The Goldberg–Rao Algorithm for the Maximum Flow Problem

COS 528 class notes – October 18, 2006

Scribe: Dávid Papp

Main idea: use of the blocking flow paradigm to achieve essentially $O(\min\{m^{2/3}, n^{1/2}\} \cdot m)$ running time (with some additional logarithmic factors) not only for unit capacity simple networks (for which Dinitz's algorithm achieves precisely this bound), but for general networks. The improvement is achieved by considering a carefully selected binary length function. The running time we prove is precisely $O(\min\{m^{2/3}, n^{1/2}\} \cdot m \log n \log nU)$ (where U is the largest arc capacity in the network), better analysis and more advanced data structures yield $O(\min\{m^{2/3}, n^{1/2}\} \cdot m \log \frac{n^2}{m} \log U)$.

1 Preliminaries

We are using the following notation: the network is a directed graph $G = (V, E)$ with a source node s and a sink node t. The capacity $u(a)$ of every arc a is an integer between 1 and U. The network has n nodes and m arcs; to simplify notation we define the shorthand $\Lambda := \min\{m^{2/3}, n^{1/2}\}$.

There is a binary **length** assigned to every arc. A function $d : V \to \mathbb{Z}^+$ is a **distance labeling** with respect to the length function l if $d(t) = 0$ and every $(v, w) \in E$ satisfies $d(v) \leq d(w) + l(v, w)$. If for a *residual* arc this is satisfied with equality, that is, $d(v) = d(w) + l(v, w)$, then the residual arc (v, w) is called **admissible**. (This is equivalent to saying that $d(v) > d(w)$ or $d(v) = d(w)$ and $l(v, w) = 0$, because the length function is binary.) We are going to augment flows by finding blocking flows in the **network of admissible arcs** with respect to the length function l and distance labels d, denoted by $A(f, l, d)$, where f is the current flow.

In accordance with the original blocking flow idea, in order to get a good running time bound, we need to ensure that the algorithm terminates after not too many augmentations by showing that every iteration increases the distance of the source and the sink. This is more difficult with a binary length function than with the usual unit length function, due to the presence of zero length arcs. To overcome this difficulty, two modifications are introduced.

First, the algorithm contracts the strongly connected components of the subnetwork formed by zero-length arcs. To make sure that the flow that is directed through a node corresponding to such a contracted component can be routed through the restored component as well, the algorithm bounds from above the size of an augmenting flow by some value Δ, and bounds from below the capacity of zero length arcs. Consequently the length function $\bar{l}$, which determines the admissible arcs and the corresponding distance labels, $d_{\bar{l}}$, will have to be updated after every augmentation.

The second introduced modification is a necessary consequence of the previous one: if the augmenting flow has a bounded value, it is possible that we do not find a blocking flow of suitable value. Then we cannot guarantee the increase of the distance of s and t, and hence Δ has to be chosen in a way which ensures that the number of such augmentations is small.

The final bit of the algorithm is a stopping condition: we iteratively increment the flow until some upper bound F of the difference of the maximal and the current flow decreases below 1. By the integrality of the capacities this implies that the current flow is maximal. Initially, for the zero flow a trivial estimate is $F = nU$.

The skeleton of the Goldberg–Rao algorithm now can be described as follows:

```
while $F \geq 1$ do
    update the parameter $\Delta$, the length function $\bar{l}$, and distance labels $d_{\bar{l}}$;
    contract the strong components of the network formed by the zero length arcs;
    determine the network of admissible arcs, $A(f, \bar{l}, d_{\bar{l}})$;
    find a flow in $A(f, \bar{l}, d_{\bar{l}})$ which is either blocking or has a value $\Delta$;
    augment the current flow by the flow found in the previous step;
    update $F$;
endwhile
```

In the following we discuss how to set the parameters $\bar{l}$ and Δ, and how to maintain F in order to ensure the desired running time.

