University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 22: Local and Dynamic Push-Relabel Flow Algorithms for Local Sparse Cuts and Expander Pruning

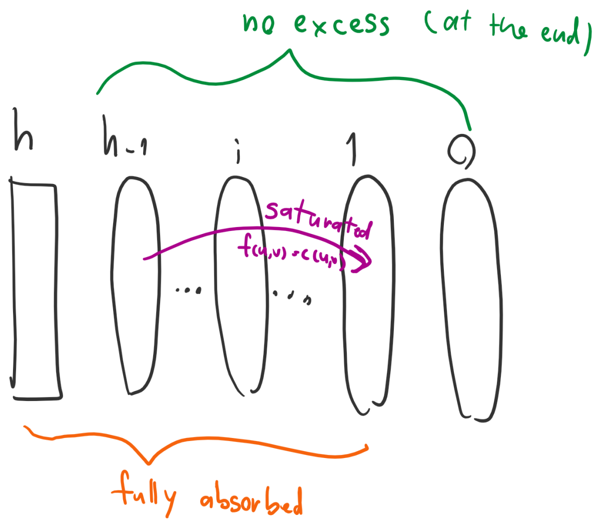
November 18th, 2021

Instructor: Thatchaphol Saranurak

Scribe: Nikhil Shagrithaya

1 Recap

Recall that we discussed PUSH/RELABEL Flow in the last class, where given the parameters G, the input graph, Δ, T, the sink and source functions respectively, and parameter h, the number of levels an active vertex can attain, and additionally, given a (G, Δ, T)-valid state (f, ℓ), we were required to output a (G, Δ, T)-valid solution.



Definition 1.1. We say that (f, ℓ) is a (G, Δ, T)**-valid state** if

1. If $\ell(u) > \ell(v) + 1$, then (u, v) is saturated from u to v (i.e. $f(u, v) = c(u, v)$)
2. If $\ell(u) \geq 1$, then u is fully absorbed (i.e. $\text{ab}(u) = T(u)$).

Definition 1.2. We say that (f, ℓ) is a (G, Δ, T)**-valid solution** if additionally we satisfy the following: if $\ell(h) < h$, then u has no excess (i.e. $\text{ex}(u) = 0$)

The time taken by this algorithm is equal to $O(mh^2)$, by using the basic implementation. However, by using the top trees data structure, we can bring down the running time to $O(mh \log n)$.

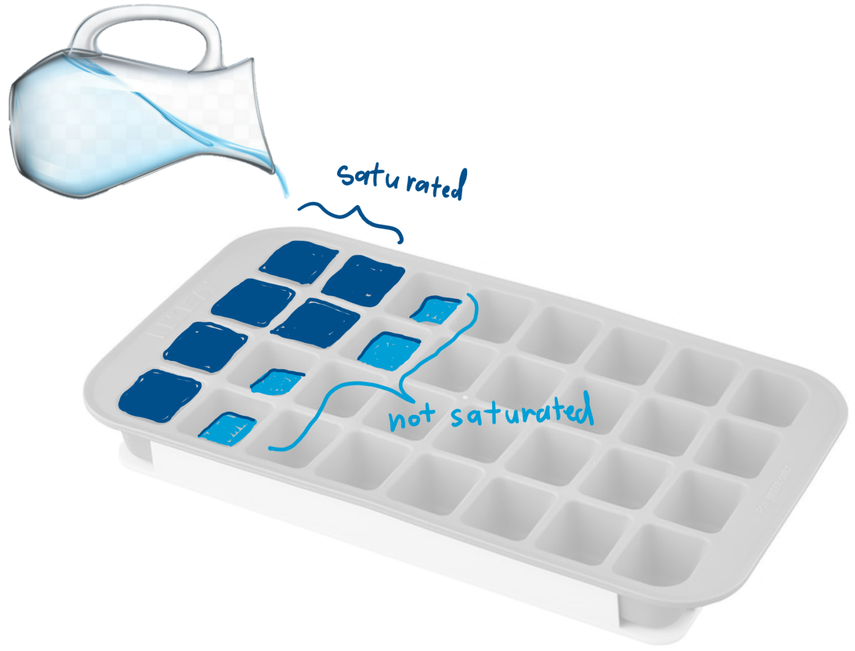
There are many applications of push-relabel, some of them being:

- **Exact Max flow:** Actually, this is a "decision version" of Max Flow. But for convenience, whenever we call BASICPUSHRELABEL with $h = n + 1$, I will just say that we call an "Exact Max Flow" subroutine. The key property here is that if the algorithm returns a cut S, then $f_{out}(S) = \delta(S)$.
- **ϕ-sparse cut or matching embedding:** We set $h = \Theta(\log(n)/\phi)$ here. The key property is: if the algorithm returns a cut S, then $f_{out}(S) \geq \delta(S) - \frac{\phi}{100} \min\{\mathrm{vol}(S), \mathrm{vol}(\overline{S})\}$.

