CS 341 - Algorithms, Spring 2021. University of Waterloo

Lecture 16: Bipartite Vertex Cover

We study the minimum vertex cover problem on bipartite graphs, the "dual" problem of bipartite matching. We will prove a min-max theorem between these two problems, and see some further interesting applications.

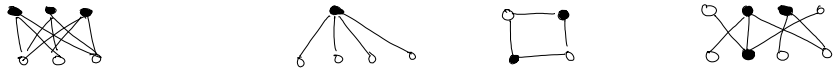
Bipartite Vertex Cover

Given a graph $G=(V,E)$, a subset of vertices $S \subseteq V$ is called a vertex cover if for every edge $uv \in E$, $\{u,v\} \cap S \neq \phi$.

In words, S is a vertex cover if S intersects every edge.

Input: A bipartite graph $G=(V,E)$

Output: A vertex cover of minimum cardinality.

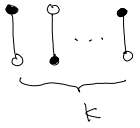
Some examples: 

We could imagine that this problem (in general graphs) is useful in computer networks, say by using the minimum number of computers to monitor all the links in the network.

Min-Max Theorem

The vertex cover problem may seem to be unrelated to the matching problem, but they are actually "dual" of each other on bipartite graphs, in a precise and meaningful way.

To start to see their connection, first we observe that if there is a matching with k edges, then any vertex cover must have at least k vertices.



The reason is simple: since the k matching edges are vertex disjoint, we must use k distinct vertices just to cover these k edges, and so any vertex cover must have at least k vertices.

Therefore, the size of a maximum matching is a lower bound on the size of a minimum vertex cover. Surprisingly, this is always a tight lower bound on bipartite graphs. Put it in another way, having a large matching is the only reason that we need a large vertex cover.

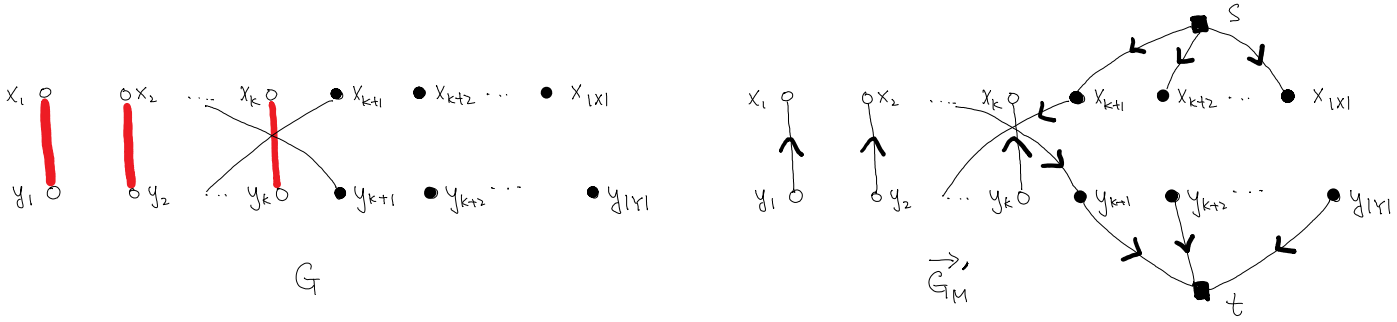
Theorem (König) In a bipartite graph, the maximum size of a matching is equal to the minimum size of a vertex cover.

Proof We will provide an algorithmic proof of König's theorem using the augmenting path algorithm. We will prove that given the current matching M of size k, if we couldn't find an augmenting path

of M, then we can find a vertex cover $C \subseteq V$ of size k.

This would imply that M is a maximum matching and C is a minimum vertex cover, because we know that there can't be a matching larger (because of C) and can't be a vertex cover smaller (because of M).

Following this plan, let's consider the iteration when there is no augmenting path of M.



Let the vertices of the bipartite graph be $X = \{x_1, x_2, \dots, x_{|X|}\}$ and $Y = \{y_1, y_2, \dots, y_{|Y|}\}$.

