COMS 4995-3: Advanced Algorithms — Feb 22, 2017

Lecture 10 – Graph Algorithms, Introduction to Network Flows

Instructor: *Alex Andoni* — Scribe: *Aubrey Alston*

1 Introduction

In today's lecture, we introduce and explore the notion of network flows as a first topic in graph algorithms. Since this is the first lecture in which we consider graphs, we begin by giving a short exposition of the formal definition of a 'graph,' and we continue in the remainder of the lecture to motivate, define, and begin to consider solutions to the maximum flow problem.

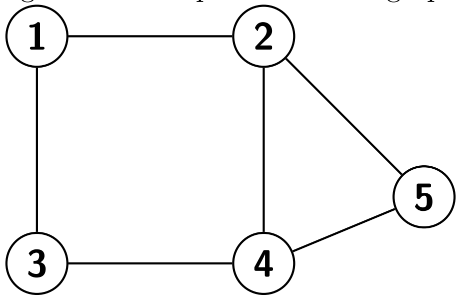
1.1 Graph-theoretic Notions

Definition 1 (Graph)**.** *A* ***graph*** *is an ordered pair $G = (V, E)$ consisting of a set of nodes V and a set of edges $E = \{(u, v) \mid u \in V, v \in V\}$ connecting nodes in V. (As a matter of convention, we say $|V| = n$ and $|E| = m$.)*

Among graphs, we further distinguish between *directed* and *undirected* graphs.

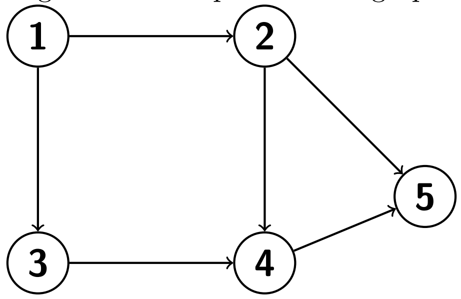
Definition 2 (Undirected Graph)**.** *A* ***undirected graph*** *is a graph G whose edges do not maintain a sense of direction. Informally stated, for edge $e = (u, v) \in E$, we may freely travel along e from u to v and from v to u. More formally, an undirected graph is a graph $G = (V, E)$ whose edge set E represents a symmetric binary relation over V.*

Figure 1: A simple undirected graph.



Definition 3 (Directed Graph)**.** *A* ***directed graph*** *is a graph G whose edges do maintain a sense of direction. Informally, for edge $e = (u, v) \in E$, we may only travel along e from u to v. Formally, an undirected graph is a graph $G = (V, E)$ whose edge set E need not necessarily be a symmetric binary relation over V.*

Figure 2: A simple directed graph.



In practice, graphs are used to model many real-world systems and scenarios (e.g. ISP interactions, distribution costs, schedulers,...); as such, graph problems are especially interesting because they have widespread influence and applicability. We mention (but do not explore in great depth) some elementary concepts in the realm of graph algorithms:

- **Breadth-first search** as a solution to the problems of general $s-t$ reachability and single-source shortest paths in unweighted graphs.
- **Dijkstra's algorithm** as a solution to the problem of single-source shortest paths in weighted digraphs.

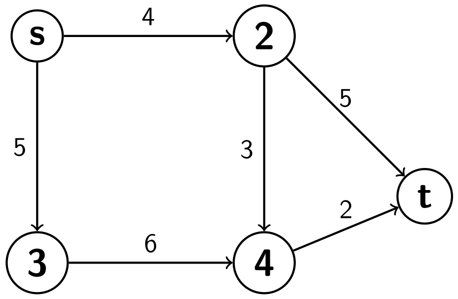
2 The Maximum Flow Problem

2.1 Informal Motivation

Let's say that we're in charge of a segment of a city's water distribution system; this system is composed of pipes connected by pump locations. The pumps at each location cannot store water, but they may arbitrate how much water enters a subset of connected pipes (some pipes will only provide incoming water; some are used as outgoing pipes). Our goal, under the constraints of our pump capabilities and the amount of water that each pipe can hold, is to configure our system to maximize the amount of water entering and exiting the system.

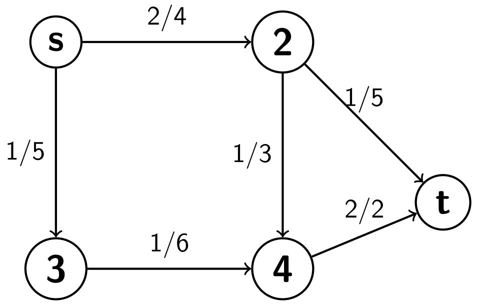
To represent our segment of the water system, we might use a directed graph, where nodes are pump locations, edges represent pipes, the direction of and edge represents the direction in which water may be pumped through said pipe. We also augment the graph by labeling the entrypoint of our system (the source) as s and the exit point of our system (the destination) as t; we also associate with each edge a *capacity* which represents how much water may flow through a pipe.

Figure 3: Our distribution system



Denoting by a/b a units of flow utilized out of capacity b, one solution we try (but which is not optimal) could be something like this:

Figure 4: A potential flow in our distribution network, value 3



In this example, our desired pump configuration corresponds precisely to a maximum (single-source, single-destination) flow in our pictured network.

2.2 Defining 'Flow'

Flow is defined with respect to a directed graph $G = (V, E)$ (a network), a source s, a destination t, and an associated function $u(\cdot, \cdot)$ indiciating non-negative edge capacities: $\forall (v, w) \in E, u(v, w)$ is the capacity of edge (v, w).

