Stanford MS&E 236 / CS 225: Lecture 5
Graphs and graph algorithms

Ellen Vitercik*

April 17, 2024

Many discrete optimization problems can be formulated in terms of *graphs.* In this lecture, we will go over basic graph notation and jargon to make sure we're on the same page. We will also explore a common algorithmic paradigm for solving graph problems: greedy algorithms.

1 Graphs

Abstractly, a graph consists of a set of objects called *nodes*, which are connected by *edges*. The nodes and edges may have data associated with them describing some real-world system, as illustrated by the following examples:

Molecules. To represent a molecule, a graph's nodes represent atoms with features describing the atom's type (e.g., carbon, nitrogen, hydrogen, ...). The edges represent bonds, with features describing the bond type (e.g., single bond, double bond, ...).

Rail networks. In this case, nodes are cities, and edges are connections with features describing the journey time between them.

Social networks. In a social network, the nodes represent people, and the edges represent friendships.

We use $G = (V, E)$ to denote a graph, where $V = \{1, \ldots, n\}$ is a set of nodes/vertices and E is a set of edges/links. The edge between $i, j \in V$ is denoted (i, j). The *neighborhood* of the node $i \in V$ is the set of nodes it is connected to by an edge, denoted as

$$N(i) = \{j : (i, j) \in E\}.$$

The cardinality of this set, $|N(i)|$, is called the *degree* of vertex i.

In the next sections, we will cover several classic graph problems together with simple *greedy* algorithms that return (approximate) solutions.

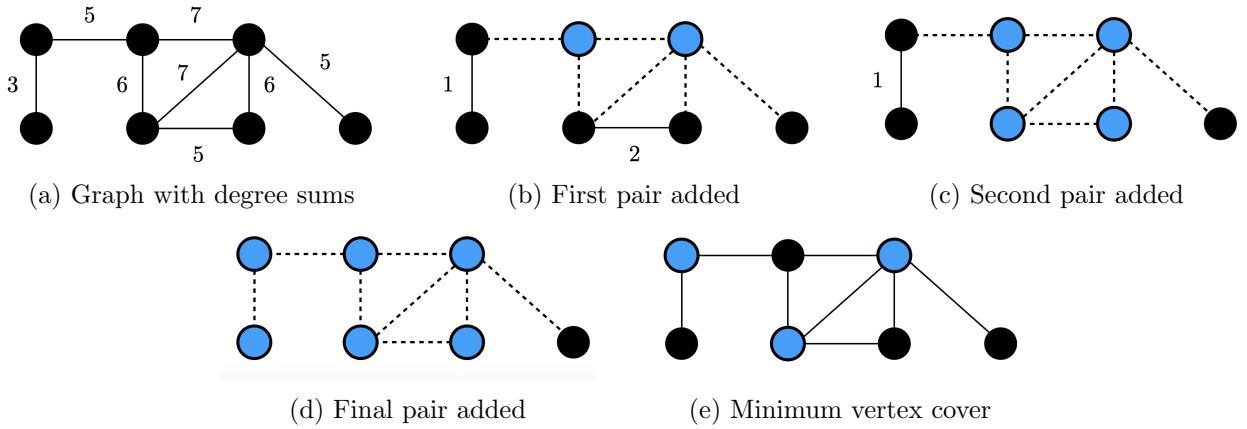
*These notes are course material and have not undergone formal peer review. Please feel free to send me any typos or comments.

Algorithm 1 MVC 2-approximation algorithm

Input: Graph $G = (V, E)$

1: Initialize the vertex cover $S \leftarrow \emptyset$
2: **while** $E \neq \emptyset$ **do**
3: Choose the edge $(i, j) \in E$ with the largest *degree sum*, i.e., which maximizes $|N(i)| + |N(j)|$
4: Add i and j to the vertex cover: $S \leftarrow S \cup \{i, j\}$
5: Remove all edges incident to i and j from E

Output: The vertex cover S



(a) Graph with degree sums (b) First pair added (c) Second pair added

(d) Final pair added (e) Minimum vertex cover

Figure 1: Illustration of Algorithm 1 for MVC. The blue nodes form the vertex cover, and the dotted edges are those that have been deleted from the graph.

2 Minimum vertex cover

We begin with the *minimum vertex cover (MVC)* problem. A vertex cover is defined as follows.

Definition 2.1 (Vertex cover)**.** A vertex cover of a graph $G = (V, E)$ is a set $S \subseteq V$ such that every edge $(i, j) \in E$ is incident to a vertex in S, i.e., $i \in V'$, $j \in S$, or both.