Let the current matching M be $\{x_1y_1, x_2y_2, \dots, x_ky_k\}$, having k edges, colored red in the picture.

Then the free / unmatched vertices are $x_{k+1}, \dots, x_{|X|}$ and $y_{k+1}, \dots, y_{|Y|}$, colored black in the picture.

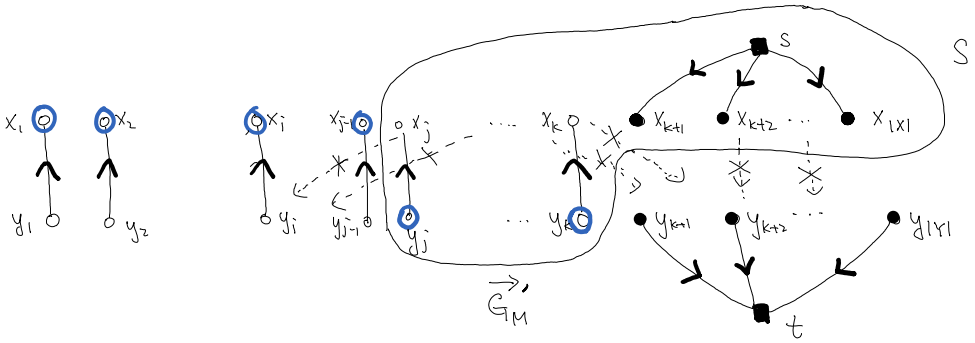
To search for an augmenting path of M, we consider the directed graph $\vec{G}_M$ as shown on the right, in which there is a super-source s with directed edges sx_i for all $k+1 \le i \le |X|$, and there is a super-sink t with directed edges y_jt for all $k+1 \le j \le |Y|$.

All the edges in $\vec{G}_M$ pointed downward from X to Y, except the edges in the matching M pointed upward from Y to X. (Not all the edges are shown in the picture.)

Now, suppose there is no augmenting path of M.

From last lecture, it is equivalent to that there is no directed path from s to t in $\vec{G}_M$.

From what we have learned in BFS / DFS, let's call the set of vertices reachable from s be S, then we know that there are no directed edges with its tail in S and its head in $V - S$ (as otherwise we could reach some vertices outside S).



Let's understand this picture.

Since s cannot reach t, no vertices in $\{y_{k+1}, \dots, y_{|Y|}\}$ are in S, as otherwise t can also be reached.

Let $\{y_j, \dots, y_k\}$ be the set of vertices in Y that are in S (i.e. reachable from s).

Then, $\{x_j, \dots, x_k\}$ are also in S because of the directed edges y_ix_i for $j \le i \le k$.

Also, since vertices $y_1, \dots, y_{j-1}$ are not in S by our assumption, the vertices $x_1, \dots, x_{j-1}$ are also not in S because the only way for s to reach $x_1 \dots x_{j-1}$ is through $y_1 \dots y_{j-1}$ as each x_i has only indegree one.

The following claim will finish the proof, by explicitly showing that there is a vertex cover of size k.

Claim $\{x_1, \dots, x_{j-1}\} \cup \{y_j, \dots, y_k\}$ is a vertex cover of G. (See the highlighted blue vertices.)

Proof It's clear that all matching edges are covered, so focus on the edges not in the matching, i.e. edges in $E-M$.

Since S has no outgoing edges, there are no directed edges from $\{x_j, \dots, x_{|X|}\}$ to $Y - \{y_j, \dots, y_k\}$, which means that there are no edges in $E-M$ are between $\{x_j, \dots, x_{|X|}\}$ and $Y - \{y_j, \dots, y_k\}$.

So, the vertices in $\{y_j, \dots, y_k\}$ cover all the edges in $E-M$ with one endpoint in $\{x_j, \dots, x_{|X|}\}$.

The remaining edges in $E-M$ have one endpoint in $\{x_1, \dots, x_{j-1}\}$, and so they are covered by $\{x_1, \dots, x_{j-1}\}$.

