Lecture 18 Maximum Bipartite Matching

MCS/CS 401 Computer Algorithms I — July 29, 2022
Instructor: Danko Adrovic

Lecture Outline

- Maximum Bipartite Matching
  - Introduction
  - Bipartite graphs
  - The maximum bipartite matching problem $\times$
  - Finding a maximum bipartite matching
  - Polynomial time reduction verification $\leftarrow$
  - Ford-Fulkerson returns correct result $\times$

1 Maximum Bipartite Matching

1.1 Introduction

Certain combinatorial problems can easily be recast as *maximum flow* problems. The surprising aspect is that two such problems, on the surface at least, appear not to be connected at all. One such example is the connection of the *maximum bipartite matching* problem to the maximum flow problem. In particular, with only relatively few steps, we can convert the maximum bipartite matching into a maximum flow problem. Consequently, because we can solve the maximum flow problem in polynomial time, we can also solve the maximum bipartite matching problem in polynomial time as well. In particular, we can say that the *maximum bipartite matching problem is* ***at most*** *as hard as the maximum flow problem* to solve.

$\text{MBMP} \leq \text{MAX-FLOW}$

While maximum bipartite matching problem is interesting and important in its own right, we want to use the connection between maximum flow and maximum bipartite matching as our starting point to learn about *polynomial-time reductions* of one seemingly unrelated problem to another problem. This kind of reduction will serve as a tool to classify problems into categories of problems that are all equally "easy" or equally "hard" to solve.

Today, we will approach polynomial-time reductions by example only. Next time we will formalize the idea of a polynomial-time reduction to introduce *NP-Completeness* and related topics. NP-Completeness and related topics form the core of theoretical computer science with important implications on solvability/decidebility of problems using algorithms.

V E

1.2 Bipartite graphs

Definition 1. *A bipartite graph is an undirected graph $G = (V, E)$ whose vertices can be divided into two disjoint and independent sets L and R such that every edge connects a vertex in L to one in R.*

$$\begin{tikzcd} & V \arrow[dl] \arrow[dr] & \\ L & & R \end{tikzcd}$$

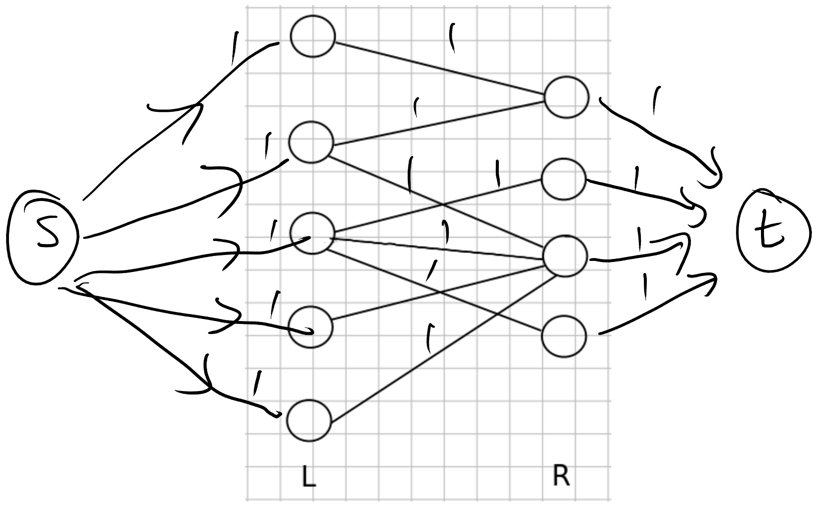


Figure 1: A bipartite graph.

1.3 The maximum bipartite matching problem

Definition 2 (Matching)**.** *Given an undirected graph $G = (V, E)$, a* ***matching*** *is a subset of edges $M \subseteq E$ such that for all vertices $v \in V$ at most one edge of M is incident on v. We say that a vertex $v \in V$ is* ***matched*** *by the matching M if some edge in M is incident on v. Otherwise, v is* ***unmatched****.*