2 Estimating the residual flow

The residual capacity $u_f(S, T)$ of any s-t cut (S, T) can serve as an upper bound on the residual flow (the maximal flow in the residual network corresponding to the current flow). One way to obtain such a bound fast is to consider the **canonical cuts** only, ie. cuts of the form (S_k, T_k), where $S_k := \{v \in V \mid d(v) \geq k\}$, $T_k := V \setminus S_k$.

Lemma. *The minimum capacity canonical cut can be found in $O(m)$ time.*

Proof. Since every arc has length at most one, every arc may cross at most one canonical cut. We can use this fact to compute the capacities of the canonical cuts in the following way. Initialize every $u_f(S_k, T_k)$ to be zero. Then for every arc (v, w) if $d(v) > d(w)$, then increase $u_f(S_{d(v)}, T_{d(v)})$ (the capacity of the only canonical cut crossed by (u, v)) by $u_f(v, w)$. Finally, find the minimum among the numbers $u_f(S_k, T_k)$, and set F to this value. Both the correctness and the linear running time is immediate. □

To obtain a running time bound we will make sure that F will decrease fast enough. A sequence of operations constitutes a *phase* of the algorithm if it decreases F by a factor of two. To simplify the analysis we do not change F in every iteration. Instead, we change F to the residual capacity of the minimum canonical cut only when the latter decreases to $F/2$ or below, marking the end of the phase. Clearly, after at most $\log nU$ phases the algorithm terminates.

3 Binary blocking flow algorithm

The blocking flow algorithm used in the Goldberg–Rao algorithm is a variant of the original blocking flow algorithm. Changes are necessary to ensure that after a blocking flow iteration either the distance of s and t strictly increases (in spite of the zero length arcs), or the current flow is increased significantly enough.

As we have mentioned in the introduction, we will have to bound how much we can increment the current flow in one iteration, and we denoted this bound by Δ. To make the initially claimed running time possible, the number of updates by non-blocking flows cannot exceed Λ per phase, hence we can set $\Delta = \lceil F/\Lambda \rceil$.

The binary length function for a phase is based on the following definition:

$$l(v,w) = \begin{cases} 0 & u_f(v,w) \geq 3\Delta \\ 1 & \text{otherwise} \end{cases},$$

that is, arcs of very large capacity have zero length. To ensure that the distance of s and t increases after augmenting along a blocking flow we need to modify this in the following way. Let us call an arc (v,w) **special** if it satisfies the following:

- $2\Delta \leq u_f(v,w) < 3\Delta$,
- $d(v) = d(w)$,
- $u_f(w,v) \geq 3\Delta$,

and let us define the binary length function we are going to use as follows:

$$\bar{l}(v,w) = \begin{cases} 0 & u_f(v,w) \geq 3\Delta \text{ or } (v,w) \text{ is special} \\ 1 & \text{otherwise} \end{cases}.$$

Note that the definition of special arcs does not change the distances: $d_l = d_{\bar{l}}$.

We can now give the details on how we handle the contraction of zero length arcs.

Lemma. *Suppose that we have found a flow of value at most Δ in the contracted network. Then we can correct this to a flow of the same value in the original network in $O(m)$ time.*

Proof. Choose an arbitrary vertex in each component as its root. Form an in-tree and an out-tree of each component rooted at their respective roots. Route all the positive balances towards the root using the in-tree arcs, and then route the resulting flow excess from the root to the negative balances using the out-tree arcs. During this process we do not violate any arc capacity constraints, since each zero length arc has capacity at least 2Δ, and at most Δ flow is routed both to and from the root using that arc. The process clearly takes linear time in the number of zero length arcs. □

The following theorem is crucial to obtain the desired running time.

