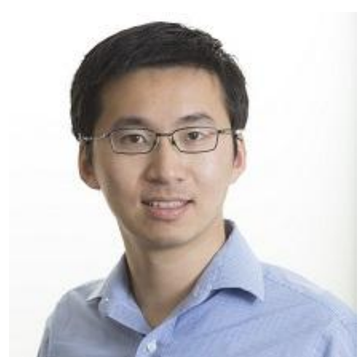
2-Norm Flow Diffusion in Near-Linear Time

Li Chen
Georgia Tech

Richard Peng
Waterloo

Di Wang
Google

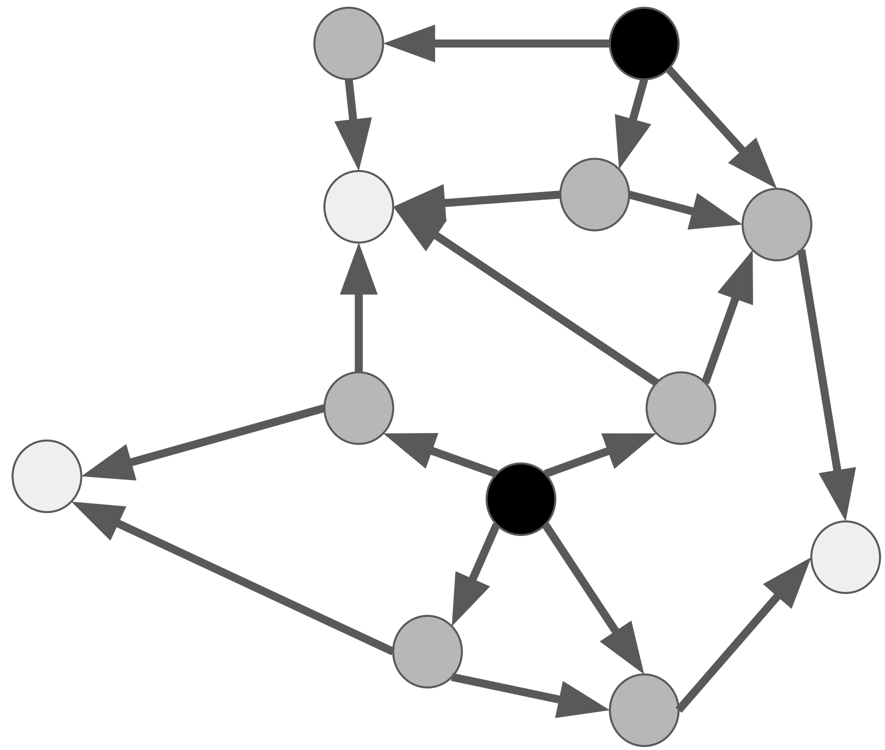


Diffusion on Graphs

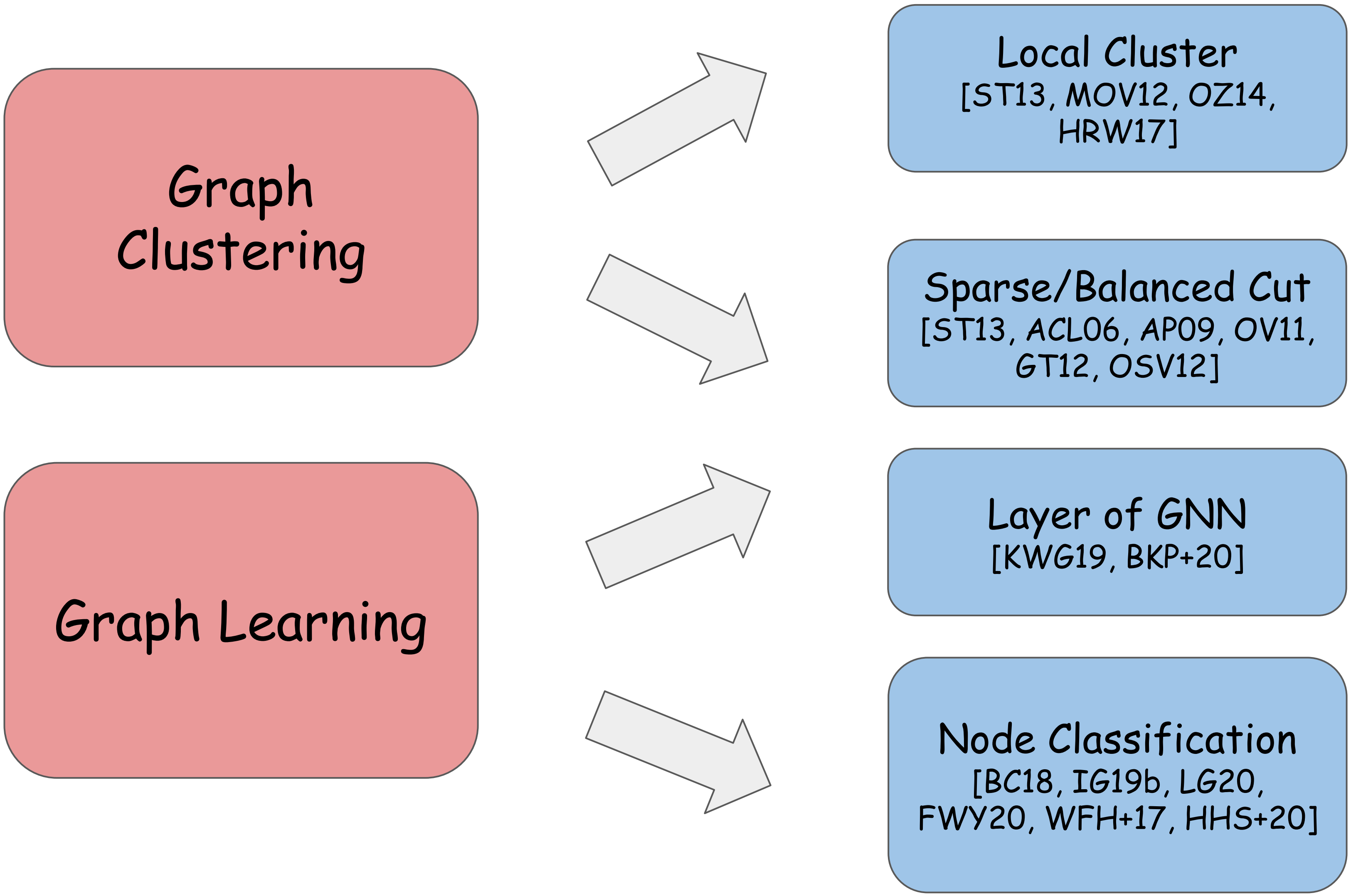
Graph G = (V, E)