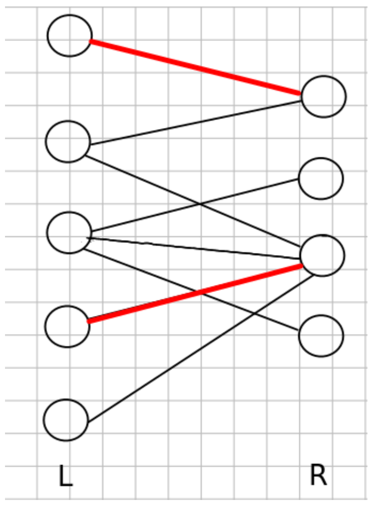
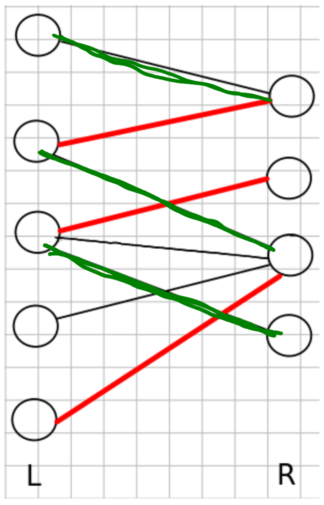


Figure 2: A matching with value 2.

Definition 3 (Maximum Matching)**.** *A maximum matching is a matching of maximum cardinality, that is, a matching M such that for any matching M', we have $|M| \geq |M'|$.*



3 G

3 R

Figure 3: A maximum matching with value 3.

Neither a matching or maximal matching must be unique. In particular, for the maximal matching case, there can be different configurations of edges that form a maximal matching but they all have a same maximal cardinality.

The problem of finding a maximum matching in a bipartite graph has many practical applications. As an example, we might consider matching a set L of machines with a set R of tasks to be performed simultaneously. We take the presence of edge $(u, v) \in E$ to mean that a particular machine $u \in L$ is capable of performing a particular task $v \in R$. A maximum matching provides work for as many machines as possible.

1.4 Finding a maximum bipartite matching

We can use the Ford-Fulkerson method to find a maximum matching in an undirected bipartite graph $G = (V, E)$ in time polynomial in $|V|$ and $|E|$. The *trick* is to construct a flow network in which flows correspond to matchings. ✓ ✓

G

We define the **corresponding flow network** $G' = (V', E')$ for the bipartite graph G as follows. We let the source s and sink t be new vertices not in V , and we let $V' = V \cup \{s, t\}$. If the vertex partition of G is $V = L \cup R$, the directed edges of G' are the edges of E, directed from L to R, along with $|V|$ new directed edges:

$$\longrightarrow E' = \{(s, u) : u \in L\} \cup \{(u, v) \in E\} \cup \{(v, t) : v \in R\}$$

To complete the construction, we assign unit capacity to each edge in E'.
An illustration of this process is given by the following figure.

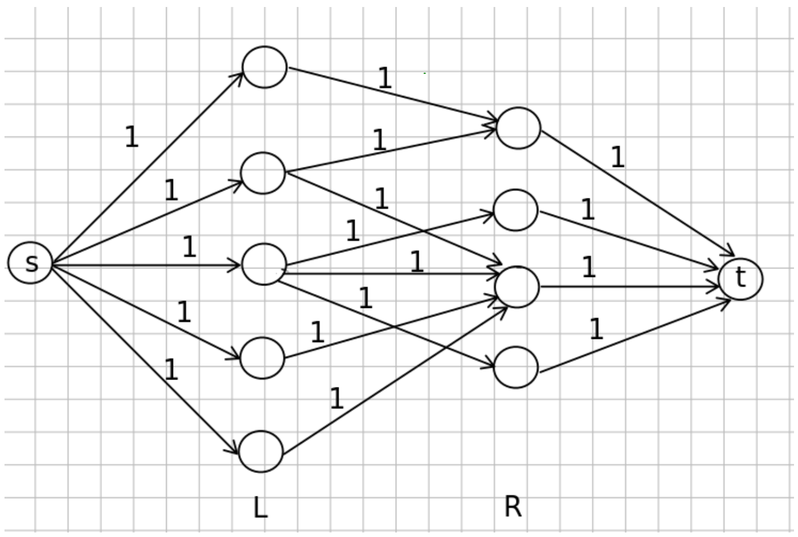


Figure 4: The corresponding flow network G'

1.5 Polynomial time reduction verification

How much effort did the reduction from one problem to another take? That is, how many more vertices and edges had to be connected to turn this maximum bipartite matching problem into a maximum flow problem? Was the reduction achieved in polynomial time?