In this lecture, we will look at **Local and Dynamic** Push-Relabel Techniques.

2 When can flow algorithms be local? The Ice Tray Analogy

Imagine pouring water into an ice tray:



The way the water flows through the tray is intuitively **local** in nature. The time taken to pour out the water is proportional to only the **volume of the source water, not the size of the ice tray.**

Let's abstract out the reason why it is local. Every node is a "large enough" sink. (Otherwise, imagine that all cells in the ice tray almost cannot absorbed any flow, except that furthest cell. Then, the water would need to "read" the whole ice tray to find a sink.) Moreover, every node forwards flow to the adjacent node only when its sink is saturated. (This is how PUSH/RELABEL works!)

3 Local PUSH/RELABEL

With the above intuition in mind, let's try to analyze PUSH/RELABEL when every node is a "large enough" sink. Formally, we consider PUSH/RELABEL in the following setting.

Parameters: G, Δ, T, h where $T(u) = \deg(u)$ for all $u \in V$ where $\deg(u)$ is the unweighted degree of u. Additionally, we think of all quantities as being integral:

- integral source function $\Delta : V \to \mathbb{Z}_{\geq 0}$
- integral edge capacities $c : E \to \mathbb{Z}_{>0}$

We also require the same guarantee: given a (G, Δ, T)-valid state (f, ℓ), output a (G, Δ, T)-valid solution. There are nice applications in this specific setting.

We will show a variant of PUSH/RELABEL, LOCALPR, that solves the above flow problem in local time. Specifically, the running time is $O(\Delta(V)h)$, so it is only proportional to total source, not graph size. This was first shown in [HRW'17].[1]

3.1 Algorithm Description

We will require one additional invariant: that $\text{ex}(u) \leq \deg(u)$ for all $u \in V$, unless that excess is initially placed at u. To maintain this, we edit the basic PUSH/RELABEL algorithm using the red part below.

```
LOCALPR($G, \Delta, T, h, (f, \ell)$)
.   Assertion: $(f, \ell)$ is $(G, \Delta, T)$-valid.
.   While $\exists$ active vertex $u$ (i.e. $\ell(u) < h$ and $\text{ex}(u) > 0$)
.   .   Let $v$ be the active vertex with smallest $\ell(v)$.
.   .   PUSH/RELABEL($u$).
```

```
PUSH/RELABEL($u$)
.   If $\exists$ admissible arc $(u, v)$ (i.e. $r_f(u, v) > 0$, $\ell(u) = \ell(v) + 1$)
.   .   PUSH($u, v$).
.   Else RELABEL($u$).
```

```
PUSH($u, v$)
.   Assertion: $u$ is active, $(u, v)$ is admissible, and $\text{ex}(v) = 0$
.   $\psi = \min\left(\text{ex}(u), r_f(u, v), \deg(v) - \text{ex}(v)\right)$ //Observe that $\psi \geq 1$. Why?
.   Send $\psi$ units of supply from $u$ to $v$: $f(u, v) \leftarrow f(u, v) + \psi$
```

```
RELABEL($u$)
.   Assertion: $u$ is active and there is no admissible arc $(u, v)$
.   $\ell(u) \leftarrow \ell(u) + 1$.
```

Observe how the excess on each node u evolves. Initially, it is possible that $\text{ex}(u) > \deg(u)$. Also, $\text{ex}(u)$ can only decrease until $\text{ex}(u) = 0$. The vertex u keeps pushing initial excess and u never receives additional mass until $\text{ex}(u) = 0$. Only then, $\text{ex}(u)$ can increase, but $\text{ex}(u) \leq \deg(u)$ always holds.

