

6.046 Recitation 11 Handout  
May 2, 2008

# 1 Max Flow as a Linear Program

As a reminder, a linear program is a problem that can be written as that of fulfilling an objective function and a set of constraints on a set of variables:

$$\begin{aligned} \max \quad & c \cdot x \\ \text{subject to} \quad & Ax \leq b \end{aligned}$$

As we learned this week, max flow is the problem of finding a maximum feasible flow in a directed graph  $G = (V, E)$  from the source  $s$  to the sink  $t$  subject to edge capacities  $c(u, v) \geq 0$ . Max flow translates fairly directly into a linear program on the variables  $f(u, v)$  for  $(u, v) \in V \times V$ .

$$\begin{aligned} \max \quad & \sum_{v \in V} f(s, v) \\ \text{subject to} \quad & f(u, v) \leq c(u, v) \quad \forall u, v \in V \\ & f(u, v) = -f(v, u) \quad \forall u, v \in V \\ & \sum_{v \in V} f(u, v) = 0 \quad \forall u \in V - \{s, t\} \end{aligned}$$

# 2 Matching

Given an undirected graph  $G = (V, E)$ , a *matching* is a subset of edges  $M \subseteq E$  such that  $\forall v \in V$ , at most one edge of  $M$  is incident on  $v$ .

A vertex  $v$  is *matched* if some edge in  $M$  is incident on  $v$ ; otherwise,  $v$  is *unmatched*.

A *maximum matching* is a matching  $M$  of maximum cardinality (the most edges possible). See Figure 1 for an example.

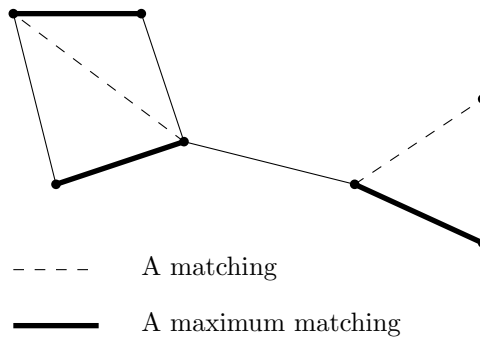


Figure 1: Here is an example of a graph and different sets of edges that form a valid matching and a maximum matching, respectively.

## 2.1 Maximum Bipartite Matching

The special case of finding a matching on a bipartite graph. A graph  $G = (V, E)$  is *bipartite* if  $V$  can be partitioned into two disjoint sets  $V = L \cup R$  such that  $\forall (u, v) \in E$ , either  $u \in L$  and  $v \in R$  or  $u \in R$  and  $v \in L$ , i.e. there are no edges connecting members within  $L$  or within  $R$ . See Figure 2 for an example.

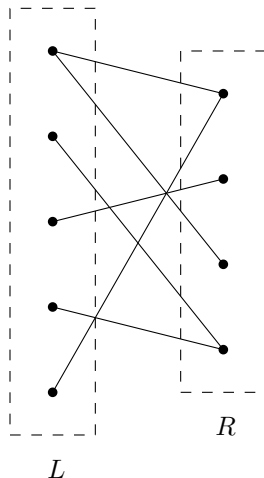


Figure 2: Here is a bipartite graph and the respective partitions of  $V$ .

Applications of finding a matching on a bipartite graph:

- Modeling a set  $L$  of machines and a set  $R$  of tasks; edges represent which tasks can be assigned on which machine. Can also think of this as a set of professors and a set of classes with edges between professors and classes if the professor is willing to teach that class. Solving this problem occurs each term since various professors go on sabbatical or choose to teach special topics courses, etc.
- Dating with edges representing compatibility.

### 2.1.1 Modeling Bipartite Matching as a Flow Problem

We can use the bipartite graph to make a new network  $G' = (V', E')$  on which finding the maximum flow gives us a matching. To do this, we do the following (see Figure 3 for an example):

1. Vertices: Add two vertices  $s$  and  $t$  to the graph.
2. Edges: Reuse edges from  $G$  and connect the graph to the source and sink. In particular,

$$\begin{aligned} E' = & \{(s, u) | u \in L\} \\ & \cup \{(u, v) | u \in L, v \in R, (u, v) \in E\} \\ & \cup \{(v, t) | v \in R\} \end{aligned}$$

3. Capacities on the edges: All edges have unit capacity.

By examining the cut  $(s, V' - s)$ , we get a simple bound that the value of the maximum flow in this network is  $|f'| = O(V)$ , and we have  $|E'| = O(V + E)$ , so if we just use Ford-Fulkerson, which runs on a graph  $G$  in  $O(E|f|)$  time, we get a running time of  $O(V(V + E)) = O(V^2 + VE)$ .