n = |V|

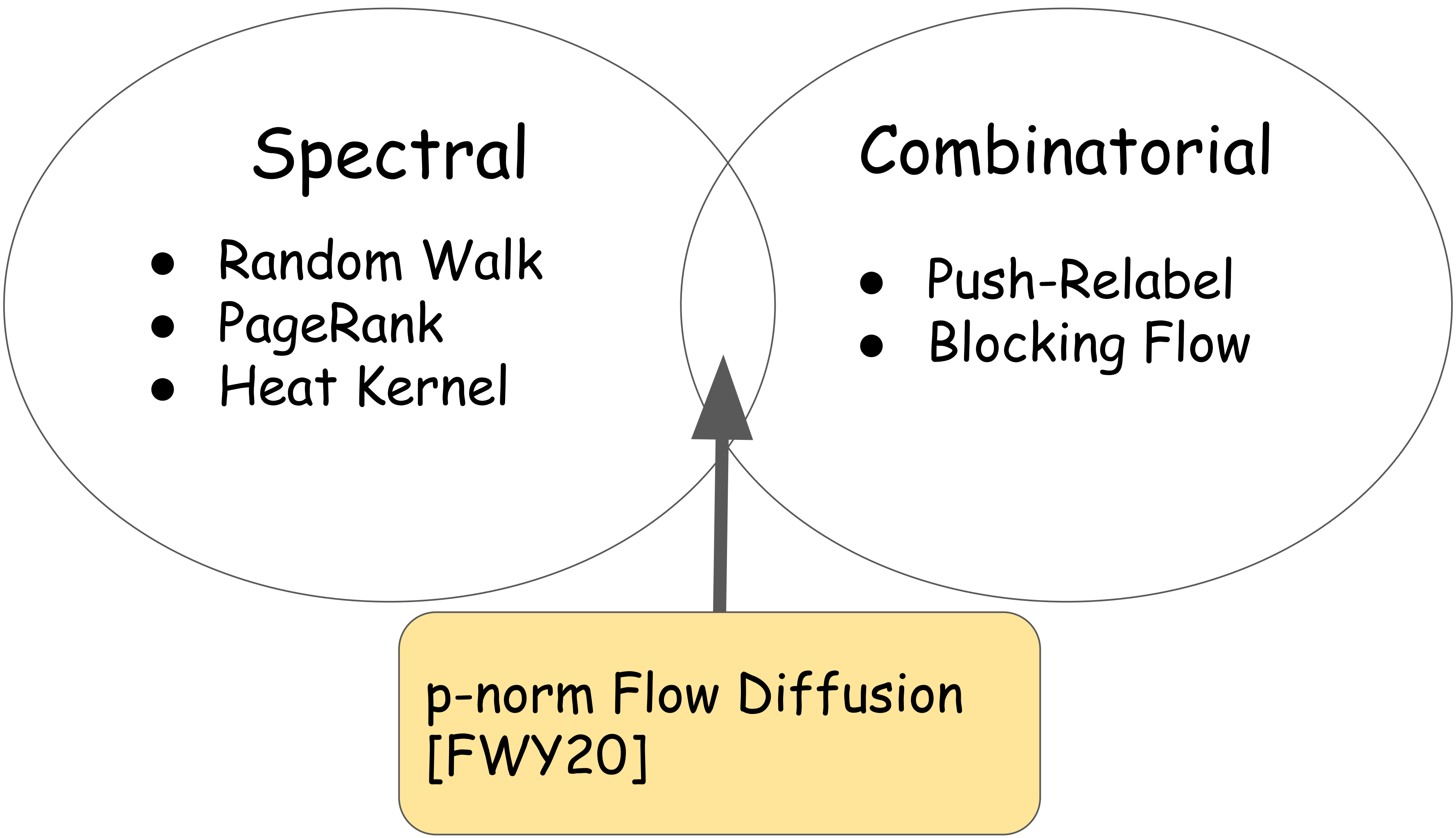
m = |E|



Applications



Diffusion on Graphs and More



p-norm Flow Diffusion [FWY20]

- $B \in \mathbb{R}^{m*n}$, Edge Incidence Matrix
- $S \geq 0 \in \mathbb{R}^n$, Supply per Vtx
- $T \geq 0 \in \mathbb{R}^n$, Sink Capacity per Vtx

$\Sigma_u S(u) \leq \Sigma_u T(u)$

$$\min_f \ ||f||_p$$

- $f \in \mathbb{R}^m$
- $S + B^T f \leq T$

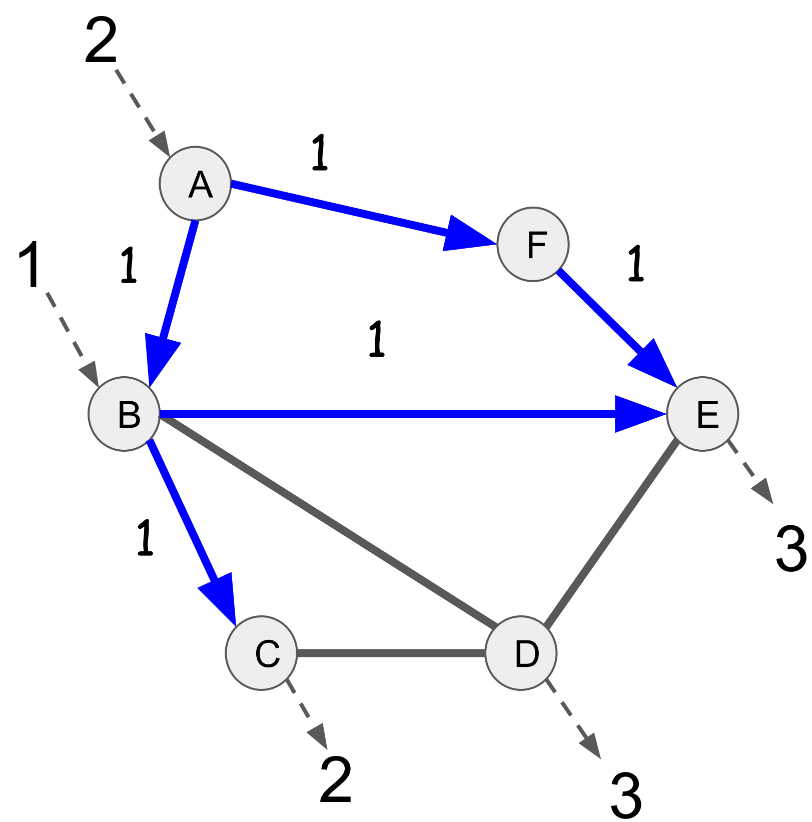
Network flow problem if $\Sigma_u S(u) = \Sigma_u T(u)$

