COMPSCI 530: Design and Analysis of Algorithms September 10, 2013

/

Lecture 5

Lecturer: Debmalya Panigrahi *Scribe: Yuzhang Han*

1 Overview

Push-Relabel algorithm and Scaling algorithms are introduced. Examples and running time analysis of the algorithms are described.

2 Algorithm and Main Operations

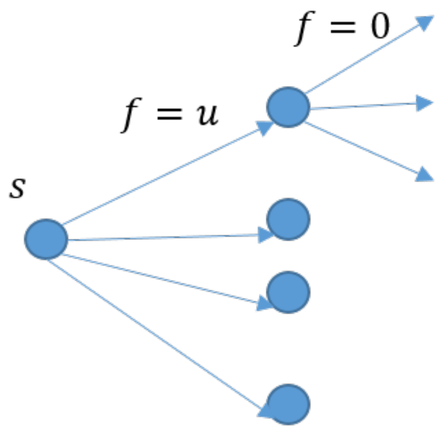
In this section we will introduce the main operations and data structures employed by push-relabel algorithms, as well as how these algorithms operate. Push-relabel algorithms are used to compute maximum flows of networks. They process one vertex at a time, examining the vertexs neighbors in the residual network. They maintain a *preflow*, which is a function in positive real numbers defined as follows: [Cor09]

Definition 1 (Preflow)**.** $f : (u,v) \to R_+$

1. $f(u,v) \le c(u,v)$

2. $for \forall u \in V \backslash \{s,t\}\ e(u) = \sum_{v \in V} f(v,u) - \sum_{v \in V} f(u,v) \ge 0$

We call $e(u)$ the *excess flow* into vertex u. It is the difference between flow in and flow out at a vertex.



The *residual network* is defined as before [Cor09]:

Definition 2 (Residual Network)**.** *A residual network G_f is such that*

- $c_f(u,v) = c(u,v) - f(u,v)$
- $c_f(v,u) = c(v,u) + f(v,u)$

Invariant: $\nexists s-t$ path in residual network.

A push-relabel algorithm performs two main operations iteratively: either *push* flow excess from a vertex of the graph to one of its neighbors, or *relabeling* a vertex. Operation choosing depends on the *heights* of the vertices, which is a function in positive integer defined as follows [Cor09]:

Let f be a preflow on network $G = (V,E)$ with source s and sink t. Function $h : V \to N$ is the height function if

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 for \forall (u,v) \in E_f$.

Then, we move on to introduce the main operations.

PUSH

$PUSH(u,v)$ can be performed if $c_f(u,v) > 0$ (called u is *overflowing*.) It reduces $min(e(u), c_f(u,v))$ from $e(u)$, the excess flow of u and adds this value to $e(v)$ and $f(u,v)$, the flow on (u,v). It is represented by the following code [Cor09]:

Algorithm 1 PUSH(u, v)

```
1: $\Delta_f(u,v) = min(e(u), c_f(u,v))$
2: if $(u,v) \in E$ then
3:     $(u,v).f = f(u,v) + \Delta_f(u,v)$
4: else$(v,u).f = f(v,u) - \Delta_f(u,v)$
5: end if
6: $e(u) = e(u) - \Delta_f(u,v)$
7: $e(v) = e(v) + Delta_f(u,v)$
```

A push is called *saturating push* if the edge (u,v) it was applied on in the residual network becomes saturated, that is, $c_f(u,v) = 0$. Otherwise, it is called a *nonsaturating* push.

RELABEL

$RELABEL(u)$ can be performed if $c_f(u,v) > 0$ and $h(u) \leq h(v)$ for all $(u,v) \in E_f$. It sets the height of u to be the height of the lowest neighbor plus 1. It is represented by the following code [Cor09]:

Algorithm 2 RELABEL(u)

```
1: $h(u) = min(h(v) : (u,v) \in E_f) + 1$
```

The generic push-relabel algorithm performs $PUSH$ and $RELABEL$ in the following manner:

Algorithm 3 PUSH-RELABEL

```
1: $INIT - PREFLOW(G)$
2: $INIT - HEIGHT(G)$
3: while Any $PUSH$ or $RELABEL$ is able to be performed do
4:     perform the $PUSH$ or $RELABEL$
5: end while
```

Algorithm 4 INIT-PREFLOW(G)

```
1: $for \forall u and v, f(u,v) = 0 and f(s,v) = c(s,v)$
```

Algorithm 5 INIT-HEIGHT(G)

```
1: $for \forall u, h(u) = 0; h(s) = |V|$
```

3 Complexity

Now that we have understand the basic operations and steps of push-relabel algorithms, we can analyze their complexity. We will first prove a series of lemmas, and finally come to the conclusion that the generic push-relabel algorithm is $O(|V|^2|E|)$ [Cor09].

