Maximum Matching and Minimum Vertex Cover of Bipartite Graphs

We design an algorithm to find one maximum matching M of a given bipartite graph $G = (X \cup Y, E)$. This algorithm will also give a vertex cover V_1 of G, and we will prove $|M| = |V_1|$. This will therefore proves that M is a maximum matching, V_1 is a minimum vertex cover, and that for every bipartite graph the size of its maximum matching and minimum vertex cover are equal.

We will not design new algorithm for this bipartite maximum matching problem, although this has been a classical problem in computer science and many algorithms exist (check wikipedia). To solve it, we will *transform this problem into a maximum-flow problem*. This typically consists of three steps:

1. construct an input for the maximum-flow problem, i.e., a network, based on the given input, i.e., a bipartite graph;
2. obtain the maximum flow and/or the minimum s-t of the constructed network, using any existing algorithm such as Ford-Fulkson algorithm;
3. construct the maximum matching of the given bipartite graph based on the optimal solution in step 2.

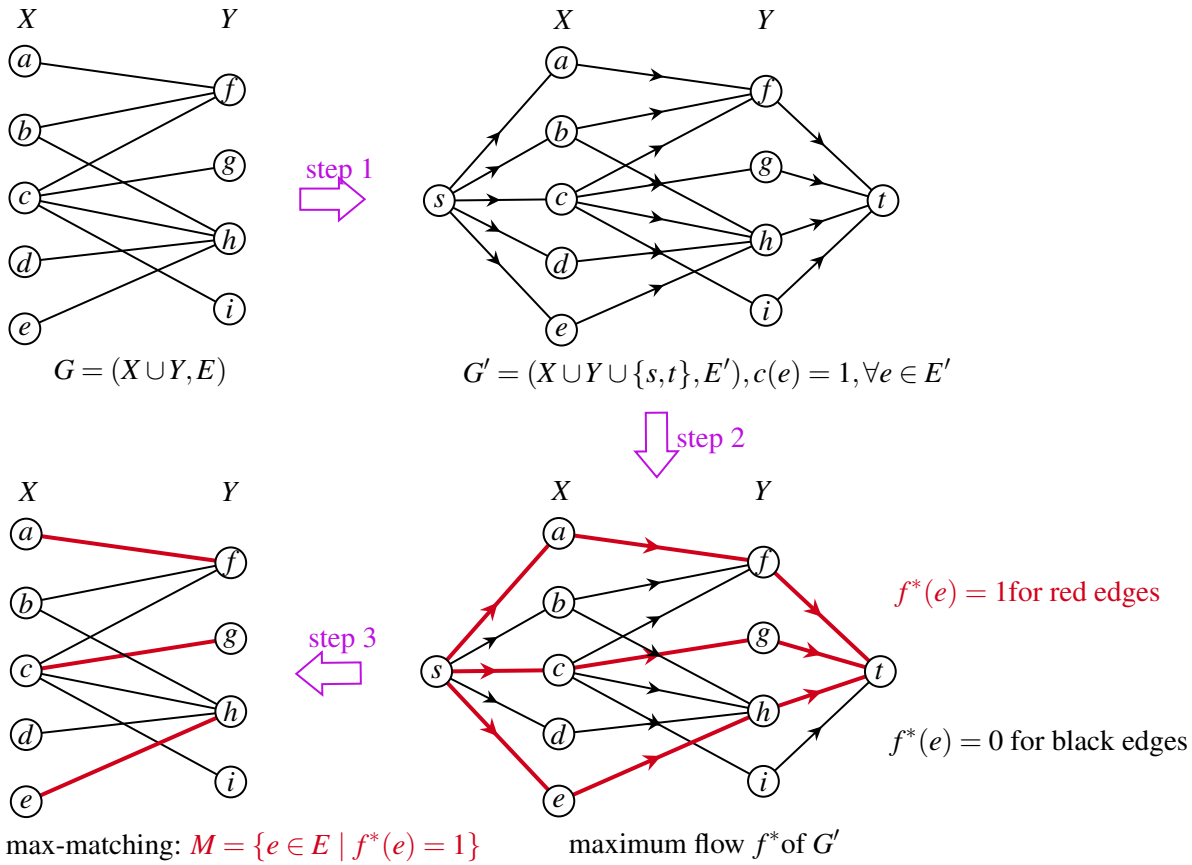


Figure 1: Solving the maximum-matching problem of bipartite graph using network flow. In the upper right network G', every edge e has a capacity of $c(e) = 1$. In the lower right figure which shows the maximum flow f^*, red edges indicate $f^*(e) = 1$. The maximum-matching of the bipartite graph G is given by $M = \{e \in E \mid f^*(e) = 1\}$.

Step 2 can be done simply by calling an existing solver. Steps 1 and 3 are closely related. How to construct the network is another "art" part of algorithm design: usually it requires trying and observing.