Theorem. *Let $\bar{f}$ be a flow in $A(f, \bar{l}, d_l)$, let $f' = f + \bar{f}$ be the augmented flow, and let l' be the length function corresponding to f'. Then*

1. d_l is a distance labeling with respect to l',
2. $d_{l'}(s) \geq d_l(s)$,
3. if $\bar{f}$ is a blocking flow, then $d_l(s) < d_{l'}(s)$,

Remark. Notice that in the theorem $\bar{l}$ is the modified distance function, which takes special edges into consideration, while l' is the "basic" length function, which assigns an arc length of 1 to special arcs. In the original paper Theorem 4.3 contains a few typos, which confuse the different length functions, making both the theorem and the proof incorrect.

Proof of Theorem.

1. By definition of the distance labeling, $d_l(v) \leq \bar{l}(v,w) + d_l(w)$ is given, and we need to show $d_l(v) \leq l'(v,w) + d_l(w)$. This is trivial if $d_l(v) \leq d_l(w)$. If $d_l(v) > d_l(w)$, which is the same as saying that $d_{\bar{l}}(v) > d_{\bar{l}}(w)$ (recall that the two distance functions are the same), then (w,v) is not admissible with respect to $\bar{l}$. Consequently $u_{f'}(v,w) \leq u_f(v,w)$, which implies $l'(v,w) \geq \bar{l}(v,w)$. The statement then follows.

2. We prove that any distance labeling d is a lower bound on the actual shortest path distances d_l from the sink. This, together with the previous part of the theorem, implies our claim.

   Suppose otherwise, ie. let $d(v) > d_l(v)$ for some node v, and let w be the last node on a shortest v–t path Γ which satisfies $d(w) > d_l(w)$, and let x be the node that follows w on Γ. (It may be that $w = v$, but certainly not $w = t$, and hence x is well-defined.) Then $d_l(w) < d(w) \leq l(w,x) + d(x) \leq l(w,x) + d_l(x)$, and hence the v–w subpath of Γ concatenated with the the shortest w–t path is shorter than Γ, contradicting to the selection of Γ.

3. By the first part of the theorem, $c(v,w) := d_l(w) - d_l(v) + l'(v,w) \geq 0$. Summing these up along an s–t path Γ in $G_{f'}$ (the residual graph corresponding to f'), the d_l terms telescope, and $l'(\Gamma) = d_l(s) + c(\Gamma)$, where $l'(\Gamma)$ and $c(\Gamma)$ are shorthands for summation of l' and c over the arcs of the path. Consequently it is enough to show that along every s–t path Γ in $G_{f'}$ there is an arc (v,w) satisfying $c(v,w) > 0$.

   Since $\bar{f}$ is blocking in $A(f,\bar{l},d_l)$, Γ must contain an arc (v,w) which is not in $A(f,\bar{l},d_l)$. Then $d_l(v) \leq d_l(w)$, either because (v,w) is in G_f, and $d_l(v) > d_l(w)$ would mean that we have to include (v,w) to $A(f,\bar{l},d_l)$, or because (v,w) is not in G_f, but it appears in $G_{f'}$, which means that (w,v) is in $A(f,\bar{l},d_l)$, implying $d(w) \geq d(v)$.

   Suppose now by contradiction that $c(v,w) = 0$. Then $d_l(v) = d_l(w)$ and $l'(v,w) = 0$. The fact that (v,w) is not in $A(f,\bar{l},d_l)$ implies that either (v,w) is not in G_f (in which case we already showed that (w,v) is in $A(f,\bar{l},d_l)$), or (v,w) is in G_f, and then $l(v,w) = 1$. In the latter case, $1 = l(v,w) > l'(v,w) = 0$ implies that the reverse arc, (w,v), is in $A(f,\bar{l},d_l)$.

   We concluded that in any case, the arc (w,v) is included in $A(f,\bar{l},d_l)$. But since $d_l(w) = d_l(v)$, this can happen only if $l(w,v) = 0$. This means that during the augmentation we pushed flow (of value at most Δ) through (w,v), which already satisfied $u_f(w,v) \geq 3\Delta$, and this increased the residual capacity of (v,w) to at least 3Δ. (Because $l'(u,w) = 0$.) So $u_f(v,w) \geq 2\Delta$, and this together with $u_f(w,v) \geq 3\Delta$ and $d(v) = d(w)$ shows that (v,w) was a special arc before the augmentation.