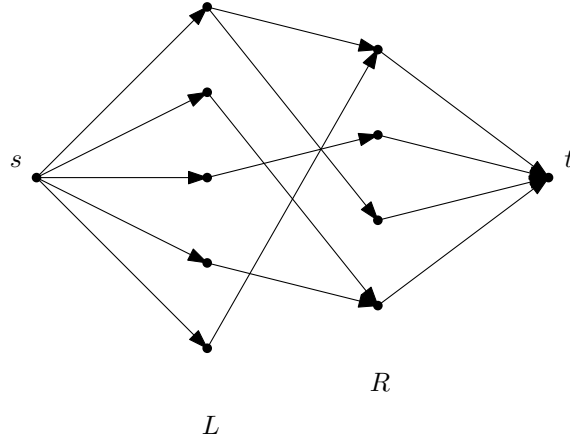


Figure 3: A network  $G'$  for finding a matching on the original bipartite graph  $G$ . All edges have unit capacity.

### 2.1.2 Correctness

We need to show that a matching corresponds to a flow in  $G'$ . A flow  $f$  is *integer-valued* if  $f(u, v)$  is an integer for all  $u, v \in V$ . We will show that we need the max flow we find in  $G'$  to be an integer-valued flow to easily obtain a matching out of it. For example, in Figure 4, a valid but not integer-valued max flow has an ambiguous matching. It can easily be fixed by diverting fractional flow to create integral flow, but there is no immediate matching that corresponds to a fractional flow.

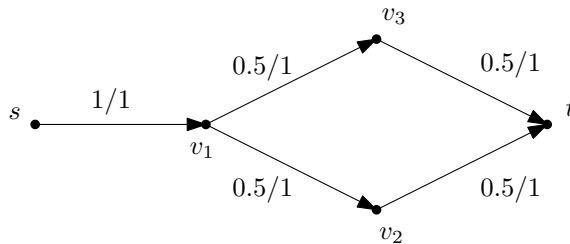


Figure 4: In this graph, the first edge leading from  $s$  serves as a bottleneck, and the fractional max flow diverts  $1/2$  unit of flow along the rest of the edges. A matching can be either between  $v_1$  and  $v_2$  or between  $v_1$  and  $v_3$ , and we would be able to get it from a max flow if only the flow were integral.

**Lemma 1.** *Let  $G = (V, E)$  be a bipartite graph with vertex partition  $V = (L, R)$  and let  $G' = (V', E')$  be the corresponding flow network. We then have the following properties.*

1. *If  $M$  is a matching in  $G$ , then there is an integer-valued flow  $f$  in  $G'$  with value  $|f| = |M|$ .*
2. *Conversely, if  $f$  is an integer-valued flow in  $G$ , then there is a matching  $M$  in  $G$  with cardinality  $|M| = |f|$ .*

*Proof.* We will prove both statements, respectively.

1. We first prove the first statement (that we can get a flow  $f$  from a matching  $M$ ) by construction. Given the matching  $M$ , define  $f$  as follows:

- If  $(u, v) \in M$ , then  $f(s, u) = f(u, v) = f(v, t) = 1$  and  $f(u, s) = f(v, u) = f(t, v) = -1$ . This preserves capacity constraints, skew symmetry, and flow conservation.
- For all other edges  $(u, v) \in E'$ , set  $f(u, v) = 0$ .

Essentially, the flow on the graph flows along paths that saturate the edges between  $L$  and  $R$  that are in  $M$ . In addition, if we look at the cut  $(L \cup \{s\}, R \cup \{t\})$  straight down the middle of the graph, the net flow across it is  $|M|$  since we saturated exactly one unit-capacity edge going from  $L$  to  $R$  for each edge in  $M$ . Thus  $|f| = |M|$ .

2. We next prove that if there is an integer-valued flow  $f$ , then there is a matching. We first show that we can find some matching. We claim that  $M = \{(u, v) | u \in L, v \in R, f(u, v) > 0\}$  is such a matching. Note that for each vertex  $u \in L$ , the vertex has only one entering edge with unit capacity. This means that at most 1 unit of positive flow can enter the vertex, and that if it does, then by flow conservation it must also leave the node. In fact, because the flow is integer-valued, the unit of flow must leave on exactly one outgoing edge. A symmetric argument can be made for flow leaving and entering nodes in  $R$ . Together, these arguments imply that no vertex can have more than one incident edge that connects  $L$  to  $R$  that has positive flow. Thus,  $M$  is a matching.