p-norm Flow Diffusion [FWY20]

$$\min_f \|f\|_p$$

- $f \in \mathbb{R}^m$
- $S + B^T f \leq T$

$S(u) + \sum_v f(u, v) \leq T(u), u \in V$



$\|f\|_1 = 5$
$\|f\|_2 = 2.236...$
$\|f\|_\infty = 1$

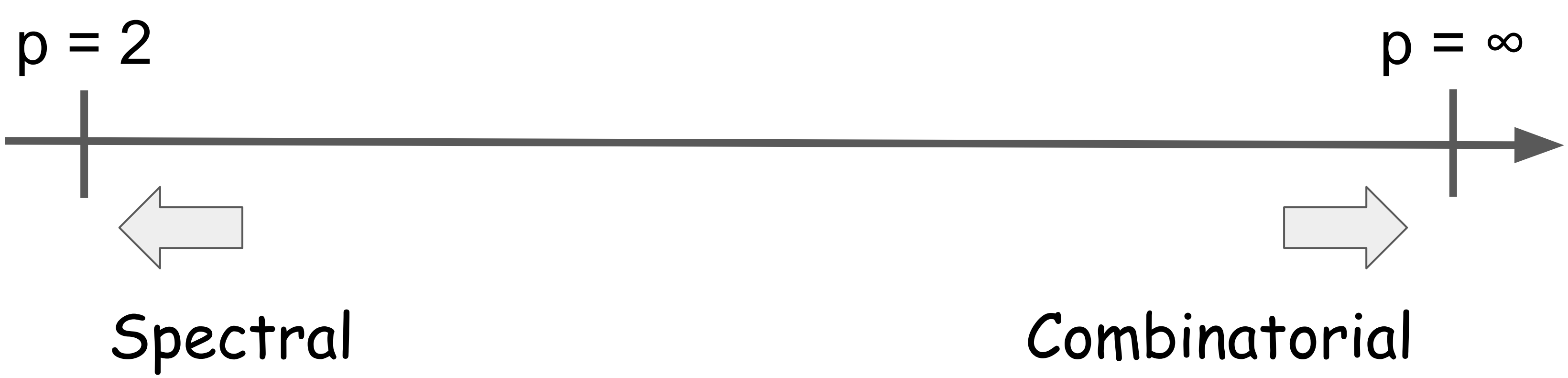
Interpolation

$$\min_f \ \|f\|_2$$

- $f \in \mathbb{R}^m$
- $S + B^T f \leq T$

$$\min_f \ \|f\|_\infty$$

- $f \in \mathbb{R}^m$
- $S + B^T f \leq T$



Our Result

Theorem

2-norm Flow Diffusion can be solved up to $(1+\varepsilon)$-error in $\tilde{O}(m \log(1/\varepsilon))$-time w.h.p.

- Algo in [FWY20] runs in $O(m^3 n^2 \log(1/\varepsilon))$-time.

Weighted Case

$$\min_f \; ||f||_2$$

- $f \in \mathbb{R}^m$
- $S + B^T f \leq T$

$\Sigma_u S(u) \leq \Sigma_u T(u)$

Weighted Case

Diagonal $R \in \mathbb{R}_{\geq 0}^{m*m}$

$\min_f \ ||f||_2$

- $f \in \mathbb{R}^m$
- $S + B^T f \leq T$

$\min_f \ \frac{1}{2} f^T R f$

- $f \in \mathbb{R}^m$
- $B^T f \leq d$

$\Sigma_u S(u) \leq \Sigma_u T(u)$

$d = T - S$

$0 \leq \Sigma_u d(u)$

Taking Dual

Diagonal $R \in \mathbb{R}_{\geq 0}^{m*m}$

$$\min_f \ \frac{1}{2} f^T R f$$

- $f \in \mathbb{R}^m$
- $B^T f \leq d$

$0 \leq \Sigma_u d(u)$

Taking Dual

Diagonal $R \in \mathbb{R}_{\geq 0}^{m*m}$

$L = B^T R^{-1} B$

$\min_f \ \frac{1}{2} f^T R f$

- $f \in \mathbb{R}^m$
- $B^T f \leq d$

⇔

$\min_x \ \frac{1}{2} x^T L x + d^T x$

- $x \in \mathbb{R}^n$
- $x \geq 0$

$0 \leq \Sigma_u d(u)$

Unconstrained = Solving Laplacian System

$L = B^T R^{-1} B$ is the Graph Laplacian matrix

$\min_x \ \frac{1}{2} x^T L x + d^T x$

- $x \in \mathbb{R}^n$
- ~~$x \geq 0$~~

=

Solve $L x = -d$

- [ST04, KMP10, KMP11, KOSZ13, LS13, CKMPPRX14, KS16, JS20]: O~(m)-Time Solver
- **Idea**: Translate these to non-negative case

Solver Framework from [ST04]

