Computer Science C73
Scarborough Campus

Fall 2023
University of Toronto

Maximum matchings, minimum vertex covers, and perfect matchings in bipartite graphs

Vassos Hadzilacos

1 Maximum matchings in bipartite graphs

A ***bipartite graph*** is an undirected graph $G = (V, E)$ whose set of nodes can be partitioned into two sets X and Y (i.e., $X \cup Y = V$ and $X \cap Y = \varnothing$), so that every edge of G connects a node in X to a node in Y (i.e., for each $\{u, v\} \in E$, $u \in X$ if and only if $v \in Y$). An example of a bipartite graph is shown in Figure 1(a) below.

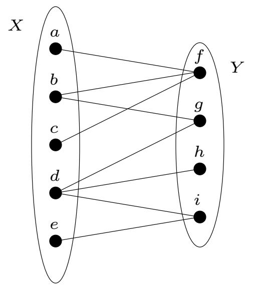

Figure 1(a)

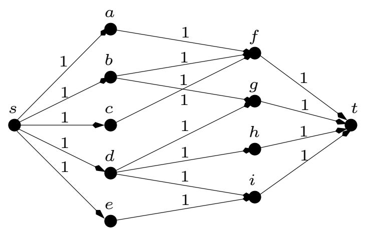

Figure 1(b)

We will assume that when we are given a bipartite graph, we are also given a partition of the set of nodes into X and Y so that the edges of G connect nodes of X to nodes of Y, and we will denote this graph as $G = (\{X, Y\}, E)$. The following facts about bipartite graphs can be proved using techniques you learned in CSCB63 and are left as exercises:

Fact 1 *Let $G = (V, E)$ be an undirected graph.*
(a) There is an algorithm that determines in time $O(|V| + |E|)$ whether G is bipartite and, if so, partitions V into two sets X and Y so that every edge in E connects a node in X to a node in Y.
(b) G is bipartite if and only if it has no cycle of odd length.

A ***matching*** of an undirected graph $G = (V, E)$ is a subset of the edges $M \subseteq E$ so that no two edges in M share an endpoint; i.e., for all $e, e' \in M$, if $e \neq e'$ then $e \cap e' = \varnothing$. The empty set is obviously a matching. A matching of maximum cardinality is called a ***maximum matching*** of the graph.

We can find a maximum matching of a bipartite graph by reducing the problem to finding a maximum flow of a network. To do so we convert the bipartite graph to the flow network defined below.

Definition 2 *Let $G = (\{X, Y\}, E)$ be a bipartite graph. The **flow network** $\mathcal{F} = (G', s, t, c)$ **that corresponds to** G is defined as follows:*
- *s and t are new nodes not in $X \cup Y$;*
- *$G' = (V, E')$ is the directed graph with set of nodes $V = X \cup Y \cup \{s, t\}$ and set of edges $E' = \{(s, x) :\ x \in X\} \cup \{(x, y) :\ x \in X,\ y \in Y, \text{ and } \{x, y\} \in E\} \cup \{(y, t) :\ y \in Y\}$; and*
- *$c(e) = 1$ for each $e \in E'$.*

Figure 1(b) shows the flow network that corresponds to the bipartite graph in Figure 1(a).

The idea now is to find an integral maximum flow in $\mathcal{F}$ (for example, using the Ford-Fulkerson algorithm); this flow will put one unit of traffic on some of the edges that go from X to Y in $\mathcal{F}$. The set of these edges turns out to be a maximum matching of G. So the algorithm is as follows:

```
BipartiteMaxMatching($G$)
▷ $G = (\{X, Y\}, E)$ is a bipartite graph
1  $\mathcal{F}$ := the flow network that corresponds to $G$
2  $f$ := FF-MaxFlow($\mathcal{F}$)
3  $M := \big\{\{x, y\} :\ x \in X,\ y \in Y, \text{ and } f(x, y) = 1\big\}$
4  return $M$
```

Figure 2(a) shows a maximum flow in the flow network that corresponds to the bipartite graph of Figure 1(a), where heavy edges have traffic 1 and the other edges have traffic 0. Figure 2(b) shows the corresponding maximum matching of the bipartite graph.