We know that we had to add two extra vertices, the sink s and source t. How about edges? Since each vertex in V has at least one incident edge, $|E| \geq |V|/2$ - or $|V| \leq 2|E|$. Because $|E'|$ contains all the edges in E, we can bound $|E'|$ as $|E| \leq |E'|$. By adding s and t to the graph and connecting them, the number of edges grows by exactly $|V|$. That means we can further bound $|E'|$ as $|E| \leq |E'| = |E| + |V|$. The term $|V|$ we have already bounded by $|V| \leq 2|E|$, so we expand the previous to $|E| \leq |E'| = |E| + 2|E| = 3|E| \leq 3|E|$. Therefore, $|E'| = \Theta(E)$. That is, we have an asymptotic tight bound on the number of edges in the new problem and it is bounded by the number of edges in the original problem $|E|$.

$|E'| = \Theta(E)$

The reduction took only an additional 2 vertices and $\Theta(E)$ edges to construct, both of which are polynomial. We can then alter (expand) the data structures (adjacency list representation of a graph, and conversion to weighted, directed graph) containing the original problems also in polynomial time. So, this whole reduction can be performed in polynomial time.

All that remains is to run the **Ford-Fulkerson** algorithm on the new problem and we obtain the result.

1.6 Ford-Fulkerson returns correct result

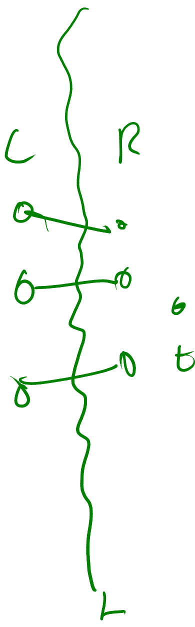
For the subsequent results, we will need the following Lemma:

Lemma 1 (Lemma 26.4, page 721). *Let f be a flow in a flow network G with source s and sink t, and let (S, T) be any cut of G. Then the net flow across (S, T) is $f(S, T) = |f|$.*

cut

How do we know that **Ford-Fulkerson** algorithm solves this problem correctly? The reduction seems plausible but it is a stretch to claim that one problem solution leads to solution of another.

The following lemma shows that a matching in G corresponds directly to a flow in G's corresponding flow network G'. We say that a flow f on a flow network $G = (V, E)$ is **integer-valued** if $f(u, v)$ is an integer for all $(u, v) \in V \times V$.

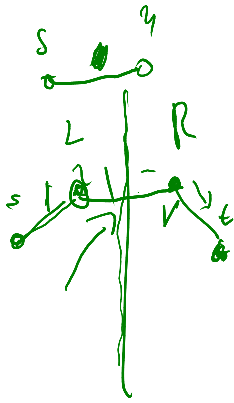


Lemma 2 (Lemma 26.9)**.** *Let $G = (V, E)$ be a bipartite graph with vertex partition $V = L \cup R$, and let $G' = (V', E')$ be its corresponding flow network. If M is a matching in G, then there is an integer-valued flow f in G' with value $|f| = M$. Conversely, if f is an integer-valued flow in G', then there is a matching M in G with cardinality $|M| = |f|$.*

Proof. We first show that a matching M in G corresponds to an integer-valued flow f in G'. Define f as follows. If $(u, v) \in M$, then $f(s, u) = f(u, v) = f(v, t) = 1$. For all other edges $(u, v) \in E'$, we define $f(u, v) = 0$. It is easy to verify that f satisfies the **capacity constraint** and **flow conservation**, see previous lecture.

Intuitively, each edge $(u, v) \in M$ corresponds to one unit of flow in G' that traverses the path $s \to u \to v \to t$. Moreover, the paths induced by edges in M are vertex-disjoint, except for s and t. The net flow across cut $(L \cup \{s\}, R \cup \{t\})$ is equal to $|M|$. Therefore, by the Lemma 26.4, the value of the flow is $|f| = |M|$.

In order to prove the converse, let f be an integer-valued flow in G' and let $M = \{(u, v) : u \in L, v \in R, and f(u, v) > 0\}$.