[1] They call this variant a *Unit-Flow* algorithm https://arxiv.org/pdf/1704.01254.pdf

3.2 Running Time

Lemma 3.1. *The total time spent on* RELABEL *is $O(\Delta(V)h)$.*

Proof. According to the algorithm, we only relabel vertices when their level is at least 1. But because each $u \in L_{\geq 1}$ is fully absorbed, i.e. $\text{ab}(u) = \deg(u)$, whenever we call RELABEL(u), we charge the cost of $O(\deg(u))$ to absorbed mass at u, each of $O(1)$. Recall the original analysis of PUSH/RELABEL: we charge $O(\deg(u))$ time to each RELABEL(u). This is the time for finding an admissible edge by scanning through list of neighbors.

How much time is each unit of absorbed mass charged? It is at most h. Since total absorbed mass is trivially at most $\Delta(V)$. The total time is then $O(\Delta(V)h)$.

To summarize the above argument in a more algebraic way, we have the total cost spent on RELABEL is:

$$\begin{aligned}\sum_{u \in L_{\geq 1}} O(1) \cdot \deg(u) \cdot h &= O(1) \cdot \text{ab}(L_{\geq 1}) \cdot h \\ &\leq O(1) \cdot \Delta(V) \cdot h\end{aligned}$$

□

Now we analyze the number of times the PUSH algorithm is called.

Lemma 3.2. *The total number of* PUSH *is at most $2\Delta(V)h$. So the total time spent on* PUSH *is $O(\Delta(V)h)$.*

Proof. Consider the potential function

$$\Lambda = \sum_v \text{ex}(v)\ell(v).$$

Each PUSH decreases Λ by at least $\psi \geq 1$, as ψ units of excess is moved from level i to $i-1$. Moreover, each RELABEL(u) increases Λ by at most $\text{ex}(u)$. If $\text{ex}(u) > \deg(u)$, all these excess at u must be initial excess of u. We charge 1 to each unit of this initial excess. Each unit of mass is charged, as initial excess, at most h. If $\text{ex}(u) \leq \deg(u)$, then $\text{ex}(u) \leq \text{ab}(u)$. We charge 1 to each unit of the absorbed mass. Each unit of mass is charged, as absorbed mass, at most h.

So in total, RELABEL increases Λ by at most $\Delta(V) \times 2h$. So #PUSH $\leq 2\Delta(V)h$.

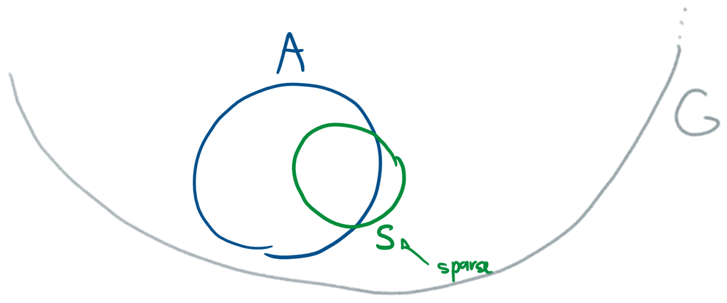
□

4 Local Sparse Cut: Application of Local PUSH/RELABEL

Consider this situation: Given a vertex set $A \subset V$ with a promise that there is some ϕ-sparse cut S^* "close to" A. More formally,

$$\text{vol}(S^* \cap A) \geq \sigma \text{vol}(S^*)$$

for some $\sigma > 0$.



Our objective is to find a sparse cut, but we don't want to read the whole graph.

The idea is that we call LOCALPR on input (G_c, Δ, T, h), where G_c is the the graph G with edge capacity $c(e) = 1/\phi$ for all $e \in E(G)$.

$$\Delta(u) = \begin{cases} \frac{1}{\sigma}\deg_G(u) & \text{if } u \in A \\ 0 & \text{if } u \notin A \end{cases}$$

and

$$T(u) = \deg_G(u) \forall u$$

$h = \Theta(\frac{\log n}{\phi})$ similar to when we use PUSH/RELABEL for finding ϕ-sparse cuts.

Can LOCALPR terminate with no excess? No, because $\Delta(S^*) > \delta_{G_c}(S^*)$:

$$\Delta(S^*) = \frac{1}{\sigma}\text{vol}_G(S^* \cap A) \geq \text{vol}_G(S^*)$$

and

$$\delta_{G_c}(S^*) = \frac{1}{\phi}\delta_G(S^*) < \text{vol}_G(S^*)$$

So LOCALPR terminates with some excess, i.e., some vertex u has label $\ell(u) = h$.

