CSC 373 H 1 Y — Summer 2007

University of Toronto — St. George Campus

Lecture Summary for Week 6

This summary is not a replacement for the lecture. If you miss a class, please arrange with a friend to take note for you.

3.4 Residual Networks

Consider the flow network from the previous lecture given in Figure 1 below. We want now to improve this flow. An idea is to try to increase the flow from s to t along some path. So the first question is along which path can we improve the flow? Here we can look at an undirected st-path (where we ignore the direction of the edges). In general, there are edges where the flow goes on the direction from s to t, and other edges where the flow goes on the opposite direction. Notice that we can improve the total flow of the network by increase the flow on the former type of edges and decrease the flow on the latter type of edges.

For example, in Figure 1, looking at the undected path Vancouver–Calgary–Saskatoon–Winnipeg, we can increase the flow by adding 4 to the flows on edges Vancouver–Calgary and Saskatoon–Winnipeg and substracting 4 from the flow Sakatoon–Calgary.

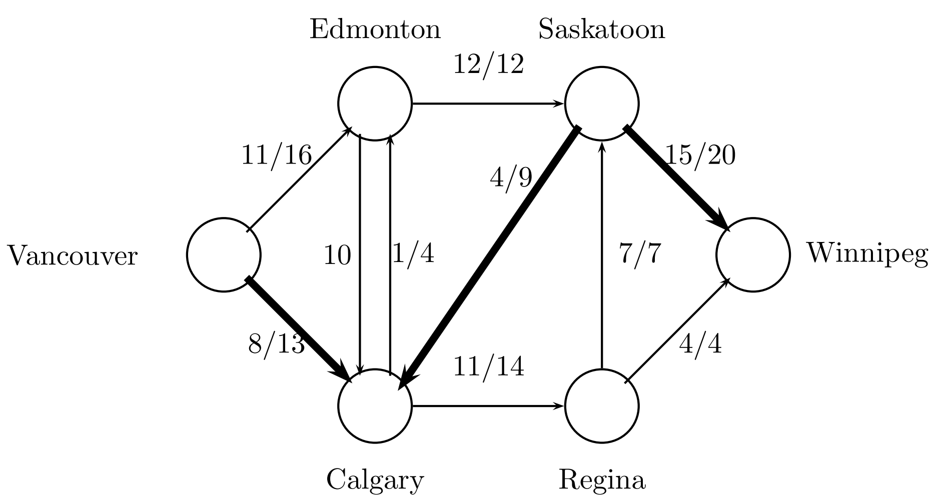

Figure 1: A "flow" of value 19

Obviously, to preserve the Capacity Condition for the flow, the change that we can make to the flow on an edge is bounded by the remaining capacity (if we want to add to the flow), or the value of the flow (if we want to cancel some flow). This motivates the following definitions:

Residual Network: Given a flow network $G = (V, E)$ and a flow f on G. The *residual network* of f, denoted by G_f, is the flow network with the same set of nodes as G, whose edges and capacities are as follows:

- For each edge e in G, if $f(e) < c(e)$ then e is in G_f with capacity $c_f(e) = c(e) - f(e)$ (the "leftover" capacity, e is also called a *forward* edge).

- For each edge $e = (u, v)$ in G, if $f(e) > 0$ then (v, u) is in G_f with capacity $c_f(v, u) = f(u, v)$. ((v, u) is also called a *backward* edge.)

Notice that if $e = (u, v)$ in G with $0 < f(e) < c(e)$, then both (u, v) (the forward edge) and (v, u) (the backward edge) are in G_f. The residual network for the flow in Figure 1 is given in Figure 2

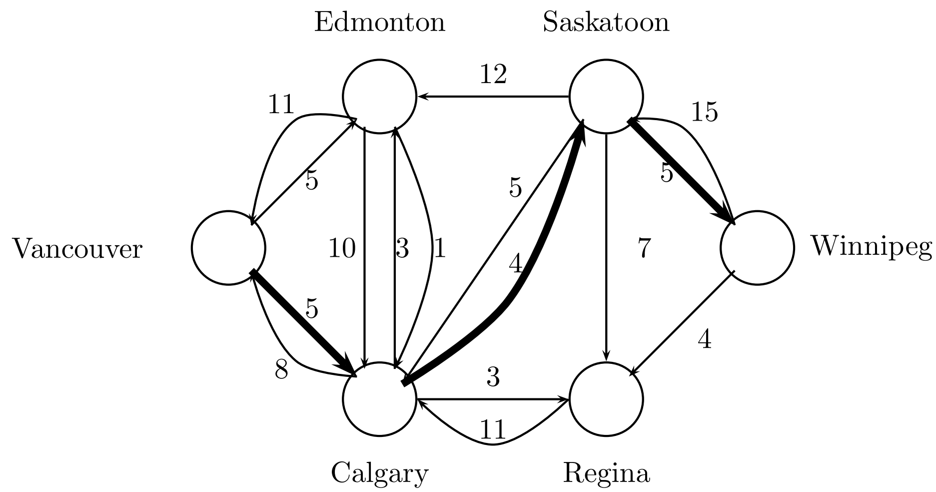

Figure 2: The residual network of the flow in Figure 1.

Augmenting Path: An st-path in G_f is also called an *augmenting path*.

Note that augmenting paths are directed paths from s to t.

The amount of flow that can be increased along an augmenting path P is denoted by bottleneck(P), and is the minimum capacity of all edges on the path:

$$bottleneck(f, P) = min\{c_f(e) \ : \ e \text{ in } P\}$$

The augmenting path that we looked at in the Lucky Puck Company example above is highlighted in Figure 2.

