Covers and independent sets

Lecture 4 – Graph Theory 2016 – EPFL – Frank de Zeeuw

---

1 Summary of matchings in arbitrary graphs

In the previous lecture we treated matchings in bipartite graphs. As mentioned there, in this course we will not cover matchings in arbitrary graphs, mostly because it would take too much time. Nevertheless, we now give a quick overview of the basic facts about matchings in arbitrary graphs.

In an arbitrary graph, it is still true that a matching is maximum if and only if there is no augmenting path for it (the proof that we gave did not use the bipartiteness). However, the augmenting path algorithm that we described does not work, mainly because it is not clear how to find an augmenting path or determine that none exists.

Both Hall's Theorem and König's Theorem fail for arbitrary graphs. Take for instance the triangle K_3. It is 2-regular, but does not have a perfect matching. The maximum size of a matching is 1, but the minimum size of a vertex cover is 2.

There is an analogue of Hall's Theorem for arbitary graphs, known as Tutte's Theorem, but it is more complicated. There is also a good algorithm due to Edmonds for finding a maximum matching in an arbitrary graph, known as the "blossom algorithm". It does use augmenting paths, but its method for finding them is considerably more complicated.

2 Matchings, covers, and independent sets

In this section we discuss several objects in graphs related to matchings and covers.

Definition. *Let G be a graph (that is not necessarily bipartite).*
- A matching *is a set $M \subset E(G)$ such that the edges in M are pairwise disjoint;*
- A vertex cover *is a set $C \subset V(G)$ such that every edge of G is incident to a vertex of C;*
- An independent set *is a set $I \subset V(G)$ such that no two vertices in I are adjacent;*
- An edge cover *is a set $C \subset E(G)$ such that every vertex of G is incident to an edge in C (note that a graph with an isolated vertex has no edge cover).*

Just like for matchings and vertex covers, we would like to find maximum independent sets and minimum edge covers. We introduce the following parameters to keep track of all these extrema.

$$\begin{aligned} \mathrm{m}(G) &= \max\{|M| : M \subset E(G) \text{ is a matching}\} \\ \mathrm{vc}(G) &= \min\{|C| : C \subset V(G) \text{ is a vertex cover}\} \\ \alpha(G) &= \max\{|S| : S \subset V(G) \text{ is independent}\} \\ \mathrm{ec}(G) &= \min\{|C| : C \subset E(G) \text{ is an edge cover}\} \end{aligned}$$

We now prove various relations between these parameters.

Lemma 2.1. *For any graph G we have* $\mathrm{m}(G) \leq \mathrm{vc}(G)$ *and* $\alpha(G) \leq \mathrm{ec}(G)$.

Proof. For the first inequality, take a matching M, and observe that any vertex cover has to have at least one vertex from each edge in M. Thus the size (number of vertices) of any vertex cover is at least the size (number of edges) of any matching, so the minimum size of a vertex cover is at least the maximum size of a matching. (This is the same argument as we saw for bipartite graphs, but there we had equality, which is not always the case here.)

For the second inequality, take an independent set I, and observe that any edge cover has to have at least one edge incident with each vertex of I, and no edge can cover two vertices of I. Thus the size of any edge cover is at least the size of any independent sets, and the minimum size of an edge cover is at least the maximum size of an independent set. □

Lemma 2.2. *For any graph G we have* $\alpha(G) + \mathrm{vc}(G) = |V(G)|$.

Proof. Observe that I is independent if and only if $V(G)\backslash I$ is a vertex cover. This is because I is independent if and only if every edge of G has at least one vertex in $V(G)\backslash I$, which means exactly that $V(G)\backslash I$ is a vertex cover. Thus, if I is an independent set of size $\alpha(G)$, then there is a vertex cover of size $|V(G)| - |I| = |V(G)| - \alpha(G)$, which implies that $\mathrm{vc}(G) \leq |V(G)| - \alpha(G)$. Similarly, if C is a vertex cover of size $\mathrm{vc}(G)$, then there is an independent set of size $|V(G)| - |C| = |V(G)| - \mathrm{vc}(G)$, so $\alpha(G) \geq |V(G)| - \mathrm{vc}(G)$. Combining the two obtained inequalities gives the equality in the lemma. □

Lemma 2.3. *For a graph G without isolated vertices we have* $\mathrm{m}(G) + \mathrm{ec}(G) = |V(G)|$.