   We conclude that (v,w) satisfies both $d_{\bar{l}}(v) = d_{\bar{l}}(w)$ (from $d_l(v) = d_l(w)$) and $\bar{l}(v,w) = 0$, and hence (v,w) is in $A(f,\bar{l},d_l)$, a contradiction. □

4 Time bounds

4.1 Time bounds on the blocking flow algorithm

Only one thing is missing to complete the proof of the claimed running time: we have to show that the number of augmentations where we augment along a blocking flow is also

$O(\Lambda)$ per phase, just like the number of augmentations by a flow of value Δ. (The latter is immediate, Δ is defined to satisfy this.) It is easy to see that the following is true.

Lemma. *For a binary length function d_l the capacity of the minimum capacity canonical cut, $(\bar{S}, \bar{T})$ satisfies*

$$u_f(\bar{S}, \bar{T}) \leq \frac{mM}{d_l(s)},$$

where M is the maximum capacity of a length one arc.

Proof. Since every arc has length at most one, each crosses at most one of the canonical cuts. The total capacity of arcs is at most mM, and there are $d_l(s)$ canonical cuts, so the minimum capacity canonical cut has capacity at most $mM/d_l(s)$. □

For dense graphs the following lemma gives a better bound.

Lemma. *With the notations of the previous lemma,*

$$u_f(\bar{S}, \bar{T}) \leq \left(\frac{2n}{d_l(s)}\right)^2 M.$$

Proof. Let $V_k := \{v \in V \mid d(v) = k\}$. Similarly to the previous proof, it is enough to show that there is a k such that both V_k and V_{k+1} have at most $2n/d_l(s)$ elements. But this follows from the pigeonhole-principle: $\sum_{k=0}^{d_l(s)} |V_k| = n$, hence $V_k \leq 2n/d_l(s)$ for at least $\lceil d_l(s)/2 \rceil + 1$ values of k, so there are two consecutive k's with this property. □

Using that during a phase of the Goldberg–Rao algorithm $M \leq 3\Delta$ by definition, we obtain that

Corollary. *During a phase of a binary blocking flow algorithm there are at most $O(\Lambda)$ blocking flow augmentations.*

Proof. Suppose $\Lambda = m^{1/2}$. Every blocking flow augmentation increases $d_l(s)$ by at least one, so after $6\lceil\Lambda\rceil$ such update steps $d_l(s) \geq 6m^{1/2}$, and

$$u_f(\bar{S}, \bar{T}) \leq \frac{mM}{d_l(s)} \leq \frac{3m}{d_l(s)}\Delta \leq \frac{3m}{6m^{1/2}} \frac{F}{m^{1/2}} = \frac{F}{2},$$

thus after this many steps the phase terminates, regardless of the augmentations by flows of value Δ. The case $\Lambda = n^{2/3}$ is similar, $5\lceil\Lambda\rceil$ augmentations are enough to terminate the phase. □

4.2 Bound on the running time of the Goldberg–Rao algorithm

Theorem. *The maximum flow problem can be solved in $O(\Lambda m \log(n^2/m) \log nU)$ time by the Goldberg–Rao algorithm.*

Proof. We have seen that the number of phases is $O(\log nU)$. In every phase there are two kinds of augmentations: along blocking flows, and along flows of value Δ. We have proven that the number of both is $O(\Lambda)$. The bottleneck in one iteration is finding a blocking flow or a flow of value Δ. This is of the same complexity as finding a blocking flow, which is $O(mn)$ with naive implementation, $O(m \log n)$ using dynamic trees, and $O(m \log(n^2/m))$ using size-bounded dynamic trees. (Every other step is $O(m)$.) Multiplying these results in the claim. □