CS 491 G Combinatorial Optimization

Lecture Notes

Junhui Jia

1. Maximum Flow Problems

Now let us discuss more details on maximum flow problems.

Theorem 1.1

A feasible flow $\vec{x}$ is maximum if and only if there is no $\vec{x}$-augmenting path.

Proof:

Let P = "A feasible flow is maximum" and Q = "there is no $\vec{x}$-augmenting path". It is obvious that $P \Rightarrow Q$. For the case of $Q \Rightarrow P$ using the construction of proof of Max-Flow and Min-Cut Theorem, we get a cut $\delta(R)$ with $f_{\vec{x}}(s) = u(\delta(R))$, then combine the Corollary of $f_{\vec{x}}(s) \leq u(\delta(R))$, we know $\vec{x}$ is maximum. So Theorem is proved.

Theorem 1.2

If $\vec{u}$ is integral and there exists a maximum flow, then there exists a maximum flow that is integral.

Proof:

Set a feasible flow, which is an integral flow of maximum value. From theorem 1.1, if there is no $\vec{x}$-augmenting path, $\vec{x}$ is a maximum flow. Suppose there is an $\vec{x}$-augmenting path, the contradiction is that $\vec{x}$ is not an integral flow of maximum value, because the new flow could be integral when $\vec{x}$ and $\vec{u}$ are integral. Hence there is no $\vec{x}$-augmenting path, and $\vec{x}$ must be a maximum flow. So Theorem 1.2 is proved.

Corollary 1.3

If $\vec{x}$ is a feasible (r, s)-flow and $\delta(R)$ is an (r, s)-cut, then $\vec{x}$ is maximum and $\delta(R)$ is minimum if and only if

$$x_e = u_e, \text{ for all } e \in \delta(R) \text{ and } x_e = 0, \text{ for all } e \in \delta(\overline{R}).$$

2. The Augmenting Path Algorithm

2.1 The Ford-Fulkerson procedure

1. Find an $\vec{x}$-augmenting path *P*.
2. Increment flow $\vec{x}$ along the path *P*.

The Ford-Fulkerson Algorithm provides a tool to find a maximum flow and a minimum cut. When we execute the step 2 of the Ford-Fulkerson procedure, there is a maximum value permitted. It is $\varepsilon = \min(\varepsilon_1, \varepsilon_2)$, we call ε the $\vec{x}$-width of *P*. That is:

The maximum value permitted = $\min(\varepsilon_1, \varepsilon_2)$
where
$\varepsilon_1 = \min(u_e - x_e \ : \ e$ forward in $P)$
$\varepsilon_2 = \min(x_e \ : \ e$ reverse in $P)$

2.2 The Auxiliary or Residual Graph *G*($\vec{x}$)

Definition: 2.1 Given a graph $G = \langle V, E, u \rangle$ and current flow $\vec{x}$, we define $G(\vec{x})$ to be the auxiliary graph, when put $V(G(\vec{x})) = V$ and $vw \in E(G(\vec{x}))$ if and only if $vw \in E$ and $x_{vw} < u_{vw}$ or $wv \in E$ and $x_{wv} > 0$.

If for $vw \in E$, both $x_{vw} < u_{vw}$ and $x_{wv} > 0$ are satisfied, then put two parallel arcs into $E(G(\vec{x}))$. It is noticed that an auxiliary graph is un-weighted. An example is given in Figure 1.1.

Observation

A (*r*, *s*) dipath in $G(\vec{x})$ is an $\vec{x}$ augmenting path.

Defining the auxiliary graph supplies us a method for searching augmenting paths. It is obvious as well that:

$$|E(G(\vec{x}))| \leq 2|E|$$

Because there are only possibility of forward and reverse edges when you obtain an auxiliary graph $G(\vec{x})$ from a graph *G*, the number of edges in an auxiliary graph $G(\vec{x})$ must be at most twice of that of original graph *G*.