First, we need to prove the following lemma:

Lemma 1. *At any iteration of PUSH-RELABEL algorithm, $h(u) \leq 2|V| - 1 for \forall u \in V$ [Cor09].*

Proof. To complete the proof, we need to prove another lemma first:

Lemma 2. *If $e_f(v) > 0$, then v has a path in E_f to s [Tre].*

Proof. Induction: Initially, only vertices v such that $(s,v) \in E$ have excess. If a flow is sent from u to v, since there is a path p from u to s by induction, and sending the flow adds an edge (v,u) to E_f, we know that v has excess and $(v,u) \cup p$ is a path from v to s. □

According to the definition of height, $h(s) = |V|$ and $h(t) = 0$. The height of both vertices are not greater than $2|V| - 1$.

Then, observe $u \in V - \{s,t\}$. At the beginning, $h(u) = 0 \leq 2|V| - 1$. After relabeling, u is overflowing. The lemma above ensures that there is a path $p = (u, v_1, ..., v_{d-1}, s)$ from u to s in G_f. By the definition of height, we know that $h(v_i) \leq h(v_{i+1} + 1)$. Then, we have $h(u) = h(s) \leq h(v_d) \leq h(s) + (|V| - 1) = 2|V| - 1$. □

Then, we can continue to find out the bound on saturating pushes.

Lemma 3. *At any iteration, the number of saturating pushes is less than $2|V||E|$ [Cor09].*

Proof. Suppose a saturating push from u to v has been performed. Consequently, $h(v) = h(u) - 1$. Before performing another push from u to v, a push from v to u must be done, which is possible only if $h(v) = h(u) + 1$. Thus, $h(v)$ must increase by at least 2. We know from the lemma above that height is never larger than $2|V| - 1$. That means that any vertex can increase its height by 2 at most $|V|$ times. We know that between two saturating pushes between u and v, height of one of these vertices must be increased by 2. Therefore, there are at most $2|V|$ saturating pushes between these two vertices. For the whole graph, there can be at most $2|V||E|$ saturating pushes. □

Next, we find out the bound on nonsaturating pushes.

Lemma 4. *At any iteration, the number of nonsaturating pushes is less than $4|V|^2(|V| + |E|)$.*

Proof. We use amortized analysis [Cor09]. Use the potential function $\phi = \sum_{v \in V, e(v) > 0} h(v)$. We observe that relabeling u increases ϕ by at most $2|V|$, since the $h(u)$ is at most $2|V| - 1$. At the same time, a saturating push from u to v increases ϕ by at most $2|V|$, since only v can become overflowing and $h(v)$ is at most $2|V| - 1$. Now consider a nonsaturating push from u to v. Since u is not overflowing after the push and it may be overflowing after the push, ϕ has decreased by $h(u)$ and increased by 0 or $h(v)$. From $h(u) - h(v) = 1$ we know that ϕ has decreased by at least 1. From above lemmas we know that the total number of relabelings and saturating pushes is at most $(2|V|)(2|V|^2) + (2|V|)(2|V||E|) = 4|V|^2(|V| + |E|)$, which will be reduced by nonsaturating pushes. Therefore, we have proved the bound of nonsaturating pushes. □

The last step before computing the complexity of the algorithm is finding out the complexity of each individual operation:

Lemma 5. *The running time of any push or relabel operation is $O(1)$ [Cor09].*

Proof. In the assignment we have implemented the algorithm with $O(1)$ running time using lists [Cor09]. □

Finally, we can compute the running time of the generic push-relabel algorithm:

Lemma 6. *The running time of the push-relabel generic algorithm is $O(V^2E)$ [Cor09].*

Proof. Comparing the bounds of numbers of relabeling as well as nonsaturating and saturating pushes, we see that the number of nonsaturating pushes is dominant. Thus, the number of any operations performed by the algorithm is at most $O(V^2E)$. Therefore, the running time of the algorithm is the same value [Cor09]. □

4 Summary

The lecture introduces push-relabel algorithms. Important features and running time analysis of these algorithms are discussed. Using dynamic trees, we can perform unsaturating pushes in parallel to obtain a running time of $O(mn \log n)$. This is the best known strongly polynomial algorithm [Cor09].

References

[Cor09] Thomas H. Cormen. *Introduction to Algorithms, 3rd Edition.* The MIT Press, 2009.

[Tre] Luca Trevisan. Standford university cs261 lecture notes lecture 12. `http://theory.stanford.edu/~trevisan/cs261/lecture12.pdf`.