There is a level cut $L_{\geq k} = \{u \mid \ell(u) \geq k\}$ where

$$\Phi_G(L_{\geq k}) \leq 1.1\phi.$$

Exercise: This follows from exactly the same analysis as in previous class. Note: We also have that $L_{\geq k}$ is small: $\text{vol}(L_{\geq k}) \leq \Delta(V) = \text{vol}(A)/\sigma$. The time take is $O(\Delta(V)h) = O(\frac{\text{vol}(A)}{\sigma\phi}\log n)$.

Exercise: show that $h = \Theta(\frac{\log(\text{vol}(A)/\sigma)}{\phi})$ works too.

To conclude:

Theorem 4.1. *Let $A \subset V$ be a vertex set where $\text{vol}(A) < \sigma\text{vol}(V)$ for some $\sigma > 0$. (Why we need this?).*

Suppose there is S^* where $\Phi_G(S^*) < \phi$ and $\text{vol}(S^* \cap A) \geq \sigma\text{vol}(S^*)$.

Then, we can find a cut S where $\Phi_G(S) < 1.1\phi$ and $\text{vol}(S) \leq \text{vol}(A)/\sigma$ in time $O(\frac{\text{vol}(A)}{\sigma\phi}\log(\text{vol}(A)/\sigma))$

4.1 Big picture: Local Clustering Algorithms

So far, we have seen 2 local algorithms for finding "clusters" in graphs. Localized PUSH/RELABEL for Local sparse cut. (Today). Localized Ford-Fulkerson for Local cut of size k. There is another class of local algorithms based on random walks or localized PageRank.

Many more interesting questions about local clustering algorithm:

- Local densest subgraph?
- Local graph with many triangles?
- On hypergraphs?

5 Expander Pruning: A Stronger Robustness Characterization of Expanders

The question we ask here is: can you "repair" an expander?

We formally describe the problem that captures "repairing" expanders:

Problem 5.1 (One-batch Expander Pruning)**.** Given a ϕ-expander $G = (V, E)$ and an edge set $D \subset E$, let $G' = G \setminus D$. We need to find a *prune set* $P \subset V$ such that:

- $G'[V \setminus P]$ is a $\Omega(\phi)$-expander, and
- $\mathrm{vol}_G(P) = O(|D|/\phi)$.

We will show how to solve this problem. This gives a *stronger robustness characterization of expanders*. Recall the characterization from the first lecture: after d edge deletions, there is P with $\mathrm{vol}(P) = O(d/\phi)$ where $G'[V \setminus P]$ is connected, and P contains all small disconnected components.

In this stronger characterization, we will show that after d edge deletions, there is a P with $\mathrm{vol}(P) = O(d/\phi)$, where $G'[V \setminus P]$ is still $\Omega(\phi)$-expander.

In fact, we can show something even stronger: we can repair an expander under sequence of deletions **at all times.**

Problem 5.2 (Expander Pruning)**.** Given a ϕ-expander $G = (V, E)$ and an online sequence of edge deletions $D = (e_1, e_2, \ldots, e_d)$, let G_i be the graph G after time i (after deleting $e_1, \ldots, e_i$). We need to maintain the prune set $P \subset V$ where P_i denote the set P after time i with the following properties:

- $G_i[V \setminus P_i]$ is a $\Omega(\phi)$-expander.
- $\mathrm{vol}_G(P_i) \le 8i/\phi$, and $P_{i-1} \subseteq P_i$ (it is an incremental set)

Expander pruning is a very important tool for almost all dynamic algorithms based on expanders. Expanders are great, but everything we have talked so far is for static graphs. Expander pruning allows us to exploit the power of expanders in the dynamic setting.

6 One-batch Expander Pruning

In this lecture, we only consider the easier version of the problem (the one-batch version). Once we understand the technique for this version, it easily extends to the full dynamic version.

6.1 Terminology

Consider an unweighted graph G and a vertex set $U \subseteq V$. (Everything that follows can be extended to capacitated graphs.) Let $G\{U\}$ denote the following **induced subgraph with boundary vertices**. Start with the induced subgraph $G[U]$ and then for each **boundary edge** $e = (u, v) \in E(U, V \setminus U)$ with endpoint $u \in U$, create a new vertex x_e and add the edge (x_e, u) to $G\{U\}$.