Therefore, $\{x_1, \dots, x_{k-1}\} \cup \{y_j, \dots, y_k\}$ covers all the edges, and thus a vertex cover of size k. □

This proves that both matching and vertex cover are optimal, and the min-max theorem follows. □

We summarize the algorithm in the proof as follows.

Algorithm (bipartite vertex cover)

1. Use an efficient algorithm to find a maximum matching M (doesn't need to be augmenting path algorithms).
2. Construct the directed graph $\vec{G}'_M$ as described above and in the last lecture.
3. Do a BFS/DFS on $\vec{G}'_M$ to identify all vertices $S \subseteq V$ reachable from s.
4. Return $(Y \cap S) \cup (X - S)$ as the vertex cover (i.e. $\{x_1, \dots, x_{j-1}\} \cup \{y_j, \dots, y_k\}$ in the picture).

Time complexity: It is dominated by the first step, as the remaining steps only take $O(n+m)$ time.

Using the augmenting path algorithm in L15 gives a $O(mn)$-time algorithm for bipartite vertex cover.

Using the $O(m\sqrt{n})$-time algorithm by Edmonds and Karp gives a $O(m\sqrt{n})$-time algorithm.

Good Characterization

König's theorem is a nice example of a graph-theoretic characterization.

To see this, imagine that you work for a company and your boss asks you to find a maximum matching say to assign some jobs to the employees.

If your algorithm finds a perfect matching, then your boss would be happy.

But suppose there is no perfect matching in the graph. how could you convince your boss about the non-existence?

Your boss may just think that you are incompetent and other smarter people could find a better assignment.

With the augmenting path algorithm, maybe you could explain that there is no augmenting path for

your matching and so it is maximum, but this may not be so easy to explain to your boss. A more convincing way is to show your boss a vertex cover of size $< \frac{n}{2}$, and explain that if there is a perfect matching such a vertex cover could not exist.

These min-max theorems are some of the most beautiful results in combinatorial optimization, providing succinct "proofs" for both the YES-case (a large matching) and the NO-case (a small vertex cover). They show the non-existence of a solution by the existence of a simple obstruction.

Remark: To put the above discussion into perspective, consider the dynamic programming algorithms. Even though we could solve the problems (such as optimal binary search tree) in polynomial time, we don't have such a nice succinct characterization for the NO-cases. If your boss asks why there is no better solution, it would be quite difficult to explain (e.g. we search for all possible solutions using this recurrence and our is an optimal solution). The "proof" is still succinct in the sense that it is a polynomial sized table, but it is not nearly as elegant as the proofs provided by the min-max theorems.

Hall's theorem characterizes when a bipartite graph $G=(X,Y;E)$ has a perfect matching or not. Without loss of generality, we assume that $|X|=|Y|$, as otherwise there is no perfect matching.

Theorem (Hall) A bipartite graph $G=(X,Y;E)$ with $|X|=|Y|$ has a perfect matching if and only if for every subset $S \subseteq X$, it holds that $|N(S)| \geq |S|$ where $N(S)$ is the neighbor set of S in Y. (In words, every subset $S \subseteq X$ has at least $|S|$ neighbors in Y.)

Exercise: Derive Hall's theorem from König's theorem.

Hall's theorem is an important result and has many applications in graph theory. We show one corollary, which may be used in homework.

Corollary Every d-regular bipartite graph $G=(X,Y;E)$ has a perfect matching.

Proof There are $d|X|$ edges in the graph. Note that $|X|=|Y|$ as G is regular.
Any subset of vertices S with $|S| \leq |X|-1$ can cover at most $d(|X|-1)$ edges.
So, any vertex cover needs at least $|X|$ vertices.
By König's theorem, there is a matching of size at least $|X|=|Y|$, hence a perfect matching. □

Applications

Applications

There are many interesting and non-trivial applications of graph matching and network flows.

We don't have time, so we will just see one such example, and we break the problem into two parts.

Capacitated job assignment

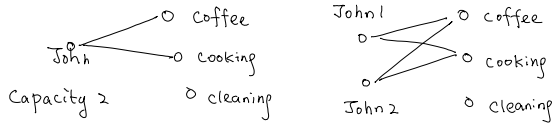
This is just a small generalization of the job assignment problem in the beginning of L15.