$T(m, k)$ = time for solving sized-**m** graph up to **k**-error

Theorem

$T(m, 1+\varepsilon) = O(m \log^c(m) \log(1/\varepsilon))$

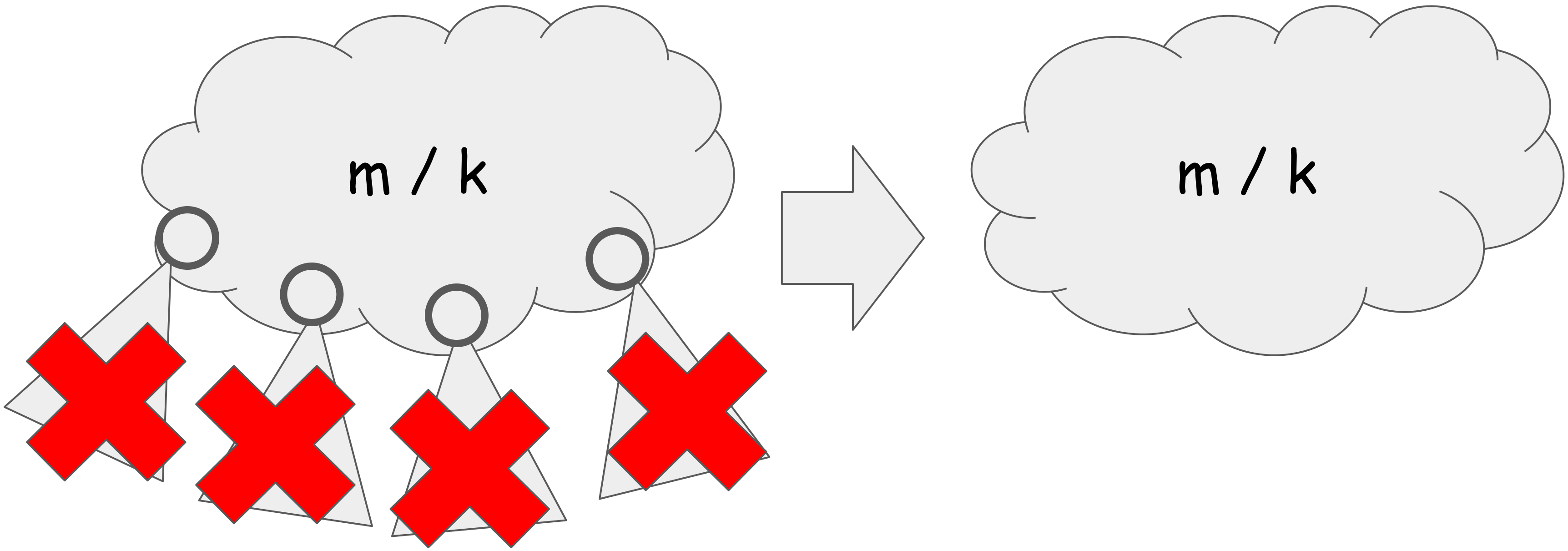
Solver Framework from [ST04]

T(m, k) = time for solving sized-**m** graph up to **k**-error

Vtx Reduce: T(n + (m / k), 2) = T(m / k, 2) + O(m)

WARNING: Hidden polylogm everywhere

Vertex Reduce in O~(m)-time



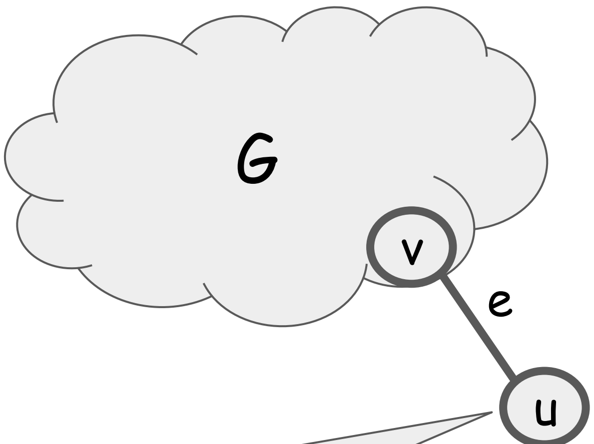
$\min_x \frac{1}{2} x^T L x + d^T x$

- $x \geq 0$

$\min_x \frac{1}{2} x^T L' x + \sum_u f_u(x(u))$

- $x \geq 0$

Vertex Reduce: Simplest Case



Determined by x(v)

Given x(v), set x(u) minimizing

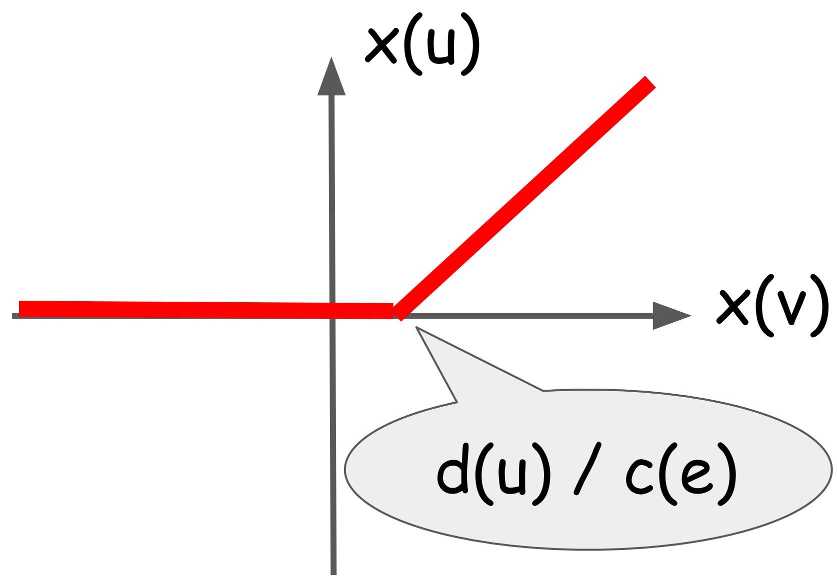
$\min_x \frac{1}{2} c(e) (x - x(v))^2 + d(u)\, x$