In order to implement the augmenting path, create $G(\vec{x})$. This can be done in $O(m)$ time having a path from r to s with Breadth-first search method.

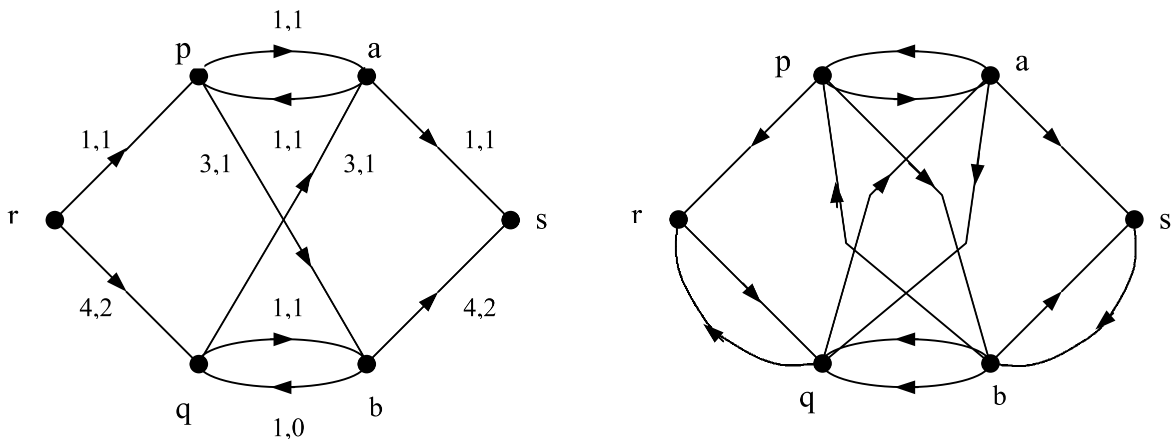


Figure 1.1 A feasible flow of value 3 and its auxiliary digraph $G(\vec{x})$

Notation: In figure 1.1, the left hand number stands for the capacity of that edge and the right hand number stands for the flow through that edge.

However, the augmenting path algorithm cannot be considered acceptable. Figure 1.2 shows a bad example for this algorithm. If every augmenting path uses arc *ab*, then each augmentation will be of value 1, the algorithm will never terminate, and there is no maximum flow.

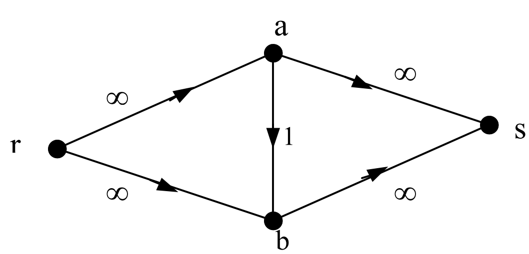


Figure 1.2. A bad Example for the augmenting path algorithm

The running time

For m edges and n vertices,

Uniform cost model:

$$\text{Input size} = m + n + m \le 3m$$

It is unbounded because the ratio = $\dfrac{\|f_{\max}\|m}{3m}$ is independent of m. The total time taken = $\|f_{\max}\|m$

Log cost model:
Let every edge has capacity of $\|f_{\max}\|$, then we have

$$\text{Input size} = m + n + m\log\|f_{\max}\| \le 3m\log\|f_{\max}\|$$
$$3m\log\|f_{\max}\| \to m\|f_{\max}\|$$

where $m\|f_{\max}\|$ is the running time, choose $\|f_{\max}\| = 2^m$, then we have

$$3m^2 \to m2^m \Rightarrow O\left(\frac{2^m}{m}\right)$$

It is an exponential function.

3. Edmonds-Karp Idea

3.1 Introduction to the Shortest $\vec{x}$-augmenting path

Definition: 3.1 A shortest augmenting path is an $\vec{x}$-augmenting path which has the minimum possible number of arcs.

