November 17, 2015

Math 3012 – Applied Combinatorics Lecture 24

William T. Trotter
trotter@math.gatech.edu

Reminders

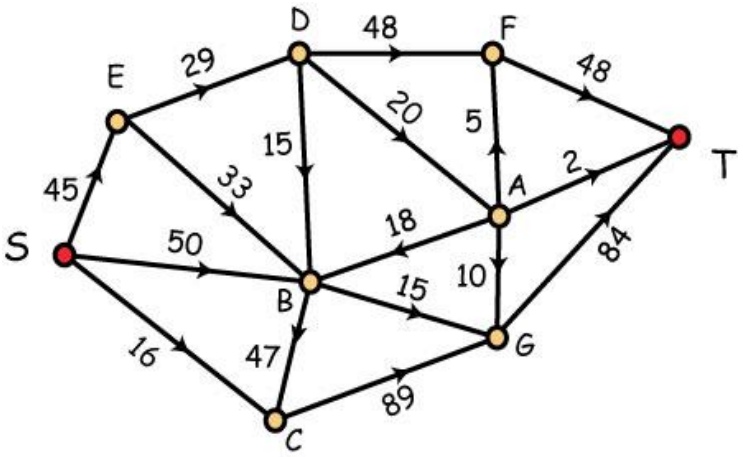
Test 3 Tuesday, November 24, 2015

Final Exam Tuesday, December 8, 2015, 8:05 – 10:55am.

Three-way Option (Full details in email)

1. Do even numbered problems from assigned set.
2. Obtain/write code for implementing one of the algorithms in our course on my data set.
3. Write 3 – 4 page (typewritten) report on one of the selected math papers, all of which are accessible to undergraduates.

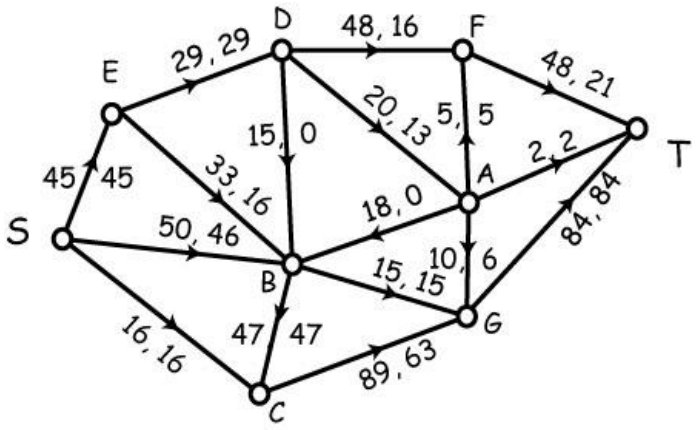
The Max Flow/Min Cut Theorem



Theorem The maximum value of a flow is equal to the minimum capacity of a cut.

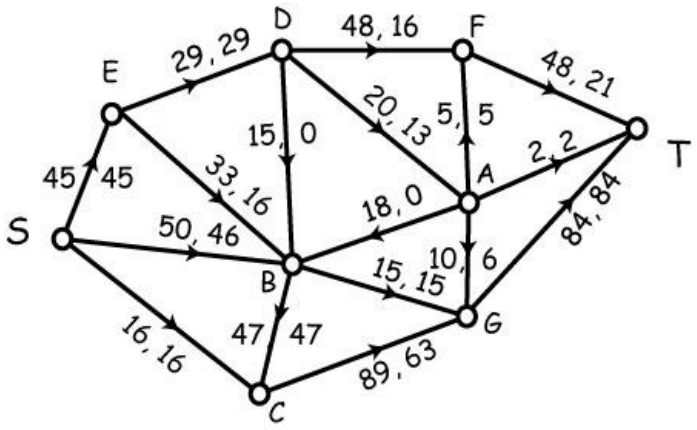
Observation If there are n vertices, there are 2^{n-2} cuts so even if we could evaluate them all to find the minimum capacity of a cut, we would gain no information about the details of a maximum flow. Remarkably, there is an efficient algorithm for finding a maximum flow and a minimum cut at the same time!