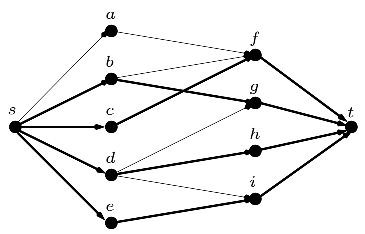

Figure 2(a)

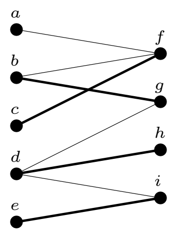

Figure 2(b)

Let n be the number of nodes and m be the number of edges of G. Line 1 takes $O(m + n)$ time; line 2 takes $O(mn)$ time (since the sum of the capacities of the edges out of s is at most n); line 3 takes $O(m)$ time, and line 4 takes $O(n)$ time. So the running time of this algorithm is $O(mn)$.

We now prove that the algorithm is correct; i.e., it returns a maximum matching of G. This follows from the following fact, which establishes a close relationship between integral flows in $\mathcal{F}$ and matchings in G:

Fact 3 *Let $G = (\{X, Y\}, E)$ be a bipartite graph and $\mathcal{F}$ be the flow network that corresponds to G.*

(a) For any integral flow f of $\mathcal{F}$, the set

$$M_f = \big\{\{x, y\} :\ x \in X,\ y \in Y, \text{ and } f(x, y) = 1\big\}$$

is a matching of G with size $\mathcal{V}(f)$.

(b) For any matching M of G, the function $f_M : E' \to \{0, 1\}$ defined by

$$f_M(e) = \begin{cases} 1, & \text{if there is a path } s, x, y, t \text{ in } \mathcal{F} \text{ that contains } e \text{ and } \{x, y\} \in M \\ 0, & \text{otherwise} \end{cases}$$

is a flow of $\mathcal{F}$ with value $|M|$.

Proof. (a) To show that M_f is a matching of G, consider any $x \in X$. We claim that there cannot be edges $\{x, y\}$ and $\{x, y'\}$ in M_f, for $y \neq y'$. Suppose, for contradiction, that we such edges exist. Then, by the definition of M_f, $f(x, y) = f(x, y') = 1$. So the outflow of f at x is at least two, while the inflow is at most one (there is only one edge into x and it has capacity one). Therefore f violates the conservation constraint at x, contradicting that f is a flow. By a similar argument we can show that no $y \in Y$ can be the endpoint of two edges in M_f. Therefore M_f is indeed a matching. Finally, by considering the (S,T) cut of $\mathcal{F}$ where $S = \{s\} \cup X$ and $T = \{t\} \cup Y$ and applying the flow theorem to that cut we have:

$$\mathcal{V}(f) = \sum_{out(S)\cap in(T)} f(e) - \underbrace{\sum_{out(T)\cap in(S)} f(e)}_{\text{sum=0: no such edges!}} = |\{(x,y):\ x \in X,\ y \in Y, \text{ and } f(x,y) = 1\}| = |M_f|.$$

(b) It is obvious that f_M satisfies the capacity constraint. For the conservation constraint, consider any node $x \in X$. If $f_M(s,x) = 0$ then there is no $s \to t$ path s,x,y,t such that $\{x,y\} \in M$, and so $f_M(x,y) = 0$ for all edges (x,y) in $\mathcal{F}$; so conservation holds at x in this case. If $f_M(s,x) = 1$ then there is an $s \to t$ path s,x,y,t such that $\{x,y\} \in M$; and there is only one such path (otherwise M would contain two edges with x as one endpoint, contradicting that it is a matching). Thus $\sum_{e \in out(x)} f_M(e) = 1$, and again conservation holds at x. By a similar argument we can show that conservation holds at any $y \in Y$. Therefore f_M is a flow in $\mathcal{F}$. By considering the (S,T) cut of $\mathcal{F}$ as in the proof of part (a) we have:

$$\mathcal{V}(f_M) = |\{(x,y):\ x \in X,\ y \in Y, \text{ and } f_M(x,y) = 1\}| = |M|. \quad \square$$

To see why this fact implies the correctness of algorithm BipartiteMaxMatching we first note that, by Fact 3(a), the set M returned by the algorithm is a matching of G whose size is the value of the maximum flow f. By Fact 3(b) M must be a maximum matching: If there was a matching M' larger than M, by part (b), there would be a flow f' of $\mathcal{F}$ whose value is $|M'| > |M| = \mathcal{V}(f)$ — contradicting that f is a maximum flow of $\mathcal{F}$.