s.t. $x \geq 0$

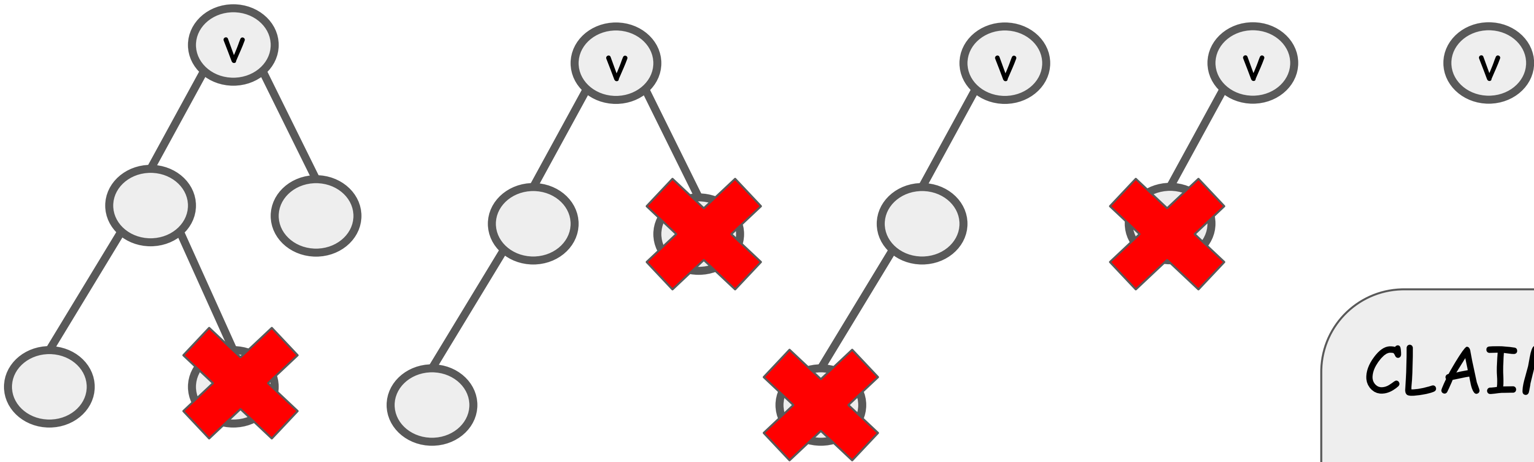
$x \geq 0$, OPT when

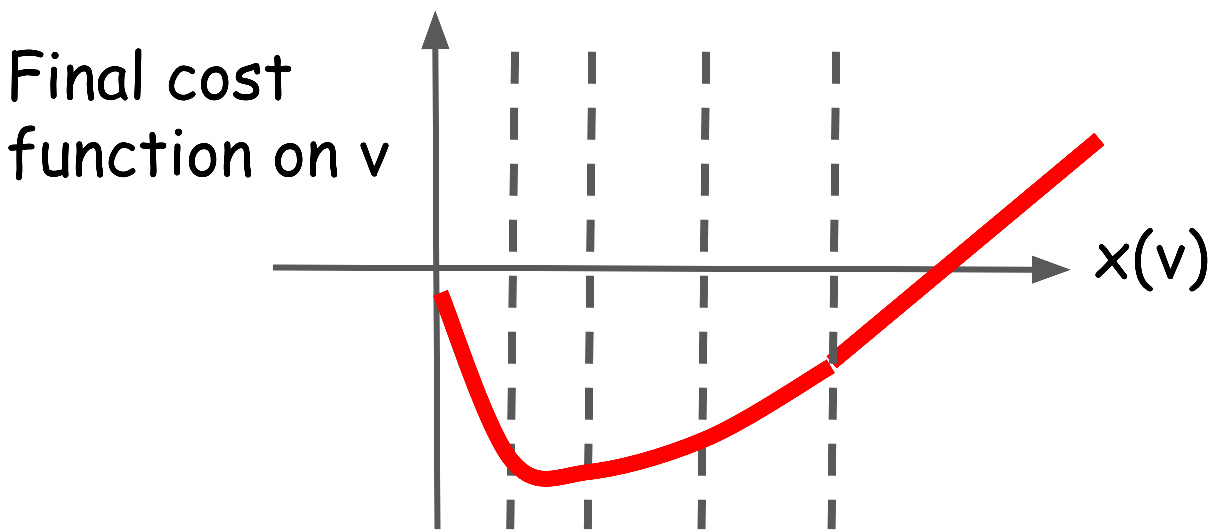
x(u)

= ReLu(x(v) - d(u) / c(e))