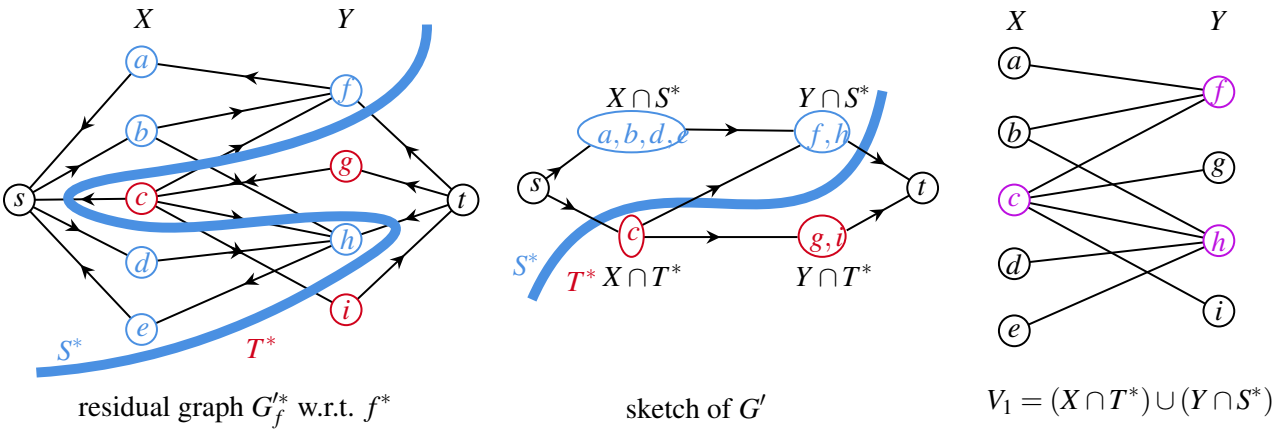
Here is a working approach. Given a bipartite graph $G = (X \cup Y, E)$, we construct a network $G' = (V', E')$ with source s sink t and edge capacities c as follows. See Figure 1. We construct G' by modifying G: not a surprise, the given bipartite graph is the main body of G'. We first add two new vertices, a source s and a sink t, a step that is usually required in transforming a problem into a maximum-flow problem. That is, $V' = X \cup Y \cup \{s,t\}$. Second, we add a new directed edge (s,x) that connects the source to every vertex $x \in X$, and add a new directed edge (y,t) that connects every vertex $y \in Y$ to the sink t. For every edge $(x,y) \in E$ with $x \in X$ and $y \in Y$ in the bipartite graph, we then copy it to G' and assign it with a direction pointing from x to y. Last, we set the capacity as 1 for every edge in G', i.e., $c(e) = 1$ for every $e \in E'$.

In step 2, we use any max-flow algorithm to find a maximum flow, denoted as f^*, of the constructed network G'. As all capacities of the network are integers, we can therefore assume that $f^*(e)$ is integer for any $e \in E'$. In particular, since all capacities are 1s, we know that $f^*(e) \in \{0,1\}$. Therefore, edges in the constructed network with $f^*(e) = 1$ forms a set of *edge-disjoint paths* from s to t, and the number of such paths is $|f^*|$, as each such path carries 1 unit of flow. We collect those edges in the bipartite graph with $f^*(e) = 1$ as the matching of the bipartite graph (although we have not proved them a matching yet). Formally, step 3 finds: $M := \{e \in E \mid f^*(e) = 1\}$.

