| **COMPSCI 330: Design and Analysis of Algorithms** | October 2, 2014 |
|---|---|
| Lecture # 12 | |
| *Lecturer: Debmalya Panigrahi* | *Scribe: Roger Zou* |

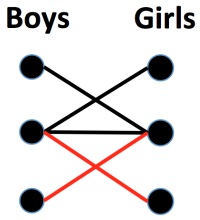
# 1 Overview

Previously we discussed the relationship between min and cut max flow, and introduced augmenting paths as a strategy to compute max flow. In this lecture we discuss applications of maximum flow, specifically for Bipartite Matching and Course selections. We show that maximum cardinality bipartite matching ("centralized dating"), and course selections to maximize "profit" given prerequisite course constraints, can be formulated and solved efficiently as max flow problems.

# 2 Bipartite Matching

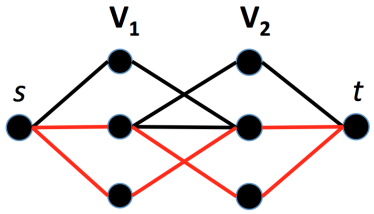
### Problem Statement

Suppose to maximize happiness Duke has centralized dating, where each boy selects girls he would potentially date, and each girl selects boys she would potentially date. The goal for Duke is to assign the maximum size pairing.
We can model this problem with bipartite matching: Let $G = (V_1 \cup V_2, E)$ be a **bipartite graph**, where $V_1 \cap V_2 = \emptyset$ and $E = \{(u,v) \in E \mid u \in V_1, v \in V_2\}$ (say the boys are $V_1$, girls are $V_2$). Note that there are no edges between any two elements in $V_1$, and similarly for $V_2$. A **matching** is a subset of edges $M \subseteq E$ *s.t.* $\{(u,v_1),(u,v_2)\} \in E' \implies (v_1 = v_2)$ and $\{(u_1,v),(u_2,v)\} \in E' \implies (u_1 = u_2)$. (one boy is only paired with one girl, and vice versa). A sample bipartite graph is shown below. A potential matching is colored red (this is actually a maximum matching). In this problem we wish to find the **maximum cardinality matching** (the most number of girls/boys are happy).




### First Attempt (Incorrect)

Add a source node $s$ and sink node $t$, and add edge $(s,u)\,\forall u \in V_1$, and $(v,t)\,\forall v \in V_2$. All capacities are set to 1. We will call this modified graph by the same name $G = (V_1, V_2, E)$ for simplicity and a matching $M \in E$. We let unit flow extend from $s$ to the vertices in $V_1$ included in the matching, and unit flow extend from all vertices in $V_2$ included in the matching to $t$.



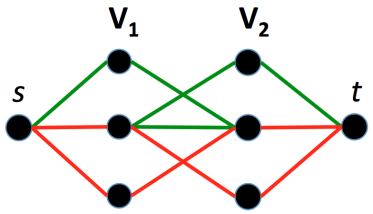

**Claim 1.** *Any matching $M \in E$ containing $k$ edges gives a flow of value $k$ from $s$ to $t$.*

*Proof.* We show that both capacity constraints and flow balance are satisfied. Capacity constraints are satisfied because flow is either 0 or 1 along any edge, and flow is set to 1 on every edge in the matching. Flow balance is satisfied because a vertex is not used more than once in the matching. For any vertex $u$ and $v$, if $(u,v) \in M$, then $f_{us} = f_{uv} = f_{vt} = 1$, where $f_{xy}$ denotes the flow along edge $(x,y)$. Similarly, if $(u,v) \notin M$, then $f_{us} = f_{uv} = f_{vt} = 0$. Thus flow in equals flow out for all vertices in $V \setminus \{s,t\}$. □

**Corollary 1.** *If there exists a maximum cardinality bipartite matching $M$ of size $k$ in $G$ (ignoring $s,t$), then there exists a maximum $s-t$ flow of size at least $k$ in $G$.*

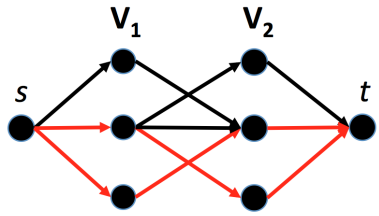
*Proof.* Let $M$ be a maximum cardinality bipartite matching of size $k$. Then from Claim 1, there exists a flow of size $k$. This might not be the max flow, so the maximum $s-t$ flow must be at least $k$. □

This looks like a promising start to finding the maximum matching. Unfortunately, it's wrong. See the pathological example below. The maximum matching is 2, with corresponding flows in red. The maximum flow however is 3, with the two maximum matches and an additional flow in green.




**Second Attempt (Correct)**

We were close in our first attempt. The reason the example above didn't work was that the edges were undirected, so we could go back from $V_2$ to $V_1$ without violating any flow constraints. An obvious option is to enforce directed edges. See the modified diagram below:




We need to prove that this modification is correct, and actually computes the maximum matching through maximum flow. First, note that Claim 1 still holds, because we haven't changed anything with the capacities

or which edges are chosen to have non-zero flow. To show that max matching equals max flow, we already showed in Corollary 1 that max matching $\leq$ max flow. Now we just need to show that max matching $\geq$ max flow.