Let $\partial_G \langle U \rangle = V(G\{U\}) \setminus U$ be the set of **boundary vertices**. For each boundary vertex $x_e \in \partial_G \langle U \rangle$ where $e = (u, v)$ and $v \in U$, we say that v is the **partner** of x_e. $G\{U\}$ is a **near ϕ-expander** if:

$$\delta_{G\{U\}}(S) \geq \phi \mathrm{vol}_{G\{U\}}(S)$$

for all $S \subseteq U$ where $\mathrm{vol}(S) \leq \mathrm{vol}(U)/2$.

Intuitively, this means that $G[U]$ is almost an expander, but we count boundary edges as well. Note that, for any U, we have $\deg_{G\{U\}}(u) = \deg_G(u) \forall u$ and $\mathrm{vol}_{G\{U\}}(S) = \mathrm{vol}_G(S) \forall S \subseteq U$. Therefore, degree and volume remain fixed even when we consider different U.

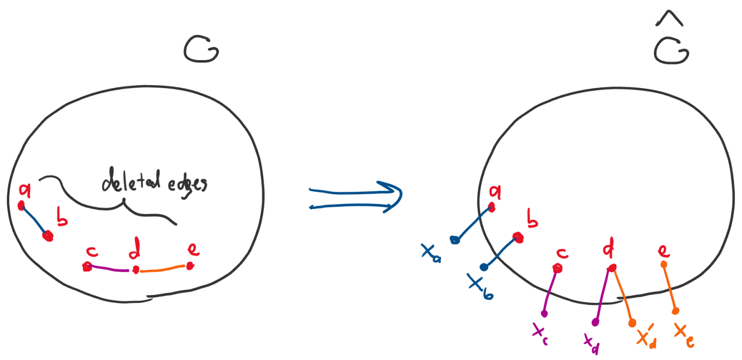
6.2 Reformulation

If we can solve the following problem, then we can solve the one-batch expander pruning problem:

Problem 6.1 (Pruning near expanders)**.** Let $G = (V, E)$ and $U \subset V$ where $G\{U\}$ is a near ϕ-expander. Compute a *prune set* $P \subseteq U$ and $U' = U \setminus P$ such that:

- $G[U']$ is a $\Omega(\phi)$-expander
- $\mathrm{vol}(P) \leq O(|\partial_G \langle U \rangle| / \phi)$.

Intuitively, we want to make a near-expander an expander by pruning small part of it. Given a ϕ-expander $G = (V, E)$ and a deleted edge $D \subset E$, create a graph $\widehat{G}$ as follows:



Observe that $\widehat{G}\{V\}$ is a near ϕ-expander because for all $S \subseteq V$ where $\mathrm{vol}(S) \leq \mathrm{vol}(V)/2$, we have

$$\delta_{\widehat{G}\{V\}}(S) \geq \delta_G(S) \geq \phi \mathrm{vol}_G(S) = \mathrm{vol}_{\widehat{G}\{V\}}(S)$$

We also have $\mathrm{vol}(P) \leq O(|\partial_{\widehat{G}} \langle V \rangle| / \phi) = O(|D|/\phi)$, and $\widehat{G}[V \setminus P] = G'[V \setminus P]$ where $G' = G \setminus D$

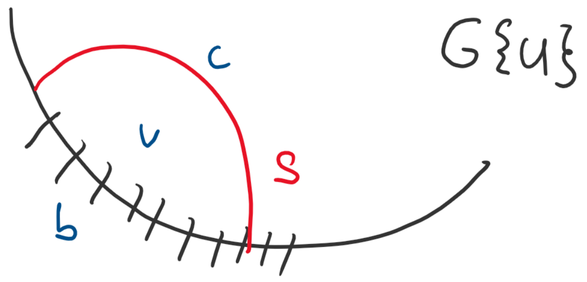
6.3 Useful Structure

Lemma 6.2. *Suppose that $G\{U\}$ is a near ϕ-expander, but $G[U]$ is not a $\phi/6$-expander. Then, there is $S \subset U$ where*

$$|E(S, U \setminus S)| \leq |E(S, V \setminus U)|/5.$$

Define the following quantities:

- $c = |E(S, U \setminus S)|$ (cut edges)
- $b = |E(S, V \setminus U)|$ (boundary edges)
- $v = \mathrm{vol}(S)$



We have:

$$c < \frac{1}{6}\phi v \text{ and } c + b \geq \phi v$$

and so $b \geq \frac{5}{6}\phi v$ and so

$$c \leq b/5.$$

Note that, if $G[U]$ is not a $\phi/1000$-expander, we would have $c \leq b/999$. You should think of $c \ll b$ for intuition.