It's All About Augmenting Paths!!



Observation Let **L** denote the set of all vertices X for which there is an augmenting path from S to X, and let **R** be the remaining vertices. Then if e is an edge from L to R, it follows that e is full. On the other hand, if f is an edge from R to L, then f is empty. In this example, **L** = (S, B, E) while the remaining vertices are in **R**. Notice that the edges (E, D), (B, G), (S, C) and (B, C) are full while (B, D) and (B, A) are empty.

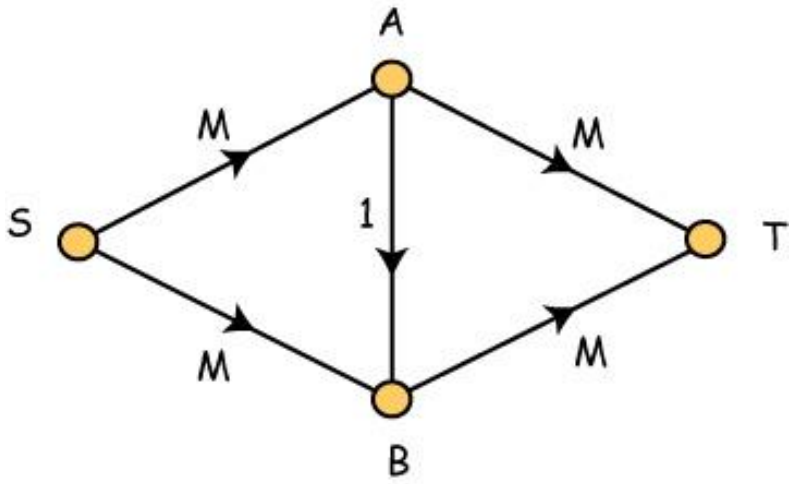
It's All About Augmenting Paths!! (2)



Observation Simple algebra now implies that when all the edges from **L** to **R** are full and all the edges from **R** to **L** are empty, then the value of the current flow is equal to the capacity of the cut (L, R).

Observation So all we have to do is keep finding augmenting paths and increase the flow each time we find one. Eventually, there will be no augmenting paths and we are done.

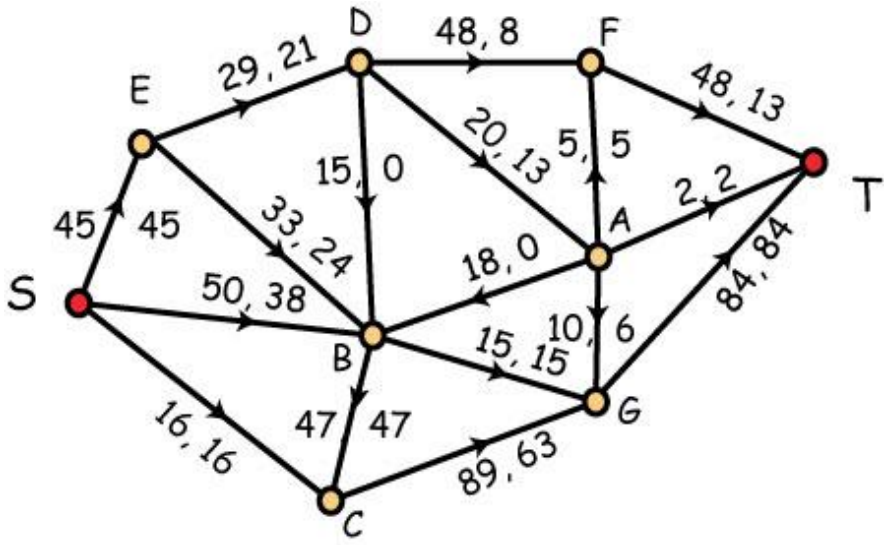
Augmenting Paths – Be Careful



Remark In the figure, suppose M = 1,000,000,000 and we advance from the zero flow using the longest augmenting path we can find. Do you see that it takes 2M steps to find the maximum flow? Furthermore, if you take shortest augmenting paths, you get to the final answer in 2 steps.