2 Minimum vertex covers in bipartite graphs

A ***vertex cover*** of an undirected graph $G = (V,E)$ is a subset of the nodes $R \subseteq V$ so that each edge of G has at least one endpoint in R; i.e., for each $\{u,v\} \in E$, $u \in R$ or $v \in R$ (or both). The set of all nodes is obviously a vertex cover. A vertex cover of minimum cardinality is called a ***minimum vertex cover*** of the graph. Finding a minimum vertex cover is an important optimization problem. Unfortunately, it is NP-hard, which means that it is unlikely to be solvable in polynomial time. Luckily, polynomial-time solutions for this problem exist for special types of graphs, including bipartite graphs, as we will see.

The following lemma states that the cardinality of any matching does not exceed that cardinality of any vertex cover.

Lemma 4 *For any undirected graph $G = (V,E)$, any matching M, and any vertex cover R of G, $|M| \leq |R|$.*

Proof. Consider any function $\phi : M \to R$ that maps each edge in M to one of its endpoints that is in R (i.e., for each $\{u,v\} \in M$, $\phi(\{u,v\}) = u$ or $\phi(\{u,v\}) = v$ and $\phi(\{u,v\} \in R)$. Such a function exists because R is a vertex cover, and so its elements "touch" every edge of G and in particular the edges in the matching M. Since M is a matching, ϕ is one-to-one. Thus, by the pigeonhole principle, $|M| \leq |R|$. $\square$

Note the similarity of the form of this lemma to the lemma stating that the value of any flow is at most the capacity of any cut. This is not a coincidence; both of these facts are instances of a more general phenomenon in linear programming (the subject of the next course unit), known as "weak duality".

Lemma 4 is true about all (undirected) graphs. For bipartite graphs, however, the inequality is actually an equality! This fact is known as König's Theorem, and its proof below also yields an efficient algorithm to find a minimum vertex cover of a bipartite graph. Note that equality does not necessarily hold for non-bipartite graphs. For example, a triangle (a graph with three nodes, every two of which are connected by an edge) has maximum matching size one, but minimum vertex cover size two.

Theorem 5 (König's Theorem) *In any bipartite graph, the cardinality of a maximum matching is equal to the cardinality of a minimum vertex cover.*

PROOF. Let $G = (\{X, Y\}, E)$ be a bipartite graph, $\mathcal{F}$ be the corresponding flow network (Definition 2), and f be an integral maximum flow in $\mathcal{F}$. Consider now the residual graph $\mathcal{F}_f$ of the maximum flow f. Since f is an integral flow and the capacity of every edge in $\mathcal{F}$ is 1,

$$f(u,v) = 0 \text{ if } (u,v) \text{ is a forward edge of } \mathcal{F}_f, \text{ and } f(u,v) = 1 \text{ if } (v,u) \text{ is a backward edge of } \mathcal{F}_f. \quad (1)$$

Let S be the set of nodes reachable from S in $\mathcal{F}_f$, and T be the set of the remaining nodes. As we saw in the proof of correctness of the Ford-Fulkerson algorithm, (S, T) is a minimum (s,t)-cut of the flow network, and its capacity is equal to the value of f.

Figure 3(a) below shows the residual graph of the maximum flow in Figure 2(a), along with the partition of the set of nodes into those that are reachable from s in the residual graph (nodes in the green "cloud") and the rest of the nodes (in the red "cloud"). Figure 3(b) shows the minimum cut of the flow network in Figure 1(b) that is derived from this partition of the nodes of the residual graph into the "green" and "red" clouds. Next we show that there is no edge in the flow network from a "green" node in X to a "red" node in Y.

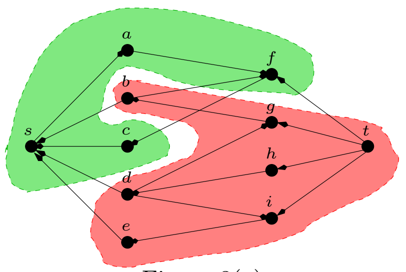

Figure 3(a)

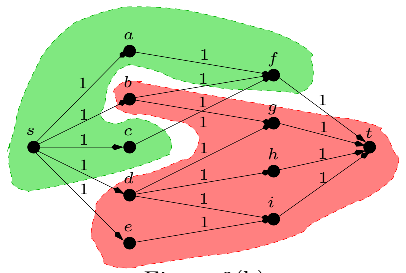

Figure 3(b)

Claim 5.1 *There is no edge (x,y) in $\mathcal{F}$ such that $x \in (X \cap S)$ and $y \in (Y \cap T)$.*

PROOF OF CLAIM 5.1. Suppose, for contradiction, that there exist $x \in X \cap S$ and $y \in Y \cap T$ such that (x,y) is an edge in the flow network $\mathcal{F}$. Then (y,x) is a backward edge in $\mathcal{F}_f$ (otherwise (x,y) would be a forward edge and since x is reachable from s so would y, meaning that $y \in S$ and contradicting that $y \in T$). Thus, by (1),

$$f(x,y) = 1. \quad (2)$$

Since $x \in S$, there is an $s \to x$ path p in the residual graph $\mathcal{F}_f$. There are two cases, both leading to contradiction.