In the MVC problem, the goal is to find a vertex cover S with minimum cardinality $|S|$. An example application is installing cameras in corners (represented by the nodes) covering all hallways (represented by the edges) on a floor. Finding the minimum vertex cover is NP-hard (meaning, informally, that there is likely not an algorithm that solves this problem much faster than a brute force algorithm that checks if all $2^{|V|}$ sets of vertices are vertex covers). There are, however, efficient algorithms that return approximate solutions, such as Algorithm 1, which is illustrated by Figure 1.

Algorithm 1 returns a vertex cover because an edge is only deleted if it is incident to a newly-added vertex in S, and the algorithm terminates when all edges have been deleted. In the following theorem, we prove that Algorihtm 1 is a 2-approximation algorithm.

Theorem 2.2. *Let S^* be the minimum vertex cover and S be the output of Algorithm 1. Then $|S| \leq 2|S^*|$.*

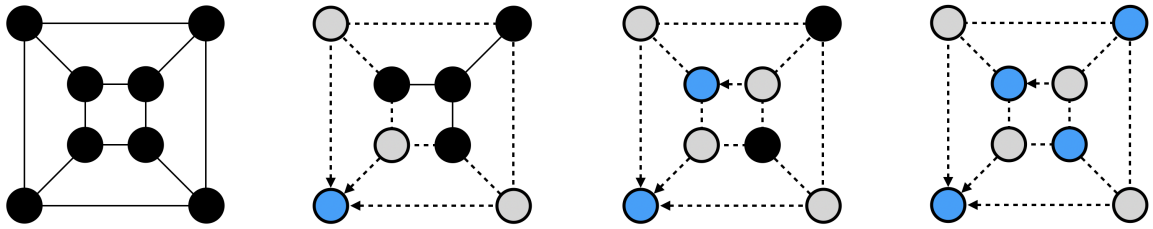
Proof. Let E' be the set of edges the algorithm picked in Step 3. Since S^* is a vertex cover, it must include at least one node incident to each edge in E'. By construction, no two edges

Algorithm 2 MIS $\frac{1}{1+\Delta}$-approximation algorithm

Input: Graph $G = (V, E)$

Initialize the independent set $S \leftarrow \emptyset$
while $V \neq \emptyset$ **do**
    Choose the vertex $v \in V$ with the minimum degree $|N(v)|$
    Add v to the independent set: $S \leftarrow S \cup \{v\}$
    Remove v and all of its neighbors from V

Output: The independent set S



(a) Original graph (b) First vertex (c) Second vertex (d) Third and fourth

Figure 2: Illustration of Algorithm 2 for MIS. In the algorithm's first round, we add the bottom-left blue vertex in Figure 2b to the independent set and delete its neighbors from teh graph, illustrated in grey. In the next round, we add the second blue vertex in Figure 2c to the independent set and delete its remaining neighbor. The dotted edges are those that have been deleted from the graph. The arrows are explained in the proof of Theorem 3.2.

in E' share an endpoint, because once an edge is chosen, all of the edges incident to its endpoints are deleted. This means that no two edges are incident to the same vertex in S^*, which means that $|S^*| \leq |E'|$. Therefore, $|S| = 2|E'| \leq 2|S^*|$. □

3 Maximum independent set

Our next example of a famous graph problem is the *maximum independent set (MIS)* problem, which is closely related to MVC. An independent set is defined as follows.

Definition 3.1 (Independent set)**.** Given a graph $G = (V, E)$, an independent set is a set $S \subseteq V$ of vertices such that no vertices in S are connected by an edge.

In the MIS problem, the goal is to find an independent set S with largest cardinality $|S|$. MIS is also NP-hard, but there are approximation algorithms. Letting $\Delta = \max_{i \in V} |N(i)|$ denote the graph's maximum degree, Algorithm 2 is a $\frac{1}{1+\Delta}$-approximation algorithm. Algorithm 2 is illustrated by Figure 2.

Algorithm 2 returns an independent set S because once a node is added to S, its neighbors are deleted from the graph. The following theorem proves that Algorithm 2 is a $\frac{1}{1+\Delta}$-approximation algorithm.