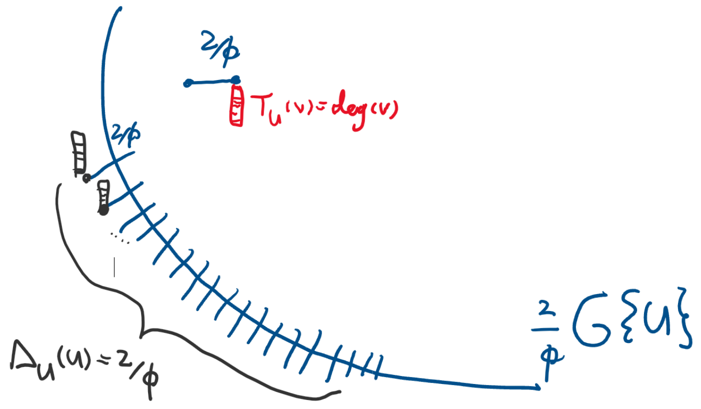
6.4 Certifying that a Near-expander is an Expander

The U-**boundary source function** $\Delta_U : V(G\{U\}) \to \mathbb{R}_{\geq 0}$ is defined as:

$$\Delta_U(u) = \begin{cases} 1 & \text{if } u \in \partial_G \langle U \rangle \\ 0 & \text{if } u \in U \end{cases}$$

The U-**sink function** $T_U : V(G\{U\}) \to \mathbb{R}_{\geq 0}$ is defined as

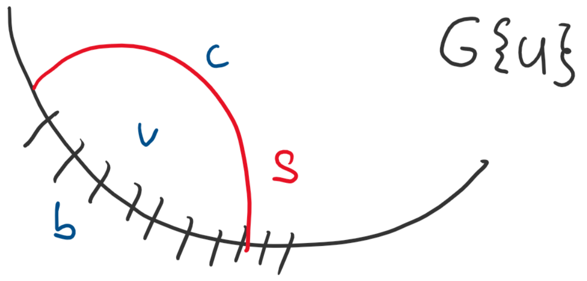
$$T_U(u) = \begin{cases} 0 & \text{if } u \in \partial_G \langle U \rangle \\ \deg_G(u) & \text{if } u \in U \end{cases}$$



Lemma 6.3. *Let $G\{U\}$ be a near ϕ-expander. If there is a feasible flow f in $\frac{2}{\phi}G\{U\}$ satisfying $(\frac{2}{\phi}\Delta_U, T_U)$, then $G[U]$ is a $\phi/6$-expander.*

In words, we can certify that $G[U]$ is already an expander, if there is a feasible flow f in $\frac{2}{\phi}G\{U\}$ where **every** boundary edge on sends flow at its full capacity, and each non-boundary u vertex absorbed at most $\deg(u)$.

Proof. Suppose for contradiction, there is an $S \subset U$ where $\mathrm{vol}(S) \le \mathrm{vol}(U)/2$



where $c \le b/5$. (Think of $c \ll b$ for intuition.)

Now, if f is feasible, we have

$$\underbrace{\frac{2}{\phi}c}_{\text{flow out}} \ge \underbrace{\frac{2}{\phi}b}_{\text{initial mass}} - \underbrace{v}_{\text{absorbed mass}}$$

But the absorbed mass is at most $v \le \frac{1}{\phi}(b + c)$. For intuition, if $c \ll b$, the absorbed mass is essentially at most half of the initial mass. So half of flow must go out, which means that $c \ge b/2$ (why?). But this is a contradiction because $c \ll b$.