The same problem as before, but now each person / machine i has a different capacity c_i, representing how many jobs that person / machine i could handle (instead of just having capacity one for each person).

Then, again, we want to assign all the jobs to people, while no person will be overloaded according to their capacity.

We don't need to solve this more general problem again. There is a simple reduction to the special case.

Say John has capacity 2. Then we can simply create 2 vertices for John, say John1 and John2, connecting to the same subset of jobs as John did.



Then, it should be clear that there is an assignment of all jobs subjected to the capacities if and only if there is a matching in the new graph with all jobs matched.

This reduction may not be very efficient, as the number of vertices in the new graph could become $|X| \cdot |Y|$, as each person could have a capacity as high as $|Y|$.

But it is still polynomial in terms of the input instance, and this is good enough for our purpose.

There is a direct and more efficient way to work on the more compact representation (in a similar spirit to what we did for Dijkstra's algorithm as BFS-simulation in the big graph in L10), but we won't do it here.

Baseball / Basketball League Winner

Suppose you are a fan of Toronto Blue Jays or Toronto Raptors or Waterloo Warriors.

They are playing in the regular season. They are doing okay but not at the top of the table.

You wonder whether it is still (mathematically) possible for them to finish on top in the end of the season.

Input: The current standing / table, and the remaining schedule.

Output: Whether it is possible that your team can win the league.

The feature of the baseball / basketball league is that there is only win or lose (i.e. no draw as in football).

Suppose your team has currently won W games.

First, it is obvious that we assume that they will win all their remaining games, for a total of W^* games.

Then, we want to know whether there is a scenario that every other team will win less than w^* games.

Let the other teams be $\{1, 2, \ldots, n\}$ and currently team i has won w_i games.

We have the remaining schedule, showing that there are still g_{ij} games to be played between team i and team j.

How do we determine if there is still a scenario that your team can still win the league?

Solution: It is not difficult to reduce it to the capacitated assignment problem once we see it the right way.

We want to "assign" the winner of each remaining match so that no team will win more than $w^* - w_i - 1$ games.

In more detail, each team i is a person in the capacitated assignment problem, with capacity $w^* - w_i - 1$.

Each remaining match is a job in the capacitated assignment problem. If a match is between team i and team j, then the corresponding job can only be handled by team i and team j.

Then, your team can still win the league if and only if all the jobs can be assigned satisfying the capacity constraints (i.e. all the remaining matches can be played without any team win more than $w^* - w_i - 1$ games).

Remark: It is an NP-hard problem to determine if your favorite football team can still win the league (i.e. winning team gets 3 points, each team gets one point in a draw).

Looking Forward

Just a quick overview of what is ahead in this topic.

Other applications: The network flow problem can be solved using the augmenting path method for bipartite matching.

One useful application is to find the maximum number of edge-disjoint paths between two vertices s and t, which is an important problem both in theory and in practice.

The famous "max-flow min-cut" theorem is a very important min-max theorem, which can be proved using similar techniques as in the algorithmic proof of König's theorem.

You can learn more about network flows from [KT, DPV, CLRS] and you should be ready now.

Extensions: The maximum matching problem on general graphs can be solved in polynomial time.

Even the maximum weighted version, where every edge has a weight, can be solved in polynomial time.

This was a groundbreaking result by Edmonds in the 70s.

The maximum weighted matching problem can be used, in a clever way, to solve the Chinese postman problem that we mentioned in L01 in polynomial time.

Duality: Why vertex cover is called the "dual" problem of matching? How could we come up with it?

Actually, there is a systematic way to define the dual problem of an optimization problem, through the use of linear programming.

Most combinatorial optimization problems can be solved in the general framework of linear programming. This is one of the most, if not the most, powerful algorithmic framework for polynomial time computation. The many beautiful min-max theorems can also be systematically derived from linear programming duality. Unfortunately, we won't learn about linear programming in this course; see [KT, DPV, CLRS] for more. There are also various courses in the C&O department on these topics.