Lab Worksheet 7: Max Flow (Solution)

The Project Selection Problem

The project selection problem involves choosing a subset of activities to maximize total profit while satisfying prerequisite constraints. We are given a set P of potential projects. Each project $i \in P$ is associated with a real-valued *value* p_i, which may be positive (profit) or negative (cost).

The relationships between projects are defined by a directed acyclic Graph (DAG) $G = (P, E)$, where P is the set of vertices (projects) and E is the set of directed edges (constraints). An edge $(i, j) \in E$ represents a precedence constraint: if project i is selected, then its prerequisite, project j, *must* also be selected. A subset of projects $A \subseteq P$ is considered *feasible* if it satisfies all the constraints. This means that for every constraint $(i, j) \in E$, if $i \in A$, then j must also be in A.

The *total profit*, denoted by $\text{profit}(A)$, resulting from selecting a feasible set A is the sum of the values of all projects included in A:

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

The goal is to find a feasible set of projects $A \subseteq P$ that *maximizes the total profit*, $\text{profit}(A)$.

Designing the Flow Network

We solve this problem via a reduction to the max-flow problem. The idea is to construct a network flow from G in such a way that the minimum cut partition the graph G corresponds to an optimal set of projects to pick.

Constructing $G' = (V', E')$. We build a directed network with a new source s and sink t, and:

1. For each $i \in P$ with $p_i \geq 0$, add edge (s, i) of capacity p_i.
2. For each $i \in P$ with $p_i < 0$, add edge (i, t) of capacity $-p_i$.

3. For each precedence edge $(i, j) \in E$ ("to take i, you must take j"), add edge (i, j) of capacity $+\infty$[1].

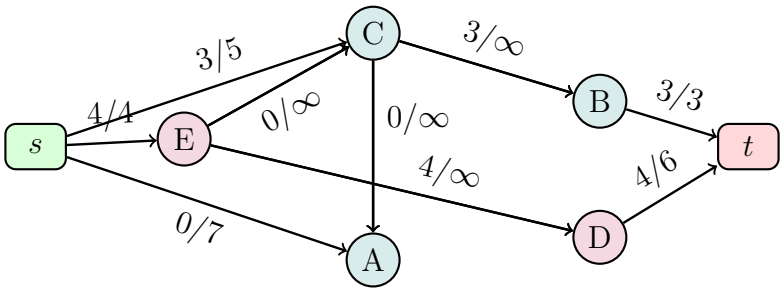
Question 1. Consider the example graph below and the following projects and values:

$$P = \{A, B, C, D, E\}, \quad p_A = 7,\ p_B = -3,\ p_C = 5,\ p_D = -6,\ p_E = 4,$$

For the given graph G below, construct the network G', run the Ford-Fulkerson algorithm on it, and find the minimum $(s - t)$-cut.

Solution: The Ford-Fulkerson algorithm terminates, finding a maximum flow of 7 units. This total flow is decomposed into 3 units pushed along the path $s \to A \to C \to B \to t$ and 4 units pushed along the path $s \to E \to D \to t$. From the final residual graph, we find the minimum cut (S, T) by partitioning the vertices into those reachable from s (set S) and those unreachable (set T). The partitions are $S = \{s, A, B, C\}$ and $T = \{E, D, t\}$.

We can verify this cut by examining the edges that cross between the sets: the edges from S to T, (s, E) and (B, t), are both *saturated* (i.e., flow equals capacity), and the edge from T to S, (E, C), has *zero flow*. This configuration confirms the *max-flow min-cut theorem*, as the capacity of the minimum cut is equal to the value of the maximum flow found.



Extracting the solution from a minimum cut

Question 2. Let $C \coloneqq \sum_{i \in P : p_i > 0} p_i$. This is the maximum possible profit that one can hope for without considering the precedent constraints. Let (S, T) be a minimum s–t cut in G'. Define $A \coloneqq S \setminus \{s\}$, representing the selected projects.

1. Explain why A must satisfy the precedence constraints.

   Solution: Note that the graph has a finite cut by setting $S = \{s\}$ and T being the rest of vertices other than s. This cut has capacity $C < \infty$. Precedence edges get capacity $+\infty$, so a minimum cut never use them; hence if i is on the s-side, all its prerequisites j must also lie on the s-side. This enforces feasibility via the cut.

[1] This can be implemented by setting the capacity to any number that is strictly larger than $C \coloneqq \sum_{i \in P :\, p_i > 0} p_i$, e.g. $C + 1$.

