Math 3272: Linear Programming[1] Mikhail Lavrov

Lecture 23: Transversals and matchings

November 3, 2022 *Kennesaw State University*

1 Transversals

Problem 1. *At Network Flow University, there are many clubs and student organizations. There are so many that the students ran into a problem: how can they elect presidents for them all?*

Of course, every club must have a president (who must be a member of the club). Moreover, being president is a lot of work, so every student should be a member of only one *club. If we know the rosters of all the clubs, can we always choose a president for each one?*

Here is a particular instance of this problem—this one is probably much smaller than Network Flow University would really have to deal with, but it lets us look at something concrete. Say that there are six students at NFU; their names are Declan (D), Evelyn (E), Finn (F), Genevieve (G), Harper (H), and Jasper (J). There are also six clubs, whose rosters we'll abbreviate to:

$$A_1 : \{F, J\} \quad A_2 : \{D, E, F, H\} \quad A_3 : \{D, F, G, J\} \quad A_4 : \{E, J\} \quad A_5 : \{E, F\} \quad A_6 : \{E, F, J\}$$

Thinking more generally, we can pose this problem whenever we have a universe U (in this case, the universe is the university: the set of students $\{D, E, F, G, H, J\}$) and a family $\mathcal{F}$ of subsets of U (in this case, $\mathcal{F} = \{A_1, A_2, A_3, A_4, A_5, A_6\}$). Our goal is to find a **transversal of** $\mathcal{F}$: from each set in $\mathcal{F}$, we want to choose one element, without choosing the same element of U twice. Formally, we want to choose elements $a_1, a_2, a_3, a_4, a_5, a_6 \in U$ such that $a_i \in A_i$ for all i and whenever $i \neq j$, we have $a_i \neq a_j$.

You may already see some issues with our particular instance of this problem, but we will ignore those issues and keep going. Let's try to model this problem using integer flows.

The idea is that we want to set up a network so that "elect student u to be president of club A_i" will be represented by "send flow 1 from node i to node u". (We could also build a network to send flow the other way: the choice to send flow from clubs to students is arbitrary.) Of course, we only include an arc (i, u) if student u is a member of club A_i.

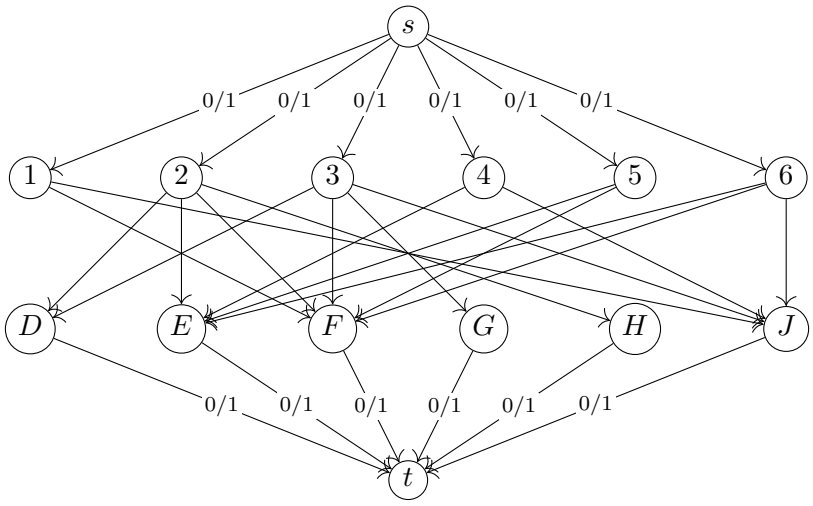
We cannot elect the same u to be president of multiple clubs; to enforce this, we can make sure that the flow out of u can be at most 1, limiting the flow into u to 1 as well.

We want to make sure every club A_i gets a president; this would mean making sure that the flow into node i is at least 1. This is not something we can really enforce in a network. Maximum flow problems are optimization problems; to make this an optimization problem, we can ask "what is the maximum number of clubs that can have presidents?"