Remark This example suggests that we should focus on finding augmenting paths using the minimum number of edges.

Primal Dual with Dijstra as the Dual



Strategy Use Dijkstra to find an augmenting path using the minimum number of edges. Normally, this is done by carrying out the Ford-Fulkerson labelling algorithm. This algorithm assigns triples to vertices of the network, starting with a labelling of the source S. If the sink T is labelled, then we will have found an augmenting path using the minimum number of edges.

Details on the Labelling Scheme

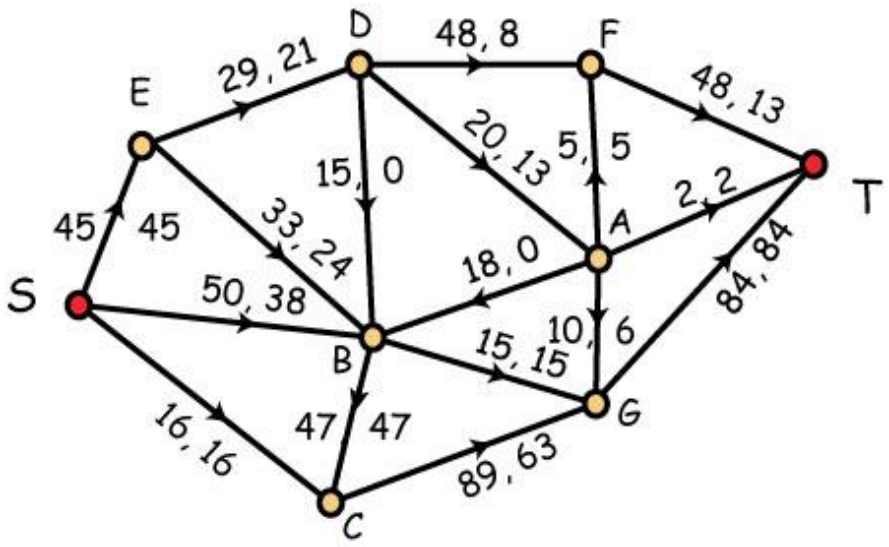
Initialize Label the source S with the triple (*, +, ∞).

Scan Using the rule: first labelled, first scan, let X be a labelled vertex and let a(X) be the positive amount on X. Scan the neighbors on X in pseudo-alphabetic order (S, T, A, B, C, D, E, F, …). If Y is scanned an the edge is (X, Y), i.e., the edge is oriented from X to Y, and it is not full, let a(X, Y) be the spare capacity. Label Y as (X, + , min { a(X), a(X, Y)).

If the edge is (Y, X), i.e., oriented from Y to X, and the edge is not empty, let a(Y, X) denote the flow on this edge. Then label Y as (X, -, min{ a(x), a(Y, X) }).

Halt if the sink T is labeled, as backtracking will specify an augmenting path using the minimum number of edges.

Primal Dual with Dijstra as the Dual



Ford-Fulkerson

S (*, +, ∞)

B (S, +, 12)

E (B, -, 9)

D (E, +, 8)

A (F, +, 7)