Definition: 3.2 $d_{\vec{x}}(u,v)$ is the length of the shortest path from u to v in $G(\vec{x})$.

Consider a typical augmentation from flow $\vec{x}$ to $\vec{x}'$ determined by augmenting path P, which have node sequence as $v_0, v_1, \ldots, v_k$. Let $d_{\vec{x}}(r,v)$ be the least length path from r to v in $G(\vec{x})$ of a (r, v) dipath, then we have $d_{\vec{x}}(r,v_i) = i$ and $d_{\vec{x}}(v_i,s) = k - i$. If an arc vw of $G(\vec{x}')$ is not an arc of $G(\vec{x})$, then $v = v_i$, $w = v_{i-1}$ for some i.

Question: How to implement in $O(m)$ time?

Breadth-first search will accomplish it in $O(m)$ time. So that it is easier to find a shortest augmenting path than to find an augmenting path. This gives us a polynomial-time algorithm for the maximum flow problem.

Lemma 3.2

For each $v \in V$, we have that

$$d_{\vec{x}'}(r,v) \geq d_{\vec{x}}(r,v).$$

Proof: Let a node *v* be a vertex for which the lemma is false. That is

$$d_{\vec{x}'}(r,v) < d_{\vec{x}}(r,v)$$

Choose the node *v* such that and *v* is the vertex for which $d_{\vec{x}'}(r,v)$ is the smallest all such "violating" vertices. Let P' be a (*r*, *v*) dipath in $G(\vec{x}')$ of length $d_{\vec{x}'}(r,v)$ and let *w* be the node before *v*. Then

$$d_{\vec{x}}(r,v) > d_{\vec{x}'}(r,v) = d_{\vec{x}'}(r,w) + 1 \geq d_{\vec{x}}(r,w) + 1$$

Now We can conclude that *wv* is not an arc of $G(\vec{x})$ since if *wv* is an arc of $G(\vec{x})$, then

$$d_{\vec{x}}(r,v) \leq d_{\vec{x}}(r,w) + 1$$

Suppose there is an arc in $G(\vec{x}')$ but not in $G(\vec{x})$, then the arc must have been reversed in $G(\vec{x})$. We have

$$w = v_i \text{ and } v = v_{i-1}$$

A contradiction comes up immediately since

$$i - 1 \geq i + 1$$

So the Lemma 3.2 is proved. Likewise, we can prove that

$$d_{\vec{x}'}(v,s) \geq d_{\vec{x}}(v,s)$$

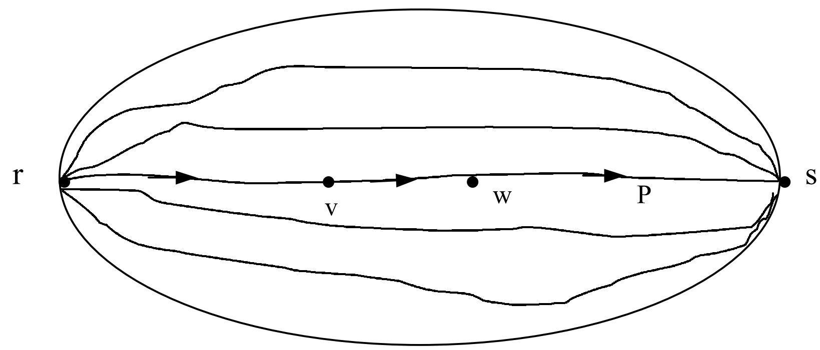
Lemma 3.3

If $d_{\vec{x}'}(r,s) = d_{\vec{x}}(r,s)$,
Let $\widetilde{E}(x)$={$e \in E$: *e* is an arc of a shortest $\vec{x}$-augmenting path},
Then $\widetilde{E}(\vec{x}') \subset \widetilde{E}(\vec{x})$.