We now prove that the identified edge-set M is a maximum matching of the bipartite graph. First, we show M is a matching, i.e., no two edges in M share vertex. Suppose conversely that there exist two edges $(u,v),(w,v) \in M$ that share v. Then, by the construction of M, we know that $f^*(u,v) = f^(w,v) = 1$. Therefore, in the constructed network G', the in-flow of v is at least $f^*(u,v) + f^*(w,v) = 2$, so its out-flow must be at least 2. This constracts to that the *out-capacity* of v is 1. From here we can also see why we set the capacities of the edges adjacent to s or t as 1: this guarantees that each vertex is matched at most once.

We then show that M is a maximum matching of the bipartite graph G. First, notice that $|M| = |f^*|$. This is because $|f^*|$ equals to the number of above edge-disjoint s-t paths, and we add the middle edge of each path to M. Therefore, if a matching M' with more edges than M exists, i.e., $|M'| > |M| = |f^*|$, we can use M' to build $|M'|$ such edge-disjoint s-t paths and thus give a flow with value of $|M'|$, which contradicts to the fact that f^* is a maximum flow.

We continue above algorithm to find a (minimum) vertex cover of the bipartite graph G. The code, like



residual graph G'^*_f w.r.t. f^* sketch of G'

Figure 2: Constructing the minimum vertex cover using the residual graph w.r.t. the maximum flow.

the minimum s-t cut, again hides in the residual graph G'_{f^*} w.r.t. the maximum flow f^*. See Figure 2. Let (S^*, T^*) be the minimum s-t cut found using G'_{f^*}. Recall that $S^* := \{v \in V' \mid s \text{ can reach } v \text{ in } G'_{f^*}\}$, and $T^* = V' \setminus S^*$. This s-t cut partitions all vertices in the bipartite graph into 4 categories: $X \cap S^*$, $X \cap T^*$, $Y \cap S^*$, and $Y \cap T^*$. See the sketch of the network in the middle of Figure 2. We collect two of them: we define $V_1 := (X \cap T^*) \cup (Y \cap S^*)$ and we will prove that V_1 is a minimum vertex cover of the bipartite graph.

We first show that V_1 is a vertex cover of the bipartite graph. This is equivalent to show the following fact. Can you spot it in Figure 2?

Fact 1. In the bipartite graph, there does not exist any edge between $(X \cap S^*)$ and $(Y \cap T^*)$.

Proof. Suppose conversely that there exists an edge $(u,v) \in E$ with $u \in X \cap S^*$ and $v \in Y \cap T^*$. Consider $f^*(u,v)$, i.e., the amount of flow carried by this edge in the maximum flow f^*. There are two possibilities (as all capacities are 1): either $f^*(u,v) = 0$ or $f^*(u,v) = 1$. See Figure 3.