**Claim 2.** *If there exists a maximum $s-t$ flow of size $k$ in $G$, then there exists a maximum cardinality bipartite matching in $G$ (ignoring $s,t$) of size at least $k$.*

*Proof.* Let $cut(S,\bar{S})$ be a cut of $G$, where $S = V_1 + \{s\}$ and $\bar{S} = V_2 + \{t\}$. Let $\delta(S)$ be the directed cut edges $(u,v)$, where $u \in S, v \in \bar{S}$. First, note that by construction, $\delta(S)$ can only have edges from $S$ to $\bar{S}$, and not the other way around. We showed in the previous lecture that indeed the $\sum_{e\in\delta(S)} f_e = f$, where $f$ is the max flow. Using flow balance, we can show that we find a matching of the same value. Observe that in $\bar{S}$, we can't have flow into any vertex $v \in V_2$ to be greater than 1 because flow out of $v$ has capacity 1 (to $t$). Similarly, in $S$, we can't have flow out of any vertex $u \in V_1$ to be greater than 1 because flow into $u$ has capacity 1 (from $s$). Thus, the only case is where an edge carries unit flow from a set of matching edges.

Since we don't know that the maximum matching, we can only say that there exists a maximum matching at least size $k$. However, Corollary 1 proved the other direction of the inequality. Thus the maximum matching equals the maximum flow. □

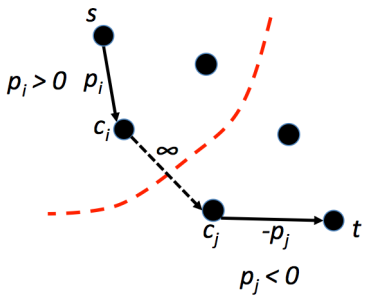
# 3 Course Selection

### Problem Statement

We have a set of courses, each with a "gain" and "pain" value. We also have a set of prerequisites for each course. We wish to find the best set of courses that maximizes the difference between gain and pain. More formally, let $C = \{c_1, c_2, \ldots, c_n\}$ be our set of courses. Let $\alpha_i \in \mathbb{R}_+$ be the gain of course $c_i$, and $\beta_i \in \mathbb{R}_+$ be the pain of course $c_i$. We define the profit $p_i$ of course $c_i$ to be $p_i = \alpha_i - \beta_i$, or the gain minus the pain. Each course $c_i$ has set of prerequisites $R_i \subseteq C$. Our goal is to choose $C' \subseteq C \, s.t. \, \forall c_i \in C', R_i \subseteq C'$ *and* $\max \sum_{c_i \in C'} p_i$.

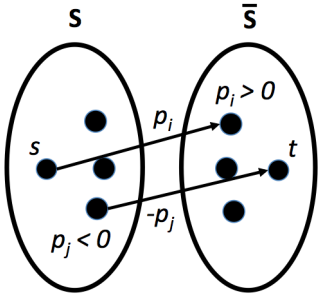
### Solution

Split the courses into ones taken and not taken. Create a source node $s$ and destination node $t$. For each class $c_i$ where $p_i > 0$, create an edge $(s, c_i)$ with capacity $p_i$. For each class $c_j$ where $p_j < 0$, create an edge $(c_j, t)$ with capacity $-p_j$. If $c_j$ is a prerequisite for $c_i$, add a edge from $c_i$ to $c_j$ of capacity $\infty$. See an example graph below:




First, note that the min cut of this graph satisfies the prerequisite property, because the min cut can't be infinite weight. We need to show that profit is maximized using this method through an argument with min cuts.

**Claim 3.** *The algorithm is correct (i.e. we maximize the course profit subject to prerequisite constraints).*




*Proof.* We wish to minimize the cut shown above. Note that the only two types of edges are shown. No $\infty$ edges are cut edges, or it wouldn't be the minimum cut. The value of the cut $v(S,\bar{S})$ is:

$$\begin{aligned} v(S,\bar{S}) &= \sum_{c_i \in \bar{S}: p_i > 0} p_i - \sum_{c_i \in S: p_i < 0} p_i \\ &= \sum_{c_i \in C: p_i > 0} p_i - \sum_{c_i \in S: p_i < 0 p_i} p_i - \sum_{c_i \in S: p_i < 0} p_i \\ &= \sum_{c_i \in C: p_i > 0} p_i - \sum_{c_i \in S} p_i \end{aligned}$$

Note that the term $\sum_{c_i \in C: p_i > 0} p_i$ is fixed, because this is the sum over all courses with positive profit. Thus, we are minimizing $-\sum_{c_i \in S} p_i$, which is equivalent to maximizing $\sum_{c_i \in S} p_i$. The result is exactly what we specified earlier in the problem statement. Thus the min cut is the correct solution. □

# 4 Summary

In this lecture we discussed two applications of maximum flows. The first was bipartite matching, for which we wish to find the maximum cardinality bipartite matching in a bipartite graph. The second application was finding the set of courses to take that maximizes a "profit" value given prerequisite constraints. In both cases, we were able to formulate our problem as a maximum flow/minimum cut problem, and prove that our algorithm is correct.