2. Show that the capacity of this cut is $C - \sum_{i \in A} p_i$.

   Solution: Since the min-cut value is finite, it cannot cross any of the infinite-capacity precedence edges from G. Thus, all edges in the cut must be the finite-capacity edges added during the construction, namely those connecting to the source s or the sink t. The edges contributing to the cut's capacity are partitioned into two groups: those (s, i) where $i \notin A$ and $p_i \geq 0$, and those (i, t) where $i \in A$ and $p_i < 0$. Using the construction, one obtains:

$$\begin{aligned}
\text{capacity}(S, T) &= \sum_{i \notin A,\, p_i \geq 0} p_i \; + \sum_{i \in A,\, p_i < 0} (-p_i) \\
&= \left(\sum_{i \notin A,\, p_i \geq 0} p_i + \sum_{i \in A,\, p_i \geq 0} p_i \right) - \left(\sum_{i \in A,\, p_i < 0} p_i + \sum_{i \in A,\, p_i \geq 0} p_i \right) \\
&= C - \sum_{i \in A} p_i \,.
\end{aligned}$$

Algorithm and Running Time

Question 3. Using a max-flow algorithm as a subroutine, provide pseudocode to solve the project selection problem. Analyze the running time of your algorithm. You may assume the max-flow subroutine runs in $O(|V|\,|E|^2)$ on any graph $G = (V, E)$.

Solution: Algorithm 1 describes the pseudocode for the project selection problem.

Algorithm 1 Project Selection via Max-Flow/Min-Cut

Input: A set of projects P with profits p_i, and a graph $G = (P, E)$ of precedence constraints.
Output: The optimal set of feasible projects $A \subseteq P$.
$C \leftarrow \sum_{i:\, p_i > 0} p_i$
Initialize graph $G' \leftarrow (\{s, t\} \cup P, \emptyset)$
for each $i \in P$ **do**
    if $p_i > 0$ **then**
        Add edge (s, i) to G' with capacity p_i
    else if $p_i < 0$ **then**
        Add edge (i, t) to G' with capacity $-p_i$
for each $(i, j) \in E$ **do**
    Add edge (i, j) to G' with capacity $+\infty$ (or $C + 1$)
Run a Max-Flow/Min-Cut algorithm on G'
Let (S, T) be the minimum $s - t$ cut
$A \leftarrow S \setminus \{s\}$
return A

Running time. Building the graph G' takes $O(|P| + |E|)$ time. The resulting graph G' has $|V'| = |P| + 2 = O(|P|)$ vertices and $|E'| = O(|P| + |E|)$ edges. Using the **Edmonds**-Karp algorithm for max-flow, the total running time is dominated by the max-flow routine, which runs in $O(|V'||E'|^2) = O(|P| \cdot (|P| + |E|)^2)$.

Proof of Correctness

Question 4 In this part, we prove the correctness of the algorithm by establishing the two key facts: completeness and soundness.

1. (*Completeness: A feasible set implies a cut.*) Let $A \subseteq P$ be any feasible set. Show that placing $A \cup \{s\}$ on the s-side and $(P \setminus A) \cup \{t\}$ on the t-side yields a cut of capacity $C - \sum_{i \in A} p_i$.

   Solution: Since A is closed under prerequisites (i.e., feasible), no precedence edge crosses the cut. Only edges of the form (s, i) with $i \notin A$ (and $p_i > 0$) and (i, t) with $i \in A$ (and $p_i < 0$) can cross. Similar to what we have shown erlier, summing the capacities gives capacity$(S, T) = C - \sum_{i \in A} p_i$.

2. (*Soundness: A finite capacity cut induces a feasible set.*) Let (S, T) be any $(s - t)$cut with capacity at most C. Prove that $A = S \setminus \{s\}$ satisfies the precedence constraints.

   Solution: Any cut of capacity at most C cannot cut a precedence edge (each has capacity $+\infty$). If the project set A were infeasible, there would exist an edge $(i, j) \in E$ such that $i \in A$ and $j \notin A$. This implies $i \in S$ and $j \in T$, which cuts the infinite capacity edge (i, j), contradicting the finite cut capacity. Thus, A must be feasible.

3. Combine the two parts to argue that a minimum cut corresponds to a maximum-profit feasible set.

   Solution: The two previous parts establish a one-to-one correspondence between feasible sets A and cuts (S, T) with capacity at most C, where the capacity is related to profit by capacity$(S, T) = C - \text{profit}(A)$. Since C is a constant, minimizing the cut capacity capacity(S, T) is equivalent to maximizing the project profit profit(A). Therefore, the s-side of any minimum cut is an optimal feasible set.