More formally, we have $v \le \frac{1}{\phi}(b + c) \le \frac{6}{5\phi}b$ and so

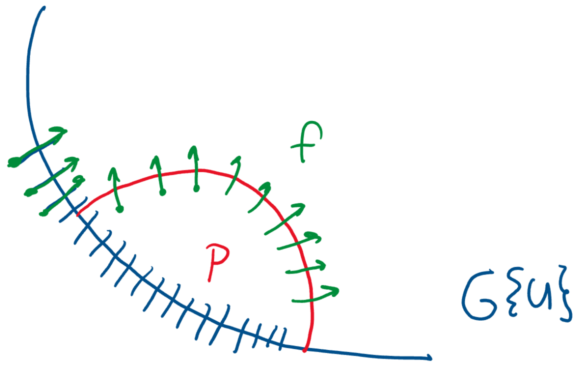
$$\frac{2}{\phi}c \ge \frac{4}{5\phi}b \iff c \ge \frac{2}{5}b$$

which contradicts the fact that $c \le b/5$. □

6.5 Warm up: Slow Algorithm by Solving Exact Max Flow

Suppose we call BASICPUSHRELABEL on $(\frac{2}{\phi}G\{U\}, \frac{2}{\phi}\Delta_U, T_U)$ with $h = n+1$, and also suppose there is no excess. Then we get a feasible flow and so $G[U]$ itself is a $\phi/6$-expander. Then we are done, with $P = \emptyset$.

But what if there is excess? In this case, BASICPUSHRELABEL returns a cut $P \subseteq V(G\{U\})$. Let f be the preflow in $\frac{2}{\phi}G\{U\}$ maintained byBASICPUSHRELABEL. Recall that $f_{out}(P) = \delta_{\frac{2}{\phi}G\{U\}}(P)$. For $u \notin P$, we have $\mathrm{ex}(u) = 0$.



Quick question: For each boundary vertex $x_e \in P$, if its unique adjacent vertex $u \in P$, then $x_e \in P$ too. We will show that we can just return P as our prune set.

Lemma 6.4. *Let $U' = U \setminus P$. Then $G[U']$ is a $\phi/6$-expander.*

Proof. Consider the preflow f computed by BASICPUSHRELABEL, and let f' be obtained from f by restricting to $G\{U'\}$. Observe that f' is a feasible flow in $\frac{2}{\phi}G\{U'\}$ satisfying $(\frac{2}{\phi}\Delta_{U'}, T_{U'})$. Note that $G\{U'\}$ is a near ϕ-expander (as $G\{U\}$ is a near ϕ-expander). So, $G[U']$ is a $\phi/6$-expander by Lemma 6.3. □

Lemma 6.5. $\mathrm{vol}_G(P) \le O(|\partial_G \langle U \rangle|/\phi)$

Proof. For each $u \in P$, we have $\mathrm{ab}(u) = T(u) = \deg_G(u)$. So

$$\mathrm{vol}_G(P) = \mathrm{ab}(P) \le \frac{2}{\phi}\Delta_{U'}(V(G\{U\}) = \frac{2}{\phi}|\partial_G \langle U \rangle|.$$

□

To summarize: our goal is a partition (P, U') of U such that P is not too big, and there is a feasible flow in $\frac{2}{\phi}G\{U'\}$ where every U'-boundary edge can send flow at its full capacity. This implies that $G[U']$ is a $\phi/6$-expander.

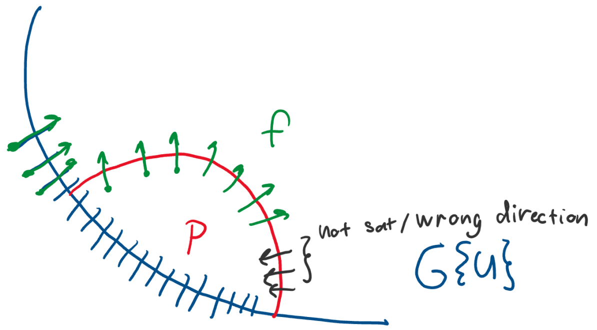
Exercise 6.6. Show that every U'-boundary edge can send flow by at least 0.9 fraction of its capacity, we can still conclude that $G[U']$ a $\Omega(\phi)$-expander.

Actually, suppose there is a subroutine that returns (P, U') and only guarantees that every U'-boundary edge can send flow by at least α fraction of its capacity for some small $\alpha > 0$, show how to adjust the algorithm to certify that $G[U']$ is a $\Omega(\alpha\phi)$-expander. *Hint: Work with $\frac{2}{\alpha\phi}G\{U\}$ instead.*

Exercise 6.7. Instead of calling BASICPUSHRELABEL, we can call LOCALPUSHRELABEL too.