Now we can add arcs from the source s to nodes $1, 2, 3, 4, 5, 6$ with capacity 1 each; sending 1 flow along arc (s, i) means that the flow out of node i will be 1, forcing club A_i to elect a president.

[1] This document comes from an archive of the Math 3272 course webpage: http://misha.fish/archive/3272-fall-2022

Here is the resulting network:



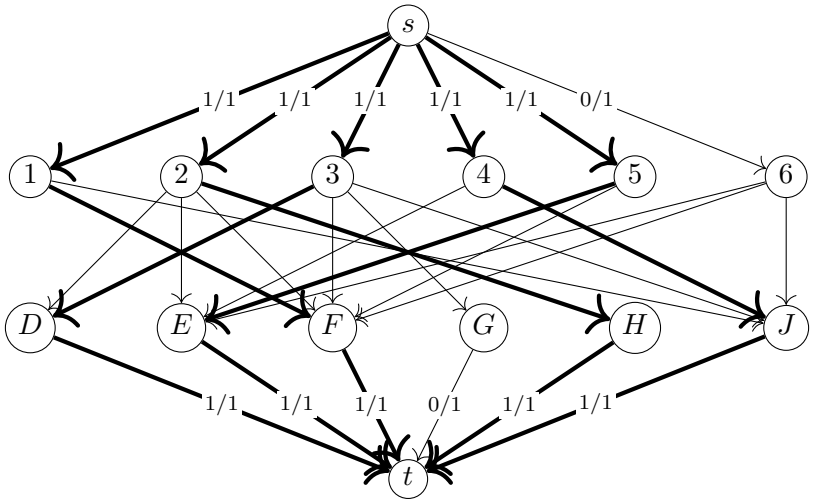
We did not put any capacities on arcs from clubs to students. In principle, we *could* put a capacity of 1 on each such arc, since a club cannot elect a student to be "double president". But that constraint is already implied by the fact that a student cannot be elected president twice (not even by the same club).

Instead, we will give these arcs infinite capacity. In the linear program, we can represent this by just not having a capacity constraint. For the Ford–Fulkerson method, using ∞ as a capacity is fine if we're careful, but we could also use any big number M. The reason to use a large capacity on these arcs is that later on, it will limit the cuts we can find to ones with a nice structure.

2 Hall's theorem

What happens if we try to solve this network flow problem?

It turns out that the maximum value of any flow is 5. There are many solutions. Here is one of them; we will use a thick edge to make it clear which arcs from clubs to students have flow 1:



So it turns out we cannot elect a president for every club. Why not? For an explanation, we must turn to the minimum cut.

We did not draw the residual graph, but let's try to reconstruct it in our heads: going forward along arcs where the capacity can be increased, and backwards where the capacity can be decreased. From s, we can only go to 6 in the residual graph. From 6, we can go to E, F, and J. From these, we cannot take any forward arcs (those are at their maximum capacity), but we can go along the backward arcs $(5, E)$ $(1, F)$, and $(4, J)$ with positive flow to get to nodes $1, 4, 5$. Finally, from these nodes, we can only go forward, but the only forward arcs lead to nodes E, F, and J, which we've seen already. So the minimum cut (S, T) we find has

$$S = \{s, 1, 4, 5, 6, E, F, J\}, \qquad T = \{2, 3, D, G, H, t\}.$$

The capacity of this cut is 5: the arcs going from S to T are the arcs $(s, 2)$, $(s, 3)$, (E, t), (F, t), and (G, t), which have capacity 1 each.

We can interpret this cut to identify an issue with the club rosters we started with. The four clubs A_1, A_4, A_5, A_6 have only three students between them altogether: Evelyn, Finn, and Jasper. So how could they possibly elect four different presidents?