CASE 1. $p = s, x$. So (s,x) is a forward edge in $\mathcal{F}_f$, and by (1) $f(s,x) = 0$. In view of (2), f violates the conservation property at x: the inflow is 0, but the outflow is at least 1.

CASE 2. $p = s, x_1, y_1, x_2, y_2, \ldots, x_k, y_k, x$, where $x_1, \ldots, x_k \in X$ and $y_1, \ldots y_k \in Y$. All these nodes are in S (since they are reachable from s in $\mathcal{F}_f$), so $y_k \neq y$. The edge (y_k, x) is a backward edge in $\mathcal{F}_f$, so by (1) $f(x, y_k) = 1$. But then, in view of (2) and the fact that $y_k \neq y$, f violates the conservation property at x: the inflow is at most 1 but the outflow is at least 2. $\square$ Claim 5.1

Claim 5.1 implies that

$$\text{the set } R = (X \cap T) \cup (Y \cap S) \text{ is a vertex cover of the bipartite graph } G.^{1} \quad (3)$$

[1] In terms of Figure 3, these are the "red" nodes of X (i.e., $\{b, d, e\}$) and the "green" nodes of Y (i.e., $\{f\}$).

This is because $S \cup T \supseteq X \cup Y$, and so every edge that connects a node in X to a node in Y in G is covered by some node in R, except for edges that connect a node in $X \cap S$ to a node in $Y \cap T$ — but, by the Claim, no such edge exists!

Claim 5.1 also implies that the only edges in the flow network $\mathcal{F}$ that enter T from S are edges of the form (s, x) such that $x \in X \cap T$, and edges of the form (y, t) such that $y \in Y \cap S$. (The only other possibility are edges (x, y) with $x \in X \cap S$ and $y \in Y \cap T$, but, by the Claim, no such edge exists.) Since all edges have capacity 1,

$$c(S, T) = |(X \cap T) \cup (Y \cap S)| = |R|. \tag{4}$$

By Fact 3 (see Section 1),

$$M = \big\{\{x, y\} : \; x \in X,\, y \in Y, \text{ and } f(x, y) = 1\big\} \text{ is a maximum matching of } G, \text{ and } |M| = \mathcal{V}(f). \tag{5}$$

By the max-flow-min-cut theorem, $\mathcal{V}(f) = c(S, T)$. Together with (4) and 5, this implies that $|M| = |R|$. So we have a matching M whose size is equal to the size of vertex cover R, and so by Lemma 4, R is a minimum vertex cover. □

The preceding proof of König's Theorem gives rise to the following algorithm for finding a minimum vertex cover of a bipartite graph.

```
  BipartiteMinVC($G$)
  ▷ $G = (\{X, Y\}, E)$ is a bipartite graph
1 $\mathcal{F}$ := the flow network that corresponds to $G$
2 $f$ := FF-MaxFlow($\mathcal{F}$)
3 $\mathcal{F}_f$ := the residual graph of flow $f$ with respect to network $\mathcal{F}$
4 $S := \{u : \; u$ is a node reachable from $s$ in $\mathcal{F}_f\}$
5 $T := \{u : \; u$ is a node in $\mathcal{F}_f$ not in $S\}$
6 return $(X \cap T) \cup (Y \cap S)$
```

Let n be the number of nodes and m be the number of edges of G. Line 1 takes $O(m + n)$ time; line 2 takes $O(mn)$ time; line 3 takes $O(m + n)$ time; line 4 takes $O(m + n)$ time (say, using depth-first search); and lines 5 and 6 take $O(n)$ time. Thus, we can find the minimum vertex cover of a bipartite graph in $O(mn)$ time.

As we saw in Sections 1 and 2, maximum matchings and minimum vertex covers are closely related for bipartite graphs. Both can be found efficiently (in polynomial time) by applying maximum flow techniques. This close relationship does not extend to all graphs, however: Maximum matchings in general graphs can be found in polynomial time, using algorithms that are beyond the scope of this course. In contrast, the minimum vertex cover problem for general graphs is NP-hard, and is therefore unlikely to be solvable in polynomial time.