Proof. Let M be a matching with $|M| = \mathrm{m}(G)$. Each of the $|V(G)| - 2|M|$ unmatched vertices must be incident with some edge, since otherwise the vertex would be isolated; one endpoint of this edge must be in an edge of M, since otherwise the edge could be added to M to give a larger matching. Pick one such edge for each vertex not in an edge of M, and let C be the union of M and these $|V(G)| - 2|M|$ edges. Then C is an edge cover of size $|M| + (|V(G)| - 2|M|) = |V(G)| - \mathrm{m}(G)$, which implies that $\mathrm{ec}(G) \leq |V(G)| - \mathrm{m}(G)$.

Let C be a minimum edge cover, so we have $|C| = \mathrm{ec}(G)$. Let H be the graph with $V(H) = V(G)$ and $E(H) = C$. An edge $e \in C$ shares at most one vertex with any other edge of C, since otherwise we could remove it and obtain a smaller edge cover. Thus every edge in H has at least one endpoint with degree one. This means that H is a forest whose connected components are all stars (a *star* is a graph $K_{1,t}$, consisting of one vertex of degree t connected to t vertices of degree one). The number of stars in H is $|V(G)| - |C|$, since that is the number of components in a forest with $|C|$ edges. Create M by picking one arbitrary edge from each star. Then M is a matching of size $|V(G)| - |C|$, which implies that $\mathrm{m}(G) \geq |V(G)| - \mathrm{ec}(G)$.

Combining the two inequalities that we have obtained gives the equality in the lemma. □

Note that in a bipartite graph G, we have $\mathrm{m}(G) = \mathrm{vc}(G)$ by König's Theorem. Combining this with $\alpha(G) + \mathrm{vc}(G) = |V(G)| = \mathrm{m}(G) + \mathrm{ec}(G)$ from the previous two lemmas shows that in a bipartite graph we also have $\alpha(G) = \mathrm{ec}(G)$.

The proof of Lemma 2.2 shows that in principle, if we have an algorithm that finds a maximum independent set, then we also have an algorithm that gives a minimum vertex cover, and vice versa. Similarly, the proof of Lemma 2.2 shows that an algorithm that finds a maximum matching also gives a minimum edge cover, and vice versa. We have mentioned that there is a fast algorithm for finding a maximum matching, so it follows that there is one for finding a minimum edge cover. However, we will see in the next section that we do not have a fast algorithm for finding a maximum independent set; it follows that we also do not have one for finding a minimum vertex cover.

3 Independent sets

Next we prove a lower bound for the independence number. We give two proofs, one simple direct proof and one algorithmic proof.

Lemma 3.1. *For any graph G we have $\alpha(G) \geq \frac{|V(G)|}{\Delta(G)+1}$.*

Direct proof. Let I be an independent set in G of size $\alpha(G)$. A vertex $v \in V(G)\backslash I$ must be connected to a vertex in I, since otherwise I could be enlarged. Since any vertex has degree at most $\Delta(G)$, at most $\Delta(G)$ vertices of $V(G)\backslash I$ are connected to the same vertex in I. This implies that

$$|V(G)| - |I| \leq \Delta(G) \cdot |I|.$$

This gives the inequality in the lemma. □

Algorithmic proof. We use the following greedy algorithm. Pick any vertex v_1 and remove v_1 and its neighbors, then pick v_2 and remove v_2 and its neighbors, etc. Repeat until no vertices are left. Then $\{v_1, v_2, \ldots\}$ is an independent set. In each step, we remove at most $\Delta(G) + 1$ vertices. Hence we can execute at least $|V(G)|/(\Delta(G) + 1)$ steps, so the resulting independent step has at least that many vertices. □

The bound in Lemma 3.1 is tight for the complete graph K_n, which has $\Delta(K_n) = n - 1$ and $\alpha(K_n) = 1$. On the other hand, for many graphs the inequality is strict, like paths or cycles, which have $\Delta(G) = 2$ and $\alpha(G) \geq \lfloor |V(G)|/2 \rfloor$. For a star $K_{1,t}$ we even have $\Delta(K_{1,t}) = |V(K_{1,t})| - 1$ and $\alpha(K_{1,t}) = t$, so in this case the inequality is about as far off as possible.

We have seen that there is a fast algorithm[1] for various questions, like shortest paths, minimum-weight spanning tree, and maximum matchings. However, there does not seem to be any fast algorithm that finds a maximum independent set. The only algorithms that can find a maximum independent set are slow, while faster algorithms only find approximations. In fact, the best known algorithm for finding large independent sets is the greedy algorithm given in the proof of Lemma 3.1. It can be improved somewhat by picking v_i to be the minimum degree vertex in the remaining graph, although in a regular graph we would have $\delta(G) = \Delta(G)$, so this would not matter much.