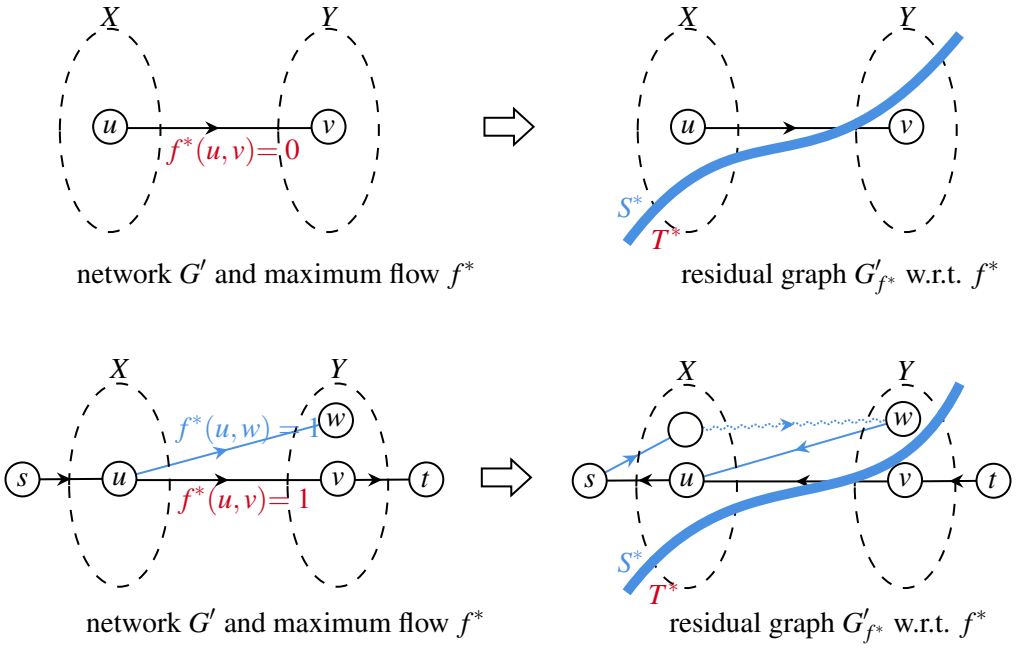


Figure 3: Illustrating the two cases in proving Fact 1.

1. Suppose that $f^*(u,v) = 0$. Then in the residual graph G'_{f^*} this edge becomes a forward edge (u,v). Then v must be in S^* as s can reach u and u can reach v following this edge, a contradiction to the assumption that $v \in T^*$.

2. Suppose that $f^*(u,v) = 1$. Then we must have that $f^*(s,u) = f^*(v,t) = 1$, in order to make u and v satisfy the conservation condition. Therefore, in the residual graph G'_{f^*}, all these three edges become backward edges. Think: in the residual graph why u is reachable from s? Clearly u cannot be reached from s using edge (s,u) as it's edge (u,s) that appears in the residual graph. Hence, there must exist a path from s to u using other intermediate vertices. Assume that on this path the vertex before u is w, i.e., edge (w,u) appears in the residual graph. Notice that w cannot be v, as otherwise this implies that s can reach v which contradicts to that $v \in T^*$. Since edge (w,u) is also a backward-edge, we have that $f^*(u,w) = 1$. Combined, we have a contradiction, as it's not possible to have both $f^*(u,v) = 1$ and $f^*(u,w) = 1$ because this breaks the capacity condition of edge (s,u). □

Above fact immediately gives that V_1 is a vertex cover of the bipartite graph. We now show that V_1 is also a minimum vertex cover. Consider the capacity of s-t cut (S^*, T^*). See Figure 2 for a sketch of G'. Since there is no edge between $X \cap S^*$ and $Y \cap T^*$, we have that $c(S^*, T^*) = |X \cap T^*| + |Y \cap S^*|$, as every edge has a capacity of 1. According to the max-flow min-cut theorem, we know $c(S^*, T^*) = |f^*|$. We also proved that $|M| = |f^*|$. Combined, we have $|V_1| = |X \cap T^*| + |Y \cap S^*| = c(S^*, T^*) = |f^*| = |M|$, as desired.

We summarize the maximum matching problem and the minimum vertex cover problem below.

1. For any *bipartite graph*, its maximum matching and minimum vertex cover always have equal cardinality, and such maximum matching and minimum vertex cover can be found using above algorithm which runs in polynomial-time.

2. For *general* undirected graphs, the size of any vertex cover is always an upper bound of that of any matching, but a gap might exist on graphs with odd-length cycles. The minimum vertex cover problem (on general undirected graphs) is NP-hard, i.e., there does not exist polynomial-time (optimal) algorithm for it. But the maximum matching problem (on general undirected graphs) can be solved in polynomial-time, for example, with the Blossom algorithm (Edmonds, 1961).