This tells how to identify a general obstacle to the existence of a transversal of a family $\mathcal{F}$: a transversal cannot exist if $\mathcal{F}$ has a subfamily $\mathcal{G}$ such that $\bigcup \mathcal{G}$ (the union of all sets in $\mathcal{G}$) has fewer than $\mathcal{G}$ elements. In our case, $\mathcal{G} = \{A_1, A_4, A_5, A_6\}$ and

$$\bigcup \mathcal{G} = \{F, J\} \cup \{E, J\} \cup \{E, F\} \cup \{E, F, J\} = \{E, F, J\},$$

so $|\mathcal{G}| = 4$ and $|\bigcup \mathcal{G}| = 3$.

In fact, we can prove that this is the *only* kind of obstacle that can rule out a transversal! To do this, we have to look at the structure of cuts in the network carefully.

Suppose that we are looking at a general example with $\mathcal{F} = \{A_1, A_2, \ldots, A_n\}$. Our network has four "layers": one layer with just s, one layer with nodes $\{1, 2, \ldots, n\}$, one layer with nodes U (the universe), and one layer with just t. Let's begin choosing our cut by saying that the set S will contain s and a subset $I \subseteq \{1, 2, \ldots, n\}$.

For every $i \in I$ and every $x \in A_i$, there is an arc (i, x) in our network with infinite capacity! We are trying to find a cut with small capacity, and it seems reasonable to aim for "not infinite" as a starting point toward "small". So we should make sure to include every such x in S as well, so that arc (i, x) does not cross from S to T. In other words, S must include the entire union $\bigcup_{i \in I} A_i$.

At this point, the following arcs cross from S to T:

- The arcs (s, j) for every $j \notin I$, since these are the nodes from $\{1, 2, \ldots, n\}$ we decided not to include in S. The total capacity of these is $n - |I|$.
- The arcs (x, t) for every $x \in S$, since we definitely have $t \in T$. The total capacity of these is $|\bigcup_{i \in I} A_i|$.
- Actually, we could add more elements of U to S, but this would only increase our capacity, so it wouldn't help.

This cut proves that we don't have a transversal exactly when its total capacity $n - |I| + |\bigcup_{i \in I} A_i|$ is less than n: in other words, when $|\bigcup_{i \in I} A_i| < |I|$. In other words, the subfamily $\mathcal{G} = \{A_i : i \in I\}$ has $|\bigcup \mathcal{G}| < |\mathcal{G}|$: this is the same type of obstacle that we identified earlier.

What we've proved is a result known as Hall's theorem:

Theorem 1 (Hall's theorem)**.** *A family $\mathcal{F}$ has a transversal if and only if every subset $\mathcal{G} \subseteq \mathcal{F}$ satisfies **Hall's condition**: $|\bigcup \mathcal{G}| \geq |\mathcal{G}|$.*

Many results about when transversals are guaranteed to exist can be shown using Hall's theorem. For example:

Theorem 2. *If every set in $\mathcal{F}$ has the same size k, every element of U occurs in ℓ sets of $\mathcal{F}$, and $k \geq \ell$, then $\mathcal{F}$ has a transversal.*

Proof 1 (using Hall's theorem). For an arbitrary subset $\mathcal{G} \subseteq \mathcal{F}$, a naive count of $|\bigcup \mathcal{G}|$ would say: well, there are $|\mathcal{G}|$ sets, and each one has k elements, so the union has $k|\mathcal{G}|$ elements. This is false when the sets in $\mathcal{G}$ are not disjoint, since the same element can be counted multiple times. However, we will never count an element more than ℓ times, since every element of U appears in ℓ sets of $\mathcal{F}$. Therefore we can at least say that $|\bigcup \mathcal{G}| \geq \frac{k}{\ell}|\mathcal{G}|$. Since $k \geq \ell$, this implies that $|\bigcup \mathcal{G}| \geq |\mathcal{G}|$, so Hall's condition holds for all $\mathcal{G}$. □