3 Hall's Theorem

A matching M of an undirected graph $G = (V, E)$ is ***perfect*** if it does not leave any node unmatched; i.e., for every $u \in V$ there is some $v \in V$ such that $\{u, v\} \in M$. A perfect matching is obviously a maximum matching, but not every graph has a perfect matching — for example, a graph with an odd number of nodes cannot have a perfect matching, but of course it has a maximum matching.

We now turn our attention again specifically to bipartite graphs. When does a bipartite graph $G = (\{X, Y\}, E)$ have a perfect matching? Clearly, we must have $|X| = |Y|$: otherwise every matching must leave unmatched some node in X or Y (whichever is larger). This, however, is not enough. For example, the graph in Figure 4 does not have a perfect matching even though each of X and Y has three nodes: Node d can only be matched with a or b; but if it is matched with one, the other cannot be matched with any other node.

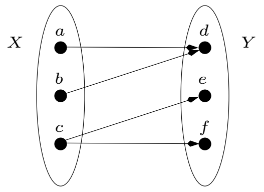

Figure 4

This example can be generalized: Let the ***neighbours*** of a node u be the set of nodes $N(u)$ that are adjacent to it; i.e., $N(u) = \{v : \{u, v\} \in E\}$. The neigbours $N(U)$ of a set of nodes U is the union of the neighbours of the nodes in U; i.e., $N(U) = \cup_{u \in U} N(u)$. For G to have a perfect matching, there must be no $X' \subseteq X$ such that $|X'| > |N(X')|$: if such a set exists, then every matching must leave some node in X' unmatched.

It turns out that this is not only a necessary, but also sufficient, condition for a bipartite graph with $|X| = |Y|$ to have a perfect matching. This fact is known as Hall's Theorem, and one way of proving it follows from our discussion of minimum vertex covers in bipartite graphs.

Theorem 6 (Hall's Theorem) *A bipartite graph $G = (\{X, Y\}, E)$ such that $|X| = |Y|$ has a perfect matching if and only if there is no $X' \subseteq X$ such that $|X'| > |N(X')|$.*

PROOF. [ONLY IF.] Suppose, for contradiction, that G has a perfect matching M but there is some $X' \subseteq X$ such that $|X'| > |N(X')|$. Let $\phi : X' \to N(X')$ be a function that maps $x \in X'$ to the neighbour with which it is matched in M; i.e., for every $x \in X'$, $\{x, \phi(x)\} \in M$. Such a function ϕ exists because M is a perfect matching. Since $|X'| > |N(X')|$, by the pigeonhole principle, the function is not one-to-one; this contradicts that M is a matching.

[IF.] We prove the contrapositive: If G has no perfect matching then there is some $X' \subseteq X$ such that $|X'| > |N(X')|$.

So, suppose G has no perfect matching and consider the flow network $\mathcal{F}$ that corresponds to the bipartite graph G. Let f be an integral maximum flow of $\mathcal{F}$, $\mathcal{F}_f$ be the residual graph of f with respect to $\mathcal{F}$, S be the set of nodes of $\mathcal{F}_f$ reachable from s, and T be the set of remaining nodes of $\mathcal{F}_f$. Partition each of X and Y into its S-part and its T-part: i.e., $X' = X \cap S$, $X'' = X \cap T$, $Y' = Y \cap S$, and $Y'' = Y \cap T$. We will prove that X' is the set we are looking for, i.e., a subset of X that is larger than the set of its neighbours.

In the proof of Theorem 5 we saw that $X'' \cup Y'$ is a minimum vertex cover of G and its size is equal to the size of a maximum matching M of G, i.e., $|X''| + |Y'| = |M|$. Since $|X| = |Y|$, a perfect matching of G would have $|X|$ edges. Since G has no perfect matching (by assumption), $|M| < |X| = |X'| + |X''|$. So, $|X'| + |X''| > |X''| + |Y'|$, and therefore $|X'| > |Y'|$.

In the proof of Theorem 5 we also saw that there are no edges from X' to Y'' (see Claim 5.1), and so $N(X') \subseteq Y'$; therefore $|Y'| \geq |N(X')|$. Combining this with the fact, proven in the previous paragraph, that $|X'| > |Y'|$, we conclude that $|X'| > |N(X')|$, as wanted. $\square$