Definition 4 (Flow)**.** *A **flow** $\vec{f_e}$ is a vector of scalars valued for all edges in G satisfying the following properties:*

1. ***Conservation:*** $\forall v \neq s, t \sum_{w,(v,w) \in E} f(v, w) - \sum_{w,(w,v) \in E} f(w, v) = 0$
2. ***Capacity:*** $\forall e \; 0 \leq f(e) \leq u(e)$
3. ***Flow value:*** $|f| = \sum_{w,(s,w) \in E} f(s, w) - \sum_{w,(w,s) \in E} f(w, s)$. [1]

Definition 5 (Path Flow)**.** *A **path flow** is a flow of G where edges with non-zero flow compose a simple path.*

Definition 6 (Cycle Flow)**.** *A **cycle flow** is a flow of G where edges with non-zero flow compose a cycle.*

For a graph G, a **maximum flow** is a flow f having maximum $|f|$ over all possible graph flows.

[1]Note that $|f|$ doesn't refer to a norm; it is simply notation used to denote the value of a flow.

2.3 The Decomposition Theorem

Theorem 7 (Decomposition Theorem)**.** *For any given flow f, f may be decomposed as a sum of path flows and cycle flows.*

Equivalently, there exist paths $p_1, ..., p_k$, cycles $c_1, ..., c_t$, and values $v_1, ..., v_k, w_1, ..., w_t$ such that, for any edge $e = (x, y) \in E$, $f_e = \sum_{i, e \in p_i} v_i + \sum_{j, e \in c_j} w_j$.

Proof. By induction on the number of edges in f with non-zero flow.

Case 1. *There exists an edge $e = (s, x) \in E$ such that $f_e > 0$.*

Then there must necessarily exist flow out of x (unless $x = t$). This implies that we may travel node-to-node via edges e', $f_{e'} > 0$ until either we (a) reach t or (b) cycle back. If (a) occurs, then we have found a path P' with $f(e') > 0$ for all $e' \in P'$. If (b) occurs, then we have found a cycle C' with the same property. Let X be either the path or cycle recovered in this process.

Let $\delta = \min_{e' \in X} f(e')$. Construct a new flow $f'(e')$:

$$\begin{cases} f'(e') = f(e') - \delta & e' \in X \\ f'(e') = f(e') & else \end{cases}$$

Note that f' has strictly fewer edges with non-zero flow and that $f = f' + \delta$ for edges on X.

Case 2. *There exists an edge $e = (x, t) \in E$ such that $f_e > 0$.*

From t, we may backtrack using edges with positive flow until we find either (a) s or (b) a cycle. If (a) occurs, we find a path flow; if (b) occurs, we find a cycle flow. We may then proceed precisely as we did in the previous case.

Case 3. *There exists an edge $e = (x, y) \in E$ such that $f_e > 0$ (but which does not fall into either of the previous two cases).*

In this case, we may necessarily be able to trace back towards s or forward towards t; we proceed in precisely the same manner as the previous two cases.

□

2.4 Towards an Algorithm for Maximum Flow

In the rest of the lecture, we begin to substantiate a method of solving the maximum flow problem. We begin by asking the following decision question: given a directed graph G, does there exist a positive flow f from s to t?

Observation 8. *The flow out of s is equal to the flow into t. This implies that a cycle containing s and t does not affect the value of a flow.*

Claim 9. *There exists a flow f, $|f| > 0$, from s to t if and only if t is reachable from s using edges with non-zero capacity.*

Proof. (⇒) Assume that there exists a flow f from s to t, $|f| > 0$. By the decomposition theorem (and as a result of the above observation), there must exist a (non-zero) path flow from s to t. By the definition of a non-zero path flow from s to t, there exists a path consisting of non-zero-capacity edges from s to t.

(⇐) Assume that there exists a path from s to t using edges with non-zero capacity. Then this path is a positive flow from s to t. □

Given the truth of this claim, then, we may answer our decision problem by performing a simple breadth-first search from s and answering 'yes' if and only if we find that t is reachable. We then ask the following related question: given a directed graph G and a positive flow f from s to t, how do we verify that f is a maximum flow?

We begin to answer this question by seeking an *upper bound* on the maximum $s-t$ flow of G. Consider an extant (though unknown) maximum $s-t$ flow f of G. Form an arbitrary *S-T cut* of G by taking some subset $S \subseteq V$ and $T = V - S$. We may then determine a value of this cut, $u(S)$, as the weighted sum of edges crossing the cut

$$\sum_{(w,x)\in E, w\in S, x\in T} u(w,x)$$

For $s \in S$, $t \in T$, since the non-zero edges of f crossing the cut are a subset of all edges of G crossing the cut, we are able to conclude that $|f| \leq u(S)$, giving us our desired upper bound. Moreover, by the **min-cut max-flow theorem**, it is the case that the value of the minimum $S-T$ cut corresponds to the maximum $s-t$ flow of G.

We conclude the lecture by giving a procedure whose intent is to find the maximum $s-t$ flow of G:

1. Find an arbitrary $s-t$ path flow P.
2. Set $f_{e\in P}(e) = \min_{e'\in P} u(e')$.
3. Repeat: find a new path P' where, for all constituent edges e, $u(e) - f(e) > 0$
   (a) Let $\delta_p = \min_{e'\in P'} u(e') - f(e')$
   (b) $f^{new}(e) = f^{old}(e) + \delta_p \; \forall e \in P'$
   (c) Set $f^{old} = f^{new}$.

The procedure given does not actually find the maximum flow. We will correct its shortcomings in the next lecture.