this: The maximum possible flow value in G' is at most C, and so no minimum cut can contain an edge with capacity above C. In the description below, it will not matter which of these options we choose.

We can now state the algorithm: We compute a minimum cut (A', B') in G', and we declare $A' - \{s\}$ to be the optimal set of projects. We now turn to proving that this algorithm indeed gives the optimal solution.

Analyzing the Algorithm

First consider a set of projects A that satisfies the precedence constraints. Let $A' = A \cup \{s\}$ and $B' = (P - A) \cup \{t\}$, and consider the s-t cut (A', B'). If the set A satisfies the precedence constraints, then no edge $(i, j) \in E$ crosses this cut, as shown in Figure 7.20. The capacity of the cut can be expressed as follows.

(7.56) *The capacity of the cut (A', B'), as defined from a project set A satisfying the precedence constraints, is $c(A', B') = C - \sum_{i \in A} p_i$.*

Proof. Edges of G' can be divided into three categories: those corresponding to the edge set E of G, those leaving the source s, and those entering the sink t. Because A satisfies the precedence constraints, the edges in E do not cross the cut (A', B'), and hence do not contribute to its capacity. The edges entering the sink t contribute

$$\sum_{i \in A \text{ and } p_i < 0} -p_i$$

to the capacity of the cut, and the edges leaving the source s contribute

$$\sum_{i \notin A \text{ and } p_i > 0} p_i.$$

Using the definition of C, we can rewrite this latter quantity as $C - \sum_{i \in A \text{ and } p_i > 0} p_i$. The capacity of the cut (A', B') is the sum of these two terms, which is

$$\sum_{i \in A \text{ and } p_i < 0} (-p_i) + \left(C - \sum_{i \in A \text{ and } p_i > 0} p_i \right) = C - \sum_{i \in A} p_i,$$

as claimed. ■

Next, recall that edges of G have capacity more than $C = \sum_{i \in P : p_i > 0} p_i$, and so these edges cannot cross a cut of capacity at most C. This implies that such cuts define feasible sets of projects.

(7.57) *If (A', B') is a cut with capacity at most C, then the set $A = A' - \{s\}$ satisfies the precedence constraints.*

Now we can prove the main goal of our construction, that the minimum cut in G' determines the optimum set of projects. Putting the previous two claims together, we see that the cuts (A', B') of capacity at most C are in one-to-one correspondence with feasible sets of project $A = A' - \{s\}$. The capacity of such a cut (A', B') is

$$c(A', B') = C - \texttt{profit}(A).$$

The capacity value C is a constant, independent of the cut (A', B'), so the cut with minimum capacity corresponds to the set of projects A with maximum profit. We have therefore proved the following.

(7.58) *If (A', B') is a minimum cut in G' then the set $A = A' - \{s\}$ is an optimum solution to the Project Selection Problem.*