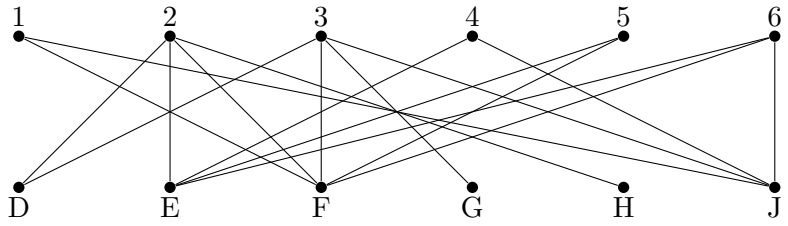
Proof 2 (using network flows). To simplify notation, let $\mathcal{F} = \{A_1, A_2, \ldots, A_n\}$.

The network for this problem has a fractional flow with value n, which is the maximum possible. Send one flow along every arc (s, i) with $1 \leq i \leq n$; split it up into $\frac{1}{k}$ flow along every arc (i, x) with $x \in A_i$; finally, at each $x \in U$, we end up with a total $\frac{\ell}{k} \leq 1$ flow going in, so we can send $\frac{\ell}{k}$ flow out to t.

By the integral flow theorem, there is also an integer flow with value n, which corresponds to a transversal. □

3 Vertex covers and matchings

Often, problems such as the club-president problem are represented by bipartite graphs. You would see this in more detail in a class all about graph theory, but essentially these are just a condensed drawing of our network, leaving out s and t, such as:



Rather than "nodes" and "arcs", it's more common to say **vertices** and **edges** in the case of a graph, but these mean nearly the ssame thing, except that edges don't have a direction. The graph is **bipartite** because its vertices can be partitioned into two sets X and Y, such that every

edge has one endpoint in X and one endpoint in Y. Let's say that $X = \{1, 2, 3, 4, 5, 6\}$ and $Y = \{D, E, F, G, H, J\}$ in our case.

In any flow, the arcs we use (except for the ones out of s and into t, which are gone now) cannot repeat a node. The equivalent notion in a graph is called a **matching**: a set of edges that does not repeat any endpoints. The solution we found in the previous section corresponds to the matching $\{1F, 2H, 3D, 4J, 5E\}$.

It is common to see problems where X and Y are the same size. In this case, a transversal of a set family corresponds to a **perfect matching** in a graph: this is a matching which uses up all the vertices in X and in Y.

What about the equivalent notion to a cut in our network? For this, it's convenient to turn our pair (S, T) into a different sort of object. We'll take a set C with the following elements:

- All vertices in X (the side that used to be closer to s) which are in T, and
- All vertices in Y (the side that used to be closer to t) which are in S.

In our example with $S = \{s, 1, 4, 5, 6, E, F, J\}$ and $T = \{2, 3, D, G, H, t\}$, we end up with $C = \{2, 3, E, F, J\}$.

Why do we do this? Well, the defining feature of a cut with finite capacity is that we cannot have an infinite capacity arc going from S to T; in our graph, this means that there cannot be an edge such that neither endpoint is in C. The resulting set C is always a **vertex cover**: a set of vertices that includes one endpoint of every edge.

The max-flow min-cut theorem has the following result when we translate it into the language of bipartite graphs:

Theorem 3 (König's theorem)**.** *In any bipartite graph, the number of edges in a maximum matching is equal to the number of vertices in a minimum vertex cover.*

Hall's theorem could also also be stated in the language of graphs, as a condition for the existence of a perfect matching. Often, that is the way you first see it. However, no matter what language we use, Hall's theorem and König's theorem have slightly different applications:

- Hall's theorem asks: when can we guarantee that we can find a perfect matching between X and Y? This sees more use in theoretical applications, where we are looking for a bijection of a certain type between two sets.
- König's theorem asks: what is the size of the largest matching we can find? This sees more use in practical applications, where we might want a large matching even if a perfect one does not exist.

In both cases, the best way to answer the question is to solve the network flow problem.