The problem of finding a maximum independent set is the first example in this course of an "NP-hard problem". We won't go into the details of what this means, but roughly speaking it means that no fast algorithm is known. Moreover, if someone were to find a fast algorithm for such a problem, it would give a fast algorithm for many other hard problems; many computer scientists and mathematicians think that this is not possible (this is the "P=NP problem").

4 Dominating sets

Finally, we introduce one more graph parameter. We have seen vertex covers, where vertices cover edges, and edge covers, where edges cover vertices. It makes sense to also consider sets of vertices that "cover" all other vertices; to avoid confusion, we use the word "dominate".

[1] By "fast" we mean that the time the algorithm takes on an input is bounded above by a polynomial function of the size of the input. By "slow" we mean that there is no such polynomial; for instance, the time that the algorithm takes might be exponential in the size of the input. We do not formalize this here.

Definition. *Given $u, v \in V(G)$, we say that u* dominates *v if $v \in N(u) \cup \{u\}$. A set $D \subset V(G)$ is* dominating *if every vertex of G is dominated by a vertex in D; in other words, $D \cup N(D) = V(G)$. We write $\mathrm{dom}(G)$ for the minimum size of a dominating set in G.*

Theorem 4.1. *For any graph G we have*

$$\frac{1}{1+\Delta(G)}|V(G)| \leq \mathrm{dom}(G) \leq \frac{1+\log(\delta(G)+1)}{\delta(G)+1}|V(G)|.$$

Proof. The first inequality follows from the fact that any vertex dominates at most $\Delta(G)+1$ vertices.

For the second inequality, we use the following greedy algorithm: Repeatedly add the vertex that dominates the most vertices that are not yet dominated, until all vertices are dominated. We first prove a technical claim, which we will then use to prove that this algorithm gives the stated bound. Let $\delta = \delta(G)$ and $n = |V(G)|$.

Let $D \subset V(G)$ and set $U = V(G) \backslash (D \cup N(D))$. We claim that there is a vertex $y \notin D$ that dominates at least $\frac{\delta+1}{n}|U|$ vertices of U. The total number of pairs $(v, u) \in V(G) \times U$ such that v dominates u equals

$$\sum_{u \in U} |N(u) \cup \{u\}| \geq (\delta+1)|U|.$$

Thus, the average number of vertices of U that a vertex in $V(G)$ dominates is at least $\frac{\delta+1}{n}|U|$, which implies that there is a vertex y that dominates at least that many vertices of U. By definition of U, we have $y \notin D$.

The algorithm starts with D empty, and repeatedly picks the best vertex, which by the claim dominates at least $\frac{\delta+1}{n}$ vertices that were previously undominated. We now prove that after $\frac{\log(\delta+1)}{\delta+1}n$ steps, the algorithm has a set D that dominates all but at most $\frac{1}{\delta+1}n$ vertices. This requires some calculation.

If there are r vertices left undominated before step i, then, by the claim, there are at most

$$r - \frac{\delta+1}{n}r = r \cdot \left(1 - \frac{\delta+1}{n}\right)$$

vertices undominated after step i. Hence, after $\frac{\log(\delta+1)}{\delta+1}n$ steps, the number of undominated vertices is (using the inequality $(1-x)^{1/x} < \mathrm{e}^{-1}$, which follows from $1 - x < \mathrm{e}^{-x}$)

$$n \cdot \left(1 - \frac{\delta+1}{n}\right)^{\frac{\log(\delta+1)}{\delta+1}n} < n \cdot e^{-\log(\delta+1)} = \frac{n}{\delta+1}.$$

Even if the algorithm picks all the remaining vertices, the dominating set returned by the algorithm has size at most

$$\frac{\log(\delta+1)}{\delta+1}n + \frac{1}{\delta+1}n = \frac{1+\log(\delta(G)+1)}{\delta(G)+1}|V(G)|.$$

This finishes the proof. □

Finding a minimum dominating set is also an NP-hard problem, so we do not know any fast algorithm. The proof above shows that the greedy algorithm does reasonably well: It provides a dominating set that is at most a factor $\log(\delta(G))\frac{\Delta(G)}{\delta(G)}$ (roughly) larger than the minimum dominating set.

Theorem 4.1 also has a *probabilistic proof*, which works by selecting a random set of vertices. It is one of the classic examples of the probabilistic method in combinatorics.