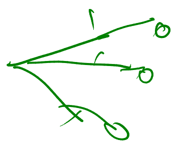
Each vertex $u \in L$ has only one entering edge, namely (s, u), and its capacity is 1. Thus, each $u \in L$ has at most one unit of flow entering it, and if one unit of flow does enter, by flow conservation, one unit of flow must leave. Furthermore, since f is integer-valued, for each $u \in L$, the one unit of flow can enter on at most one edge and can leave on at most one edge. Thus, one unit of flow enters u if and only if there is exactly one vertex $v \in R$ such that $f(u, v) = 1$, and at most one edge leaving each $u \in L$ carries positive flow. A symmetric argument applies to each $v \in R$. The set M is therefore a matching.

In order to see that $|M| = |f|$, observe that for every matched vertex $u \in L$, we have $f(s, u) = 1$, and for every edge $(u, v) \in E - M$, we have $f(u, v) = 0$. Consequently, $f(L \cup \{s\}, R \cup \{t\})$, the net flow across cut $(L \cup \{s\}, R \cup \{t\})$ is equal to $|M|$. Now, we can apply the Lemma 26.4 and we get $|f| = f(L \cup \{s\}, R \cup \{t\}) = |M|$. □

Based on Lemma 26.9, we would like to conclude that a maximum matching in a bipartite graph G corresponds to a maximum flow in its corresponding flow network G', and we can therefore compute a maximum matching in G by running a maximum-flow algorithm on G. The only hitch in this reasoning is that the maximum-flow algorithm might return a flow in G for which some $f(u, v)$ is not an integer, even though the flow value $|f|$ must be an integer. The following theorem shows that if we use the Ford-Fulkerson method, this difficulty cannot happen.

$c(u,v) = 1$

Theorem 1 (Integrality theorem)**.** *If the capacity function c takes on only integral values, then the maximum flow f produced by the Ford-Fulkerson method has the property that $|f|$ is an integer. Moreover, for all vertices u and v, the value of $f(u, v)$ is an integer.*



Proof. The proof is by induction on the number of iterations of the while loop in the Ford-Fulkerson algorithm.

The for loop sets $(u, v).f = 0$. At the end of the first pass of the while loop, the residual capacity c_f is integral since edge weights are integral. Since $(u, v).f = 0$ and c_f is integral, line 7 or 8 of the algorithm set $(u, v).f$ in either case to an integer value. Assume now that at the n^{th} iteration, $(u, v).f$ is integral. At the $(n + 1)^{th}$ iteration c_f is minimal and, by induction hypothesis, it is integral. The lines 7 or 8 of the algorithm set again the $(u, v).f$ to an integer as in both cases the summands are integral. Therefore, after the $(n + 1)^{th}$ iteration $(u, v).f$ is integral for all $(u, v) \in E$. The flow f is a sum over integral edges and so it is integral. Consequently, the maximum flow $|f|$ is integral. □

Next, we prove the following corollary to Lemma 26.9 on our way to show that **Ford-Fulkerson** algorithm solves this problem correctly.

Corollary 1 (Corollary 26.11)**.** *The cardinality of a maximum matching M in a bipartite graph G equals the value of a maximum flow f in its corresponding flow network G'.*

Proof. We use the same terminology as in Lemma 26.9. Suppose that M is a maximum matching in G and that the corresponding flow f in G' is not maximum. Then there is a maximum flow f' in G' such that $|f'| > |f|$. Since the capacities in G' are integer-valued, by Theorem above, we can assume that f' is integer-valued. Thus, f' corresponds to a matching M' in G with cardinality $|M'| = |f'| > |f| = |M|$, contradicting our assumption that M is a maximum matching. In a similar manner, we can show that if f is a maximum flow in G', its corresponding matching is a maximum matching on G. □

Therefore, given a bipartite undirected graph G, we can find a maximum matching by creating the flow network G' , running the Ford-Fulkerson algorithm, and directly obtaining a maximum matching M from the integer-valued maximum flow f found.

Since any matching in a bipartite graph has cardinality at most $min(L, R) = O(V)$, the value of the maximum flow in G' is $O(V)$. We can therefore find a maximum matching in a bipartite graph in time $O(VE') = O(VE)$, since $|E'| = \Theta(E)$.

$V \cdot ?$

$V = L \cup R$

$E' = 3E$