Reduce Degree 1 Vertices





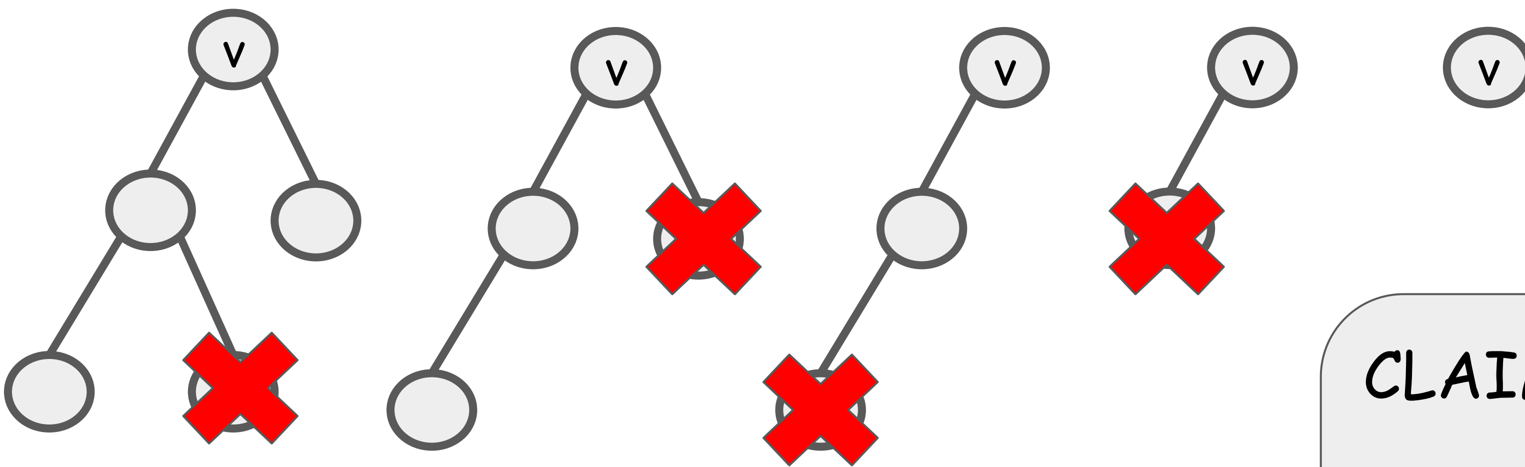
CLAIM

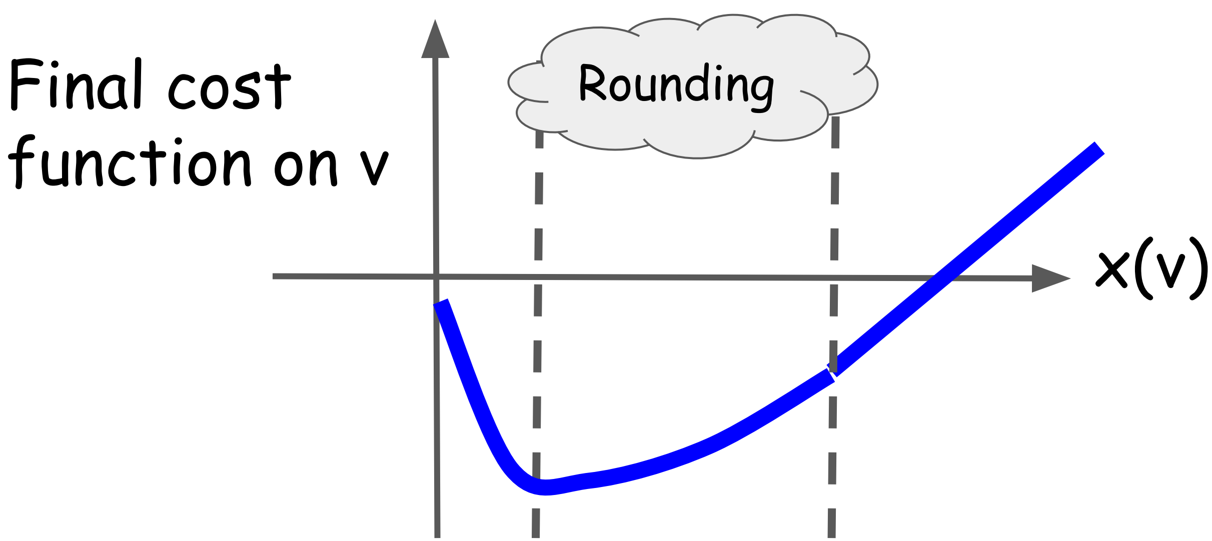
k-sized tree

⇓

k-piece quadratic convex function

Reduce Degree 1 Vertices





CLAIM

k-sized tree

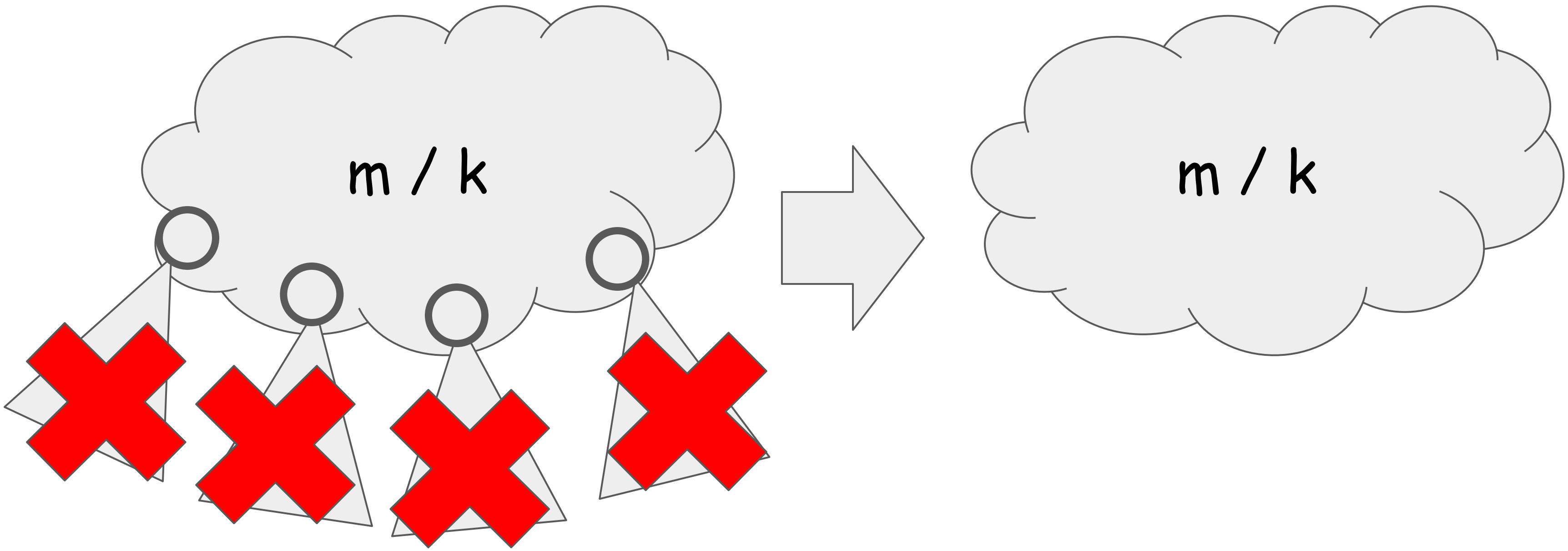
↓

k-piece
quadratic convex
function

↓

Approxed by
log m-pieces

Vertex Reduce in $\tilde{O}(m)$-time



$\min_x \frac{1}{2} x^T L x + d^T x$
- $x \geq 0$

$\min_x \frac{1}{2} x^T L' x + \sum_u f_u(x(u))$
- $x \geq 0$

Solver Framework from [ST04]

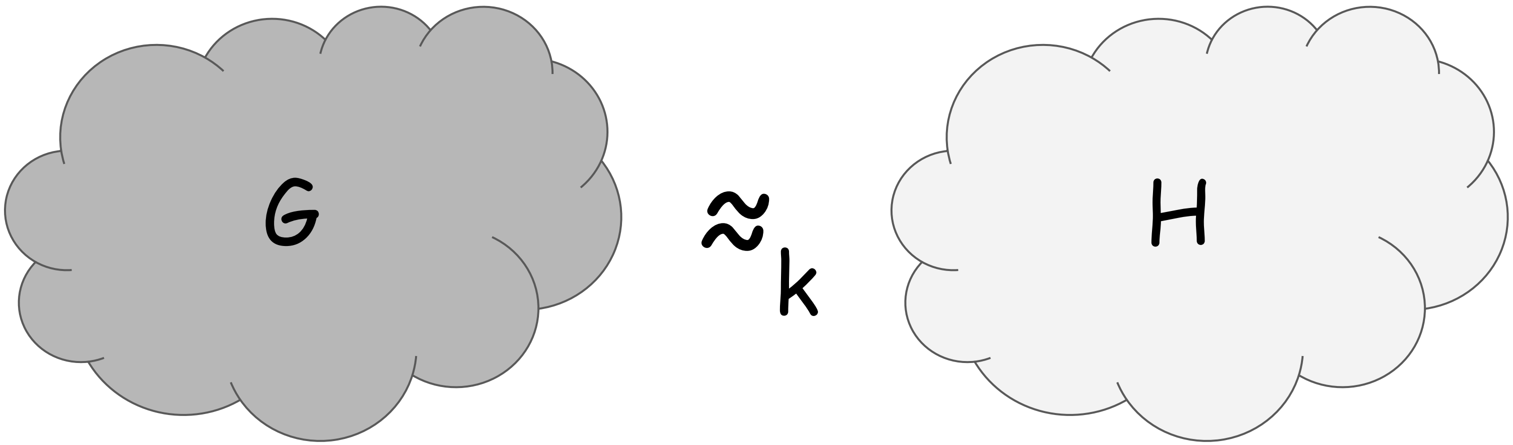
$T(m, k)$ = time for solving sized-**m** graph up to **k**-error