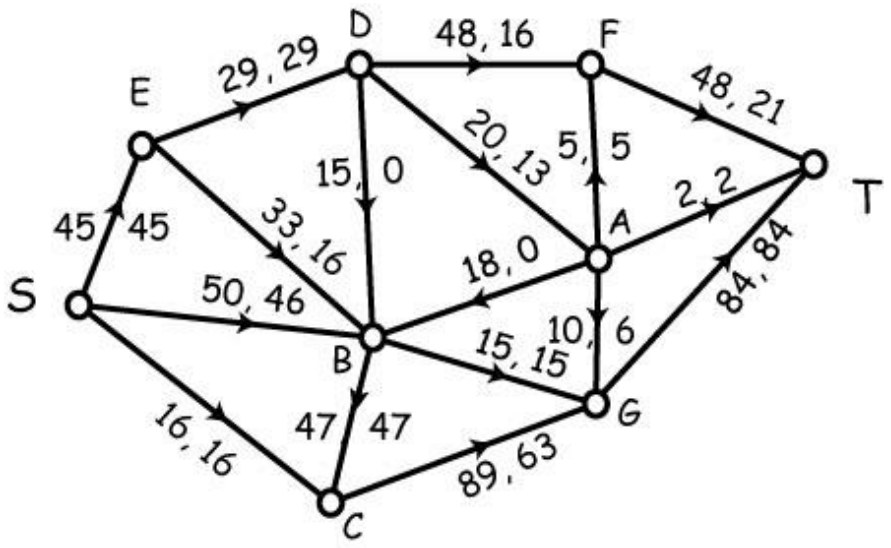
F (D, +, 8)

G (A, +, 4)

T (F, +, 8)

Augmenting Path (S, B, E, D, F, T) with value 8.

The Example Updated



Ford-Fulkerson

S (*, +, ∞)

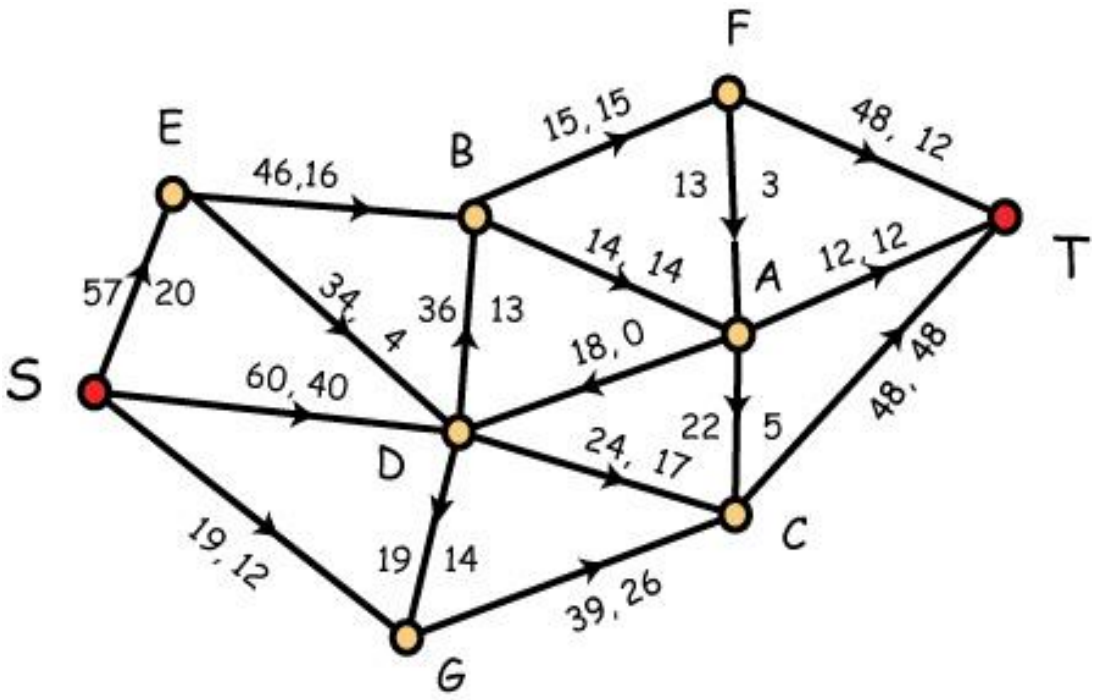
B (S, +, 4)

E (B, -, 4)

The algorithm halts with **S** = { S, B, E }. The remaining vertices belong to **R**.