6.6 What is Wrong if We Don't Call Exact Max Flow?

Say we call BasicPushRelabel with $h < n + 1$. If BasicPushRelabel terminates with excess, then the returned cut P can be such that $f_{out}(P) < \delta_{\frac{2}{\phi}G\{U\}}(P)$.



When h is big enough, like $h = O(\frac{\log n}{\phi})$, we know the portion of "bad cut edges" is quite small.

This motivates the following idea: the (restriction of) flow f is almost a good enough certificate of $G[U \setminus P]$. Therefore, we can work with $U' \leftarrow U \setminus P$ and try to reuse flow f.

6.7 Fast Algorithm via Dynamic Push/Relabel

Initialize the following as follows:

- $U_1 = U$, $f_1 \equiv 0$, $\ell_1 \equiv 0$.
- $h = O(\frac{\log n}{\phi})$
- $t = 1$

While LocalPushRelabel$(\frac{2}{\phi}G\{U^{(t)}\}, \frac{2}{\phi}\Delta_{U^{(t)}}, T_{U^{(t)}}, h, (f^{(t)}, \ell^{(t)}))$ terminates with some excess, let $P^{(t)}$ be the returned cut such that

$$\begin{aligned} f_{out}^{(t)}(P^{(t)}) &\geq \delta_{\frac{2}{\phi}G\{U^{(t)}\}}(P^{(t)}) - \frac{\phi}{100}\text{vol}_{\frac{2}{\phi}G\{U^{(t)}\}}(P^{(t)}) \\ &= \delta_{\frac{2}{\phi}G\{U^{(t)}\}}(P^{(t)}) - \frac{1}{50}\text{vol}_{G\{U^{(t)}\}}(P^{(t)}) \end{aligned}$$

Let $U^{(t+1)} \leftarrow U^{(t)} \setminus P^{(t)}$, and let $(f^{(t+1)}, \ell^{(t+1)})$ be obtained from $(f^{(t)}, \ell^{(t)})$ by restricting from $\frac{2}{\phi}G\{U^{(t)}\}$ to $\frac{2}{\phi}G\{U^{(t+1)}\}$.

Exercise: what to do exactly at new boundary vertices?

Exercise 6.8. After restriction, $(f^{(t+1)}, \ell^{(t+1)})$ is a $(\frac{2}{\phi}G\{U^{(t+1)}\}, \frac{2}{\phi}\Delta_{U^{(t+1)}}, T_{U^{(t+1)}})$-valid state.

So the algorithm description is valid. By Lemma 6.3, we havethe following:

Lemma 6.9. *When* LocalPushRelabel$(\frac{2}{\phi}G\{U^{(t)}\}, \frac{2}{\phi}\Delta_{U^{(t)}}, T_{U^{(t)}}, h, (f^{(t)}, \ell^{(t)}))$ *terminates with no excess, then* $G[U^{(t)}]$ *is a* $\phi/6$*-expander.*

Lemma 6.10. *The total new amount of mass before round* $t+1$ *is* $\frac{2}{\phi}|\partial_G \langle U^{(t+1)} \rangle| - f_{out}^{(t+1)}(\partial_G \langle U^{(t+1)} \rangle \leq \frac{1}{50}\text{vol}_{G\{U^{(t)}\}}(P^{(t)})$.

Lemma 6.11. *The total destroyed mass before round* $t+1$ *is* $\mathrm{vol}_{G\{U^{(t)}\}}(P^{(t)} \cap U^{(t)})$.

Lemma 6.12. *The total amount of mass we ever added is at most* $2 \times \frac{2}{\phi}|\partial_G \langle U \rangle|$.

Lemma 6.13. *Let* $P = \bigcup_t P^{(t)}$. *We have* $\mathrm{vol}(P) \leq 2 \times \frac{2}{\phi}|\partial_G \langle U \rangle|$.

All that remains is to bound the total running time. But we saw that the total running time is proportional to the total amount of mass times h. So the total running time is $2 \times \frac{2}{\phi}|\partial_G \langle U \rangle| \times h = O(|\partial_G \langle U \rangle|\frac{\log n}{\phi^2})$.