Vtx Reduce: $T(n + (m / k), 2) = T(m / k, 2) + O(m)$

Edge Reduce: $T(m, k) = T(n + (m / k), 2) + O(m)$

WARNING: Hidden polylogm everywhere

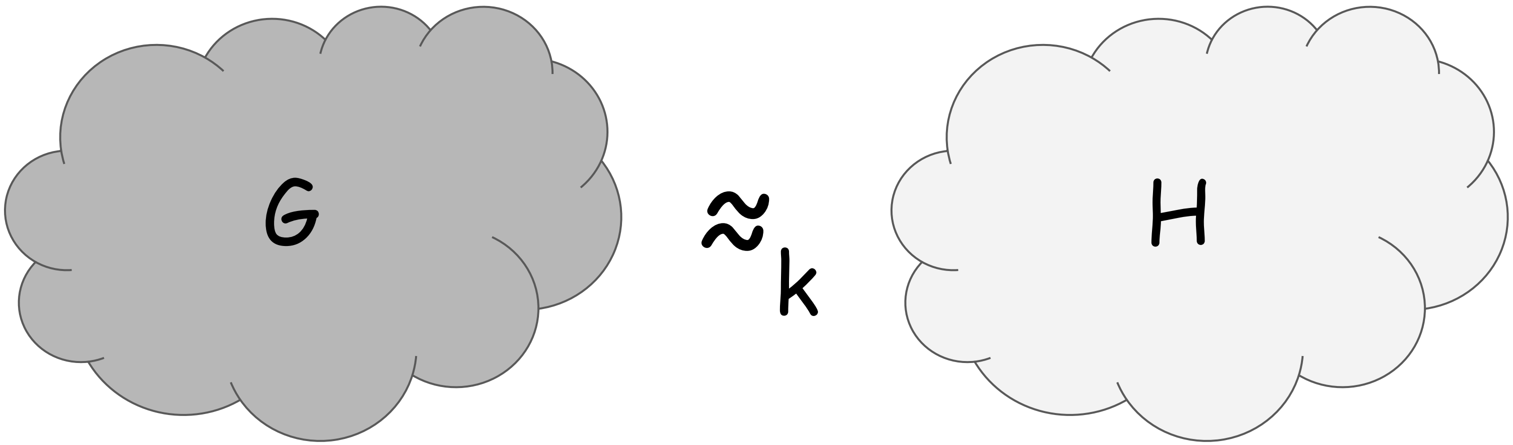
Spectral Approximation



$G \approx_k H$ if

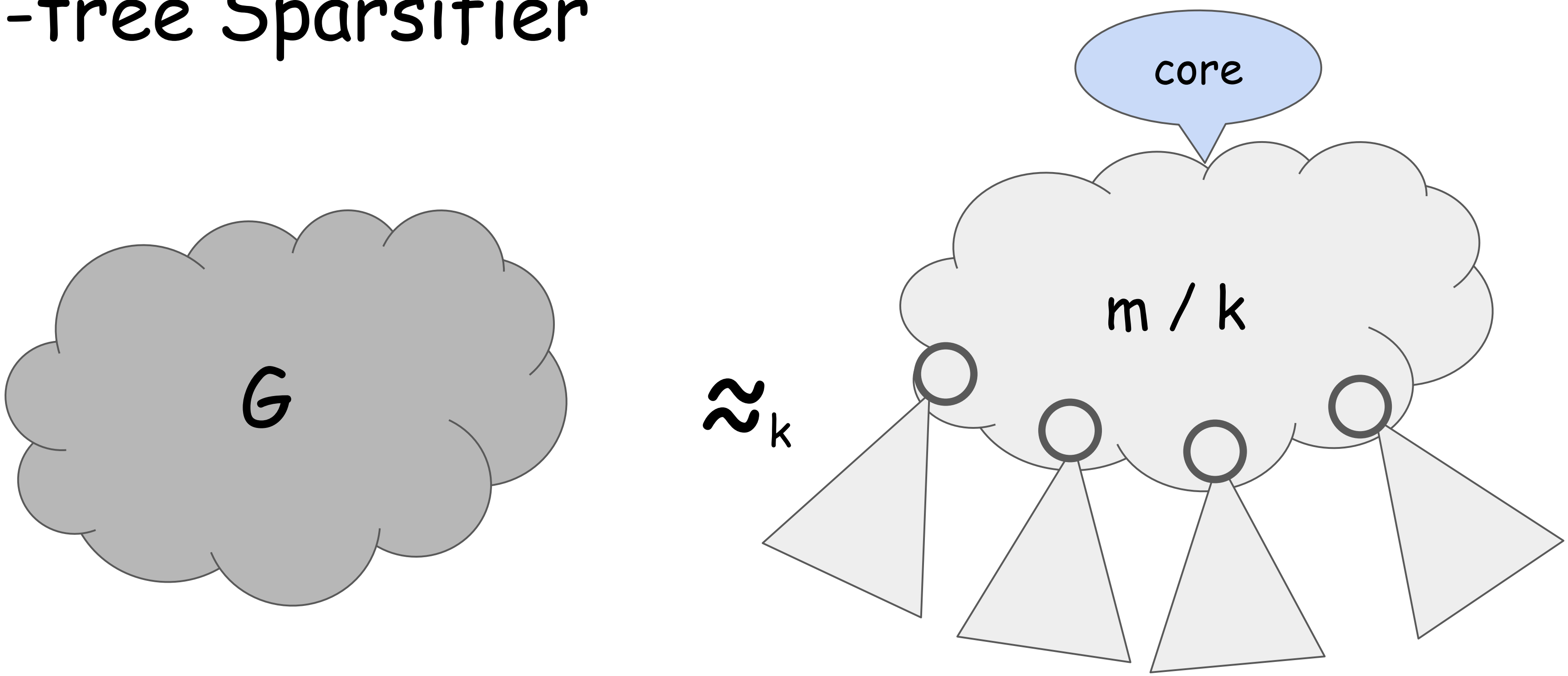
$$L(H) \leq L(G) \leq k\, L(H)$$

Spectral Approximation



O(1)-approx on H => **O(k)**-approx on G

J-tree Sparsifier



The only randomized part

J-Tree: First appear in [Mad10] for cut approximation.
Can approx distance as well. [CGHPS20]

Solver Framework from [ST04]