Proof: Pick some edges that satisfy $e \in \widetilde{E}(\vec{x}')$ and

$$e \in \widetilde{E}(\vec{x})$$

Let $d_{\vec{x}}(r,s) = k$, e induces an arc *vw* in $G(\vec{x}')$ and $d_{\vec{x}'}(r,v) = i-1$, $d_{\vec{x}'}(w,s) = k-i$ for some *i*, where $v = v_{i-1}$ and $w = v_i$, as shown in Figure 3.1.



$G(\vec{x}')$

Figure 3.1 Illustration for the proof of $\widetilde{E}(\vec{x}') \subseteq \widetilde{E}(\vec{x})$

Notation: In figure 3.1, the only edge that can have capacities changed arc must along path P. The other paths are exactly the same as those of $G(\vec{x})$.

Therefore, $d_{\vec{x}}(r,v) + d_{\vec{x}}(w,s) \leq k-1 = d_{\vec{x}'}(r,v) + d_{\vec{x}'}(w,s) = (i-1)+(k-i)$, Suppose this ($e \in \widetilde{E}(\vec{x})$) is not true, then $\vec{x}_e \neq \vec{x}_e^{'}$, but $e \in P$ in $G(\vec{x})$. This is a contradiction. Therefore, every edge in $\widetilde{E}(\vec{x}')$ must be in $\widetilde{E}(\vec{x})$, that is

$$\widetilde{E}(\vec{x}') \subseteq \widetilde{E}(\vec{x})$$

Now let's prove $\widetilde{E}(\vec{x}')$ is the proper subset of $\widetilde{E}(\vec{x})$. Consider the edge $e: vw$ that is saturated. If it must be used on another shortest path, the direction must be reversed at what it is on $G(\vec{x})$. It is illustrated in Figure 3.2. what we need to find out is the critical edge of the path P, the length of any path using it is at least k+2 and it does not belong to $\widetilde{E}(\vec{x}')$. Then the Lemma 3.3 would be proved.

It is obvious that for the original graph we have

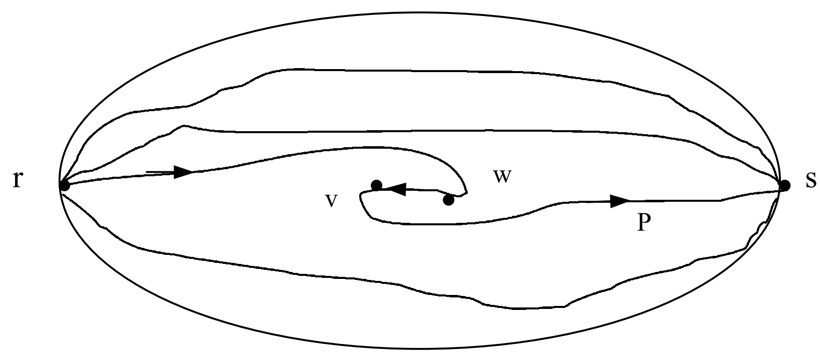
$d_{\vec{x}}(r,v) = i-1$, $d_{\vec{x}}(w,s) = k-i$
and
$d_{\vec{x}}(r,w) = i \Rightarrow d_{\vec{x}'}(r,w) \geq i$
$d_{\vec{x}}(v,s) = k-i+1 \Rightarrow d_{\vec{x}'}(v,s) \geq k-i+1$

So the length of P will be at least

$$i + k - i + 1 + 1 = k + 2$$

where $P = \langle V_0 = r, v_1, \cdots v = v_i, w = v_{i+1}, \cdots s = v_k \rangle$, $d_{\vec{x}}(r, w) = i + 1$, $d_{\vec{x}}(v, s) = k - i$, $d_{\vec{x}}(r, s) = k$.

So Lemma 3.4 is proved.



$G(\vec{x}')$

Figure 3.2 Illustration for the proof of $\widetilde{E}(\vec{x}')$ is the proper subset of $\widetilde{E}(\vec{x})$.

References

1. William Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization*. John Wiley & Sons, INC., 1998.