Conclusion The current flow is maximum and the cut (**L**, **R**) is minimum. Done!

A Second Example



Exercise Carry out the Ford-Fulkerson labelling algorithm on the network flow.

Network Flows and Linear Programming

Definition A problem in n variables $x_1, x_2, \ldots, x_n$ is called a linear programming (LP) problem when it has the form:

Maximize $c_1x_1 + c_2x_2 + \ldots c_nx_n$

subject to m constraints of the form:

$a_{i1}x_1 + a_{i2}x_2 + \ldots a_{in}x_n \leq b_i$

With all variables $x_1, x_2, \ldots, x_n$ non-negative.

Example Maximize $5x_1 + 7x_2$

subject to $14x_1 + 3x_2 \leq 42$

$5x_1 + 9x_2 \leq 45$

$x_1, x_2 \geq 0$

Some Observations on LP problems

Fact There are many equivalent formulations of the class of LP problems. In the set of constraints, you can allow equations and inequalities in the opposite direction. Also you call allow some or all of the variables to be negative.

Fact The solution space to the set of constraints forms a convex body in $\mathbf{R}^n$ with a bounded number of extreme (corner points). The maximum value always occurs and an extreme point. The number of extreme points can be exponentially large in terms of the number n of variables.

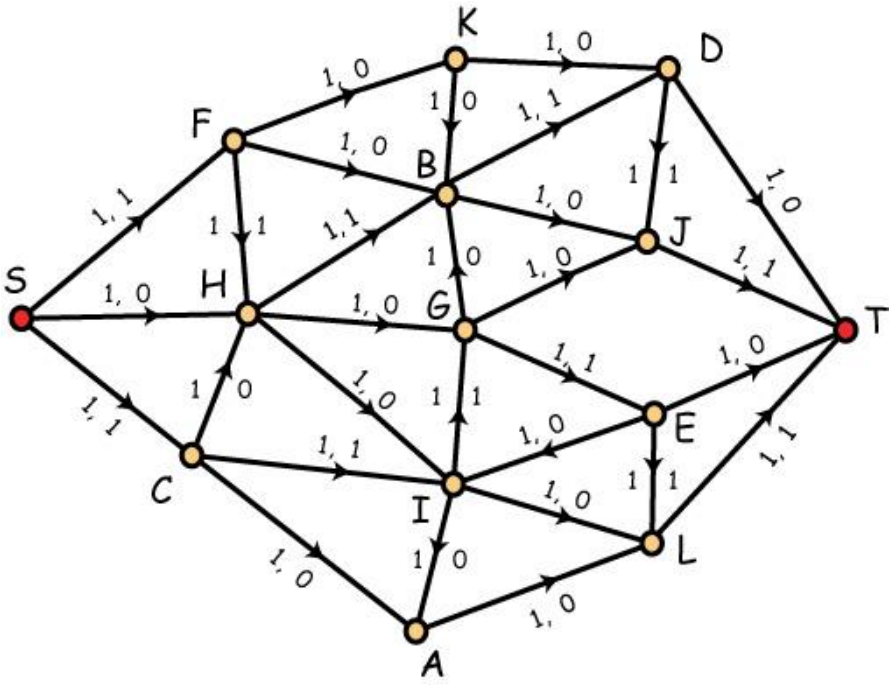
Fact LP problems posed with integer constraints have solutions in the rational number system, but in general they do not have solutions with all variables being integers.

A Key Detail on Network Flow Problems

Fact A network flow problem posed with integer capacities on edges always has a maximum flow in which the flow on every edge is an integer. The proof of this fact is an immediate consequence of the fact that the Ford-Fulkerson labelling algorithm uses only addition, subtraction and minimum as its three operations.

Remark It is an important general problem to determine when optimization posed with integer constraints have integer valued solutions. Network flows are just one example.

Network Flow Problems with Capacities 1



Exercise Carry out the Ford-Fulkerson labelling algorithm on the network flow.