T(m, k) = time for solving sized-**m** graph up to **k**-error

Vtx Reduce: $T(n + (m / k), 2) = T(m / k, 2) + O(m)$

Edge Reduce: $T(m, k) = T(n + (m / k), 2) + O(m)$

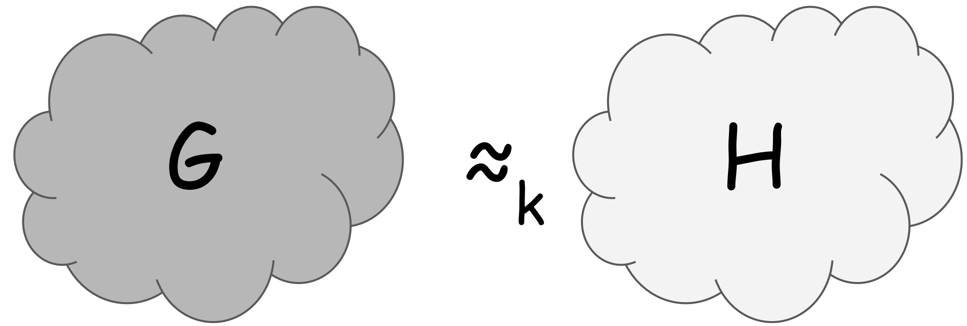
Quality Boost: $T(m, 1+\varepsilon) = \log (1 / \varepsilon) * T(m, \log m)$

WARNING: Hidden polylogm everywhere

Accelerated Proximal Gradient Method

$$\min_x \ \tfrac{1}{2}\, x^T L(G)\, x + d^T x$$

- $x \geq 0$



Accelerated Proximal Gradient Method

$\min_x \frac{1}{2} x^T L(G)\, x + d^T x$

- $x \geq 0$

$G \approx_k H$

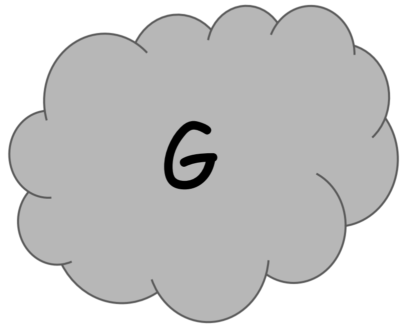
⇒ $k^{0.5}$ iters of

$\min_x \frac{1}{2} x^T L(H)\, x + d_i^T x$

- $x \geq 0$

Combine Everything

Edge Reduce

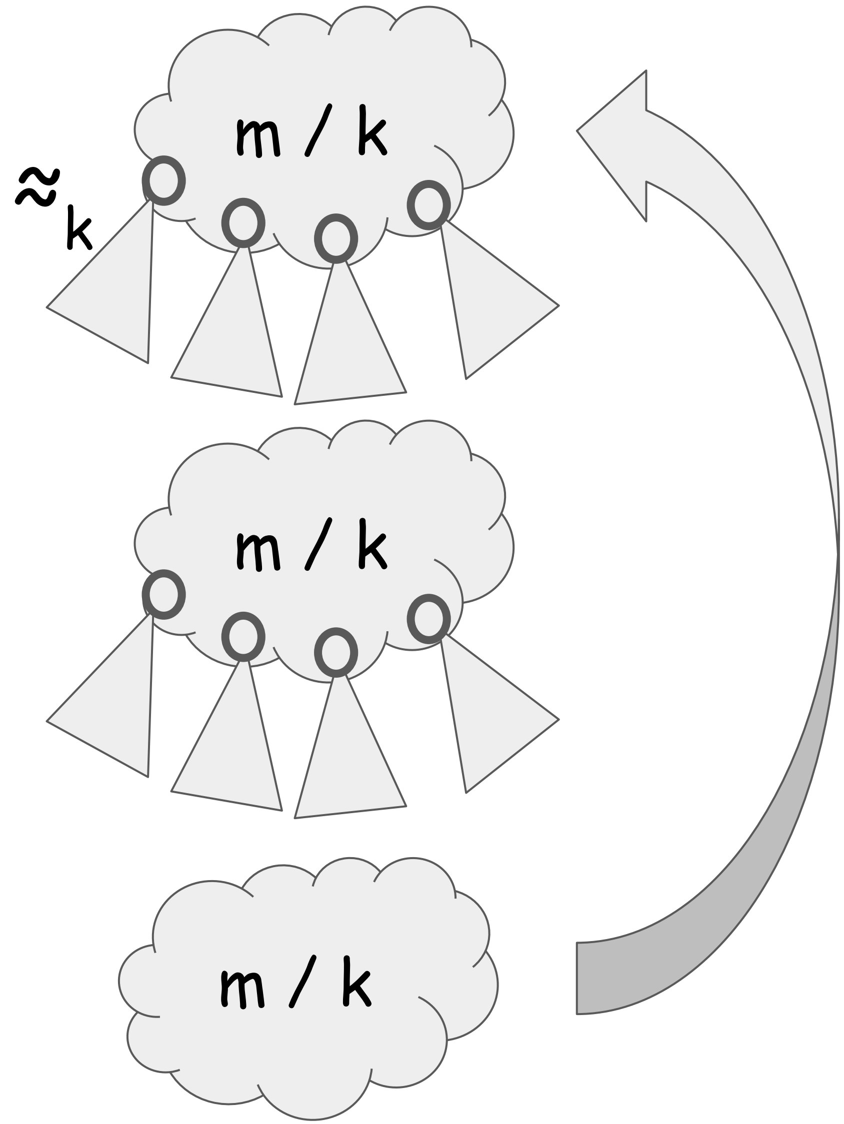


Quality Boost

$k^{0.5}$ iters of

Vertex Reduce

$k^{0.5}$ iters of



Conclusion

- $\tilde{O}(m)$-time solver for a wider class of flow problems
- $\tilde{O}(m)$-time solver for a class of Non-Negative QP
- Φ-free $\tilde{O}(m)$-time algo finding locally biased clusters
- New way dealing $\mathbf{x} \geq \mathbf{0}$ beyond Interior Point Methods
- Ultra-sparsifier with NICE properties

Open Problems

- Simpler Algo: No recursion?
- Runtime depend on sparsity of S
- Faster runtime

  $O(m \log m)$, $O(m (\log\log m)^c)$for 2-approx
- Fast p-norm Flow Diffusion Algo
- Use to solve other problem?

  Nonnegative-Laplacian Paradigm?

$$\min_f \; ||f||_p$$

- $S + B^T f \leq T$

Thank You!