Theorem 3.2. *Let V^* be the maximum independent set and S be the output of Algorithm 2. Then $|S| \geq \frac{|V|}{\Delta+1} \geq \frac{|V^*|}{\Delta+1}$.*

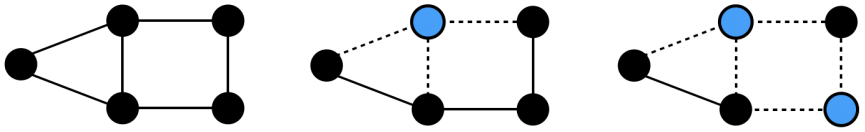
Proof. We will begin by bounding the size of the set $|V \setminus S|$. A node u is in $V \setminus S$ if and only if it's removed as a neighbor of some node $v \in S$, and it's removed when v is added to

Algorithm 3 Max-cut $\frac{1}{2}$-approximation algorithm

Input: Graph $G = (V, E)$

1: Begin with an arbitrary initial cut $S \subseteq V$
2: For each vertex $i \in V$, let w_i be the cut's weight if you switched the side i was on
3: If $w_i \leq w(S)$ for all i, terminate and return S
4: Otherwise, let $i^* = \text{argmax}\{w_i\}$. Switch the side of the cut that i^* is on and return to Step 2

Output: The independent set S



(a) Original graph (b) First swap (c) Second swap

Figure 3: Illustration of Algorithm 3 maximum cut. The dotted edges cross from one side of the cut to the other.

S. We will "charge" u to v, as illustrated in Figure 2. For example, when we add the blue node in the bottom left-hand corner of Figure 2b, we remove its three neighbors, illustrated in grey. The arrows illustrate that we "charge" each of these grey nodes to the blue node. Similarly, in Figure 2c, when we added the next blue node to the independent set, we remove its last remaining neighbor, and the arrow indicates that we "charge" the grey node to the blue node.

Each node $v \in S$ can be charged at most Δ times since it has at most Δ neighbors. This implies that $|V \setminus S| \leq \Delta|S|$. Therefore, $|V| = |V \setminus S| + |S| \leq |S|(1 + \Delta)$, which implies the theorem statement. □

4 Maximum cut

The last graph problem we will explore today is *maximum cut*. One of the most famous approximation algorithms—the Goemans-Williamson algorithm [1]—was developed for max-cut. Today, we will analyze a simpler approximation algorithm.

First, a *cut* in a graph $G = (V, E)$ is a subset of its vertices $S \subseteq V$. The cut's *weight* $w(S)$ is the number of edges that cross between S and $V \setminus S$. The goal is to find a cut with maximum weight.

Algorithm 3, which is illustrated in Figure 3, is a $\frac{1}{2}$-approximation algorithm for max-cut. Algorithm 3 terminates in at most $|V|$ rounds because in each round, the cut weight increases by at least 1 (or else the algorithm terminates), and $w(S) \in [0, |V|]$. Next, we prove that it is a $\frac{1}{2}$-approximation algorithm.

Theorem 4.1. *Let S^* be the maximum weight cut and S be the cut that Algorithm 3 returns. Then $w(S) \geq \frac{1}{2}w(S^*)$.*

Proof. First, we claim that $w(S) \geq \frac{|E|}{2}$. To see why, for any vertex $i \in S$, some of its neighbors in $N(i)$ are in S and some of its neighbors are in $V \setminus S$. It must be that more of its neighbors are in $V \setminus S$ than S, or else we could switch i to $V \setminus S$ and improve the

cut's weight. Therefore, the number of edges incident to i that cross the cut is at least $\frac{|N(i)|}{2}$. Moreover,

$$w(S) \geq \frac{1}{2} \sum_{i \in V} (\text{\# of } i\text{'s incident edges that cross the cut}),$$

where the $\frac{1}{2}$ factor keeps us from overcounting. Therefore,

$$w(S) \geq \frac{1}{2} \sum_{i \in V} \frac{|N(i)|}{2} \geq \frac{1}{4} \sum_{i \in V} |N(i)| = \frac{1}{4} \cdot 2|E|.$$

Meanwhile, $w(S^*) \leq |E|$, so $w(S) \geq \frac{|E|}{2} \geq \frac{1}{2} w(S^*)$. □

5 Greedy algorithms

Algorithm 1, 2, and 3 are all *greedy* algorithms: in an iterative fashion, they "greedily" choose a node to add/remove from the solution which maximizes or minimizes some score. For example:

MVC. Algorithm 1 chooses the edge (i, j) with maximum degree sum $|N(i)| + N(j)|$ and adds its endpoints to the vertex cover.

MIS. Algorithm 2 chooses the node with minimum degree $|N(i)|$ and adds it to the independent set.

Max-cut. Algorithm 3 swaps the node that leads to the larges improvement of the cut weight.

In this lecture, we saw that these intuitive, simple, hand-designed rules yield decent approximation algorithms. In the next few classes, we will see if we can use ML to learn an even better scoring rule (perhaps not in the worst case, but for non-worst-case families of graphs).

References

[1] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.