The following algorithm increase the flow f along an augmenting path P by $bottleneck(f, P)$:
Augment(f, P)

1. $f' \leftarrow f$ % f' is the output flow.
2. $b \rightarrow bottleneck(f, P)$
3. For each $e = (u, v)$ in P do
4.   If e is a forward edge do
5. $f'(e) \leftarrow f(e) + b$
6.   Else % e is a backward edge
7. $f'(v, u) \leftarrow f(v, u) - b$
8.   End If
9. End For

10. Output f'.

We have to show that the output f' of Augment(f, P) is a valid flow, and that it does improves on f:

Lemma: The output f' of Augment(f, P) is a flow with value:

$$Value(f') = Value(f) + bottleneck(f, P)$$

Proof: To show that f' is a flow, we have to verify the Capacity condition and Conservation condition.

For the Capacity condition, for each edge $e = (u, v)$ on the augmenting path P, we consider two cases: e is a forward edge, or e is a backward edge. First, suppose that e is a forward edge, then

$$f'(e) = f(e) + bottleneck(f, P) \leq f(e) + c_f(e) = f(e) + (c(e) - f(e)) = c(e)$$

Also, $f(e) \geq 0, bottleneck(f, P) \geq 0$, so $f'(e) \geq 0$. Next, suppose that e is a backward edge, then

$$f'(v, u) = f(v, u) - bottleneck(f, P) \geq f(v, u) - c_f(v, u) = 0$$

and $f'(v, u) \leq f(v, u) \leq c(v, u)$. Thus the Capacity condition holds.

For the Conservation condition, we have to verify that

$$f'^{out}(v) - f'^{in}(v) = 0 \tag{1}$$

for each node v on P. Suppose that (u, v) and (v, w) are the two edges in P that are incident on v. Each of these can be either forward or backward. Thus there are 4 cases to consider. Consider, for example, the case where (u, v) is forward, and (v, w) is backward. The other cases will be left as an exercise.

Then $f'(u, v) = f(u, v) + b$, and $f'(w, v) = f(w, v) - b$. Therefore $f'^{in}(v) = f^{in}(v)$, and $f'^{out}(v) = f^{out}(v)$. So

$$f'^{out}(v) - f'^{in}(v) = f^{out}(v) - f^{in}(v) = 0$$

Exercise: Verify that (1) holds for other cases.

Now we show that $Value(f') = Value(f) + bottleneck(f, P)$. This is true because there is only one edge e on P that leaves s, and that must be a forward edge. Then

$$f'(e) = f(e) + b$$

So the total flow going out of s increases by b. QED

The Augment procedure will be useful later for the Ford-Fulkerson algorithm. Before presenting this algorithm, we prove the Max Flow Min Cut Theorem, which is useful for proving that the algorithm is correct.

3.5 Max Flow–Min Cut Theorem

Theorem: Suppose that f is a flow in a flow network G with source s, sink t. Then the following are equivalent:

1. f is a maximum flow in G.
2. The residual network G_f has no augmenting path.
3. $Value(f) = c(A, B)$ for some cut (A, B).

Proof: We will show that (1) $\Longrightarrow$ (2) $\Longrightarrow$ (3) $\Longrightarrow$ (1).

(1) $\Longrightarrow$ (2): We prove the contrapositive: If G_f has an augmenting path, then from the Lemma in the previous section there is a flow f' which has larger value than f, and f cannot be a maximum flow.

(2) $\Longrightarrow$ (3): The cut (A, B) is defined as follows: A is the set of all nodes reachable from s in G_f. By (2), t is not in A. Let $B = V - A$.

First, for all edges $e = (u, v)$ from A to B (i.e., $u \in A, v \in B$) we show that $f(e) = c(e)$. This is because u is reachable from s in G_f. If $f(u, v) < c(u, v)$ then the forward edge (u, v) in G_f will make v also reachable from s, contradict the fact that $v \notin A$.

Next, for all edge $e = (u, v)$ from B to A (i.e., $u \in B, v \in A$) we show that $f(e) = 0$. The argument is similar as above: If $f(u, v) > 0$ then the backward edge (v, u) will make u also reachable from s, contradict the fact that $u \in B$.

As a result,

$$f^{out}(A) = c(A, B), \qquad \text{and} \qquad f^{in}(A) = 0$$

So $c(A, B) = f^{out}(A) - f^{in}(A)$, i.e., $c(A, B) = Value(f)$.

(3) $\Longrightarrow$ (1): Corollary 3 we proved last week states that

$$Value(f') \leq c(A, B)$$

for all flow f'. So for $Value(f') \leq Value(f)$ for all flow f'. Thus f is a maximum flow. QED.

3.6 Ford-Fulkerson Algorithm

1. For each e in G do $f(e) \leftarrow 0$ End For % start with a 0 flow
2. While there is an st-path in G_f do
3. $\quad P \leftarrow$ an st-path in G_f
4. $\quad f' \leftarrow Augment(f, P)$
5. $\quad f \leftarrow f'$
6. $\quad$ Update G_f.
7. End While
8. Return f.

Before concerning with the correctness of the algorithm, we will have to show that it terminates. It turns out that this algorithm is NOT guaranteed to terminate unless all capacities are rational numbers. Next week we will show that when all capacities are integers, the algorithm will terminate. (This suffices, since if all capacities are rational numbers, we can multiply all by a common factor which will make all capacities integers.) The fact that the algorithm in that case returns a maximum flow follows from the Max Flow–Min Cut Theorem. Then we will discuss the running time of the algorithm.