To see why  $|M| = |f|$ , we note the following statements are true due to flow conservation and skew symmetry:

$$\begin{aligned}
 |M| &= f(L, R) \\
 &= f(L, V') - f(L, L) - f(L, s) - f(L, t) \\
 &= 0 - 0 + f(s, L) - 0 \\
 &= |f|
 \end{aligned}$$

□

We're done, right? Actually, we need to make sure we get an integer-valued flow with our max flow algorithm. We will prove it more generally for running Ford-Fulkerson on any graph with integer capacities.

**Theorem 1 (Integrality).** *If a graph  $G$  only has integral capacities, then the max flow produced by the Ford-Fulkerson method (repeatedly finding augmenting paths and saturating them) has the property that  $|f|$  is integer-valued. Moreover, for all vertices  $u$  and  $v$ , the value of  $f(u, v)$  is an integer.*

*Proof.* Proof by induction on iteration of Ford-Fulkerson.

- Base case: No iterations performed yet. The flow is integral because it's 0.
- Inductive case: at  $i$ th iteration and all previous iterations resulted in integral flow. This implies all residual capacities are also integral (addition and subtraction with integers result in more integers). As a result, the augmenting path found in this iteration has an integral bottleneck, and the algorithm pushes an integral amount of flow on each edge in the augmenting path, thus keeping all flows integral.

□

By proving that Ford-Fulkerson always produces an integral max flow with integral flow on the edges given a graph with integral capacities, we know that for any graph with integral capacities, there must exist some max flow with those properties.

### 2.1.3 Hall's Theorem

A *perfect matching* is a matching in which every vertex is matched. As before, let  $G = (V, E)$  be an undirected bipartite graph with vertex partition  $V = L \cup R$ , where  $|L| = |R|$ . For any  $X \subseteq V$ , define the *neighborhood* of  $X$  as

$$N(X) = \{y \in V \mid (x, y) \in E \text{ for some } x \in X\}$$

**Theorem 2** (Hall's Theorem). *There exists a perfect matching in  $G$  iff  $|A| \leq |N(A)|$  for every subset  $A \subseteq L$ .*

*Proof.* We shall prove the statement by proving both directions.

$\Rightarrow$  If there is a subset  $X$  of  $L$  such that  $N(X) < |X|$ , then there is some element in  $X$  that cannot be matched.

$\Leftarrow$  Use induction on size of  $L$

- Base case:  $n = 1$ . Trivially true. We can clearly match the one member in  $L$  with its neighbor in  $R$  and achieve a perfect matching.
- Inductive case: Assume true for size  $L < n$ . We will consider 2 cases.
  1. For *every* subset  $X$  of any size,  $N(X) > |X|$ , i.e. the inequality is strict. In this case, we pair any two members  $u$  and  $v$  that are connected by an edge. We still have  $N(X) \geq |X|$  for all subsets of the  $n - 1$  leftover elements in  $L$  because  $N(X)$  decreased by at most 1 for any subset  $X$ . Since the inequality was strict before, the worst that happens is that it isn't anymore.
  2. If there is some subset such that  $N(X) = |X|$ , then it's clear that for any perfect matching to exist, the members  $X$  must be paired with the members of  $N(X)$ . Let  $L' = X$  and  $R' = N(X)$ . If some member in  $R'$  were paired with some other element not in  $L'$ , then there can be no perfect matching because we get  $N(X) < |X|$ . By induction, we know that we can solve this subproblem for the size  $|L'|$ .

Next we show that the induction hypothesis will remain satisfied after pairing off  $L'$  and  $R'$ . That is, for all subsets  $Y$  of  $L - L'$ ,  $N(Y) \geq |Y|$  is true.

Consider any subset  $X \subseteq L$  such that  $L' \subset X$ . Remove  $L'$  from  $X$  to form  $Y$ , and remove  $R'$  from  $N(X)$  to form  $N(Y)$ . Since  $N(X) \geq |X|$  and  $|L'| = |R'|$ , we still have  $N(Y) \geq |Y|$ . This argument holds for any subset  $X$ , so we're done.

□