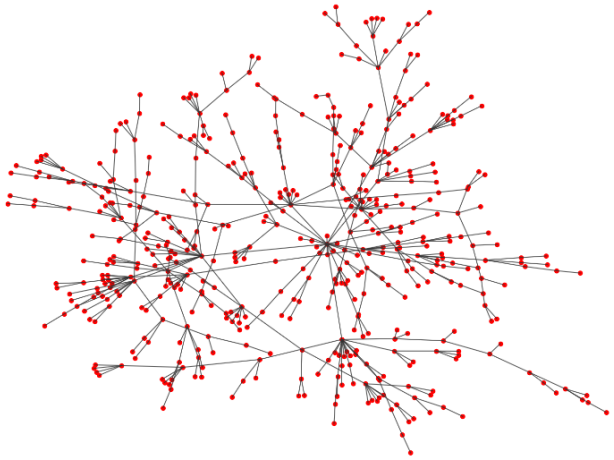
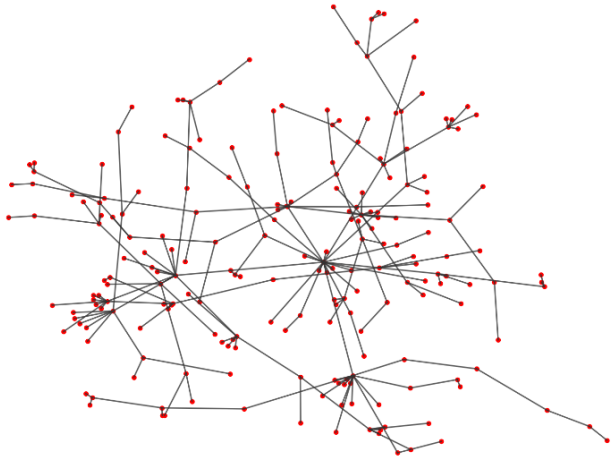
Graph reduction by local variation

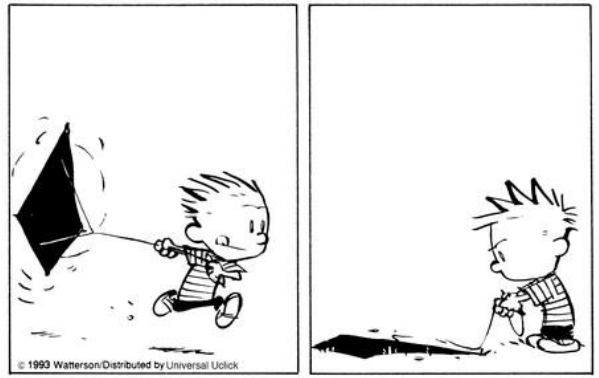


Andreas Loukas

École Polytechnique Fédérale de Lausanne



The joys/frustrations of graph data



In the graph world, data come in large sizes

- Wikipedia has 45 million articles
- Facebook has 721 million active users

Great news! … if only …

- Space is an issue
  - storage and retrieval is tedious
  - can't fit in memory
- Computationally
  - *Unsupervised algorithms* typically super-linear (e.g., clustering, embedding)
  - Even linear algorithms cannot be used repeatedly (e.g., this makes training GCN a frustrating task)
- Deriving meaning from large graphs is hard.

How to simplify graphs

A common solution:

a) Instead of solving the original (large) problem, solve a similar simpler problem;

b) Refine the solution (if needed)

To simplify graphs, we can either use

a) Sparsification: reduce the #edges M

- *spanners*, *cut* and *spectral* sparsifiers [Spielman & Teng 2011, Koutis et al 2011]
- Solve SDD linear systems in $\tilde{O}(M \log N)$ time.

b) Reduction: reduce the #vertices N, as well as M

- Graph coarsening, Kron reduction

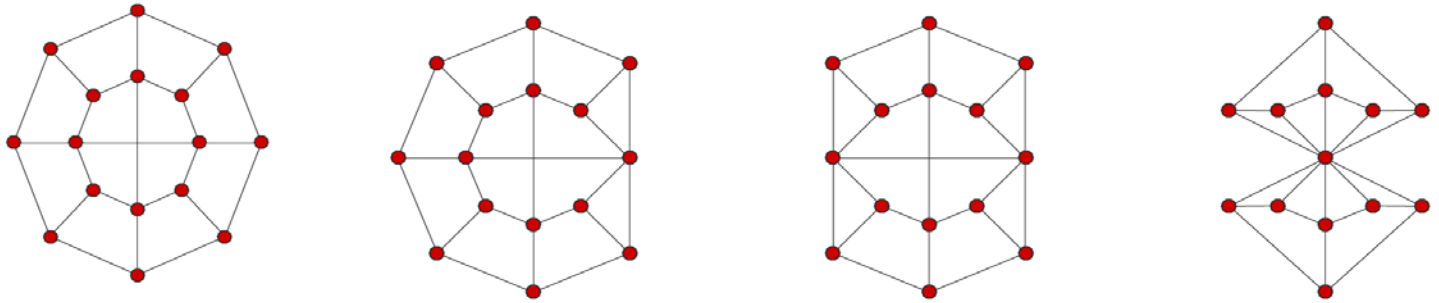
A brief history of coarsening

Coarsening is frequently used to make code scalable:

- Long list of algorithms use it for partitioning and visualizing large graphs.
  - [Hendrickson and Leland 1995; Karypis and Vipin 1998; Wang et al. *How to partition a billion-node graph,* 2014].
- Common way to create multi-scale representations of graph-structured data.
  - Coarse-grained diffusion maps [Lafon and Lee 2006], multi-scale wavelets [Gavish et al. 2010] and pyramids [Shuman et al. 2016].
- It as a component of graph convolutional networks (GCN) analogous to pooling.
  - [Bruna et al. 2014; Defferrard et al. 2016, Bronstein et al. 2017].
- It is used for solving linear systems on graphs.
  - Multigrid first adapted to graphs by [Koutis et al. 2011] and [Livne and Brandt 2012]
  - Extensions for Fiedler vector [Urschel et al. 2014; Gandhi 2016] and regression [Colley et al. 2017]

The elephant in the room

Obviously, we are losing information.



Coarsening even few vertices may deform the graph structure,
severely affecting the efficacy of graph algorithms.

- Can we still guarantee that G and G_c are similar (w.r.t. *distances*, *cuts*, or *spectrum*)?
- To-date, very little is known.
- Can we design coarsening algorithms with *guarantees* that are also *efficient*?

Today's menu

a) Study a generic graph reduction scheme

b) Introduce restricted similarity, a measure for graph approximation
- Inspired by spectral sparsifiers
- Strong implications for *spectrum* and unsupervised learning (*partitioning*, *clustering*, and *node embedding*)

c) Coarsening algorithms that maximize restricted similarity
- *Non-heuristically* defined
- *Nearly-linear* computational complexity

d) It works in practice
- Almost as fast as the fastest coarsening algorithm
- Approximate the spectrum of real graphs *2x-5x* better than best known algorithms.

Graph reduction and coarsening

Basics and restricted similarity

A generic graph reduction scheme

Let L be an $N \times N$ PSD matrix s.t. $L(i,j) \neq 0$ only if e_{ij} is an edge of $G = (\mathcal{V}, \mathcal{E}, W)$:

Scheme 1: Graph reduction

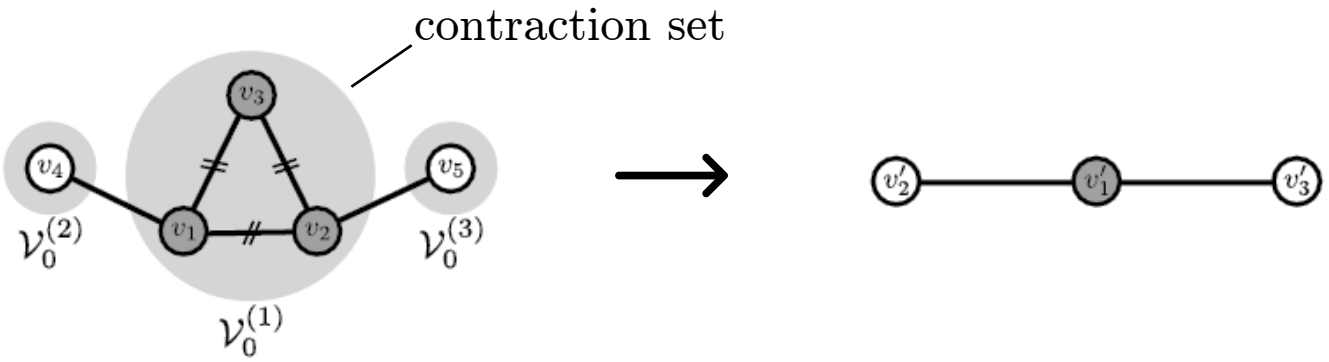
Commence by setting $L_0 = L$ and $x_0 = x$ and proceed according to the following two recursive equations:

$$L_\ell = P_\ell^{\mp} L_{\ell-1} P_\ell^{+} \quad \text{and} \quad x_\ell = P_\ell \, x_{\ell-1},$$

where $P_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ are matrices with more columns than rows, $\ell = 1, 2, \ldots, c$ is the level of the reduction, symbol $\mp$ denotes the transposed pseudoinverse, and N_ℓ is the dimensionality at level ℓ such that $N_0 = N$ and $N_c = n \ll N$.

Vector x_c is lifted back to $\mathbb{R}^N$ by recursion $\widetilde{x}_{\ell-1} = P_\ell^{+} \widetilde{x}_\ell$, where $\widetilde{x}_c = x_c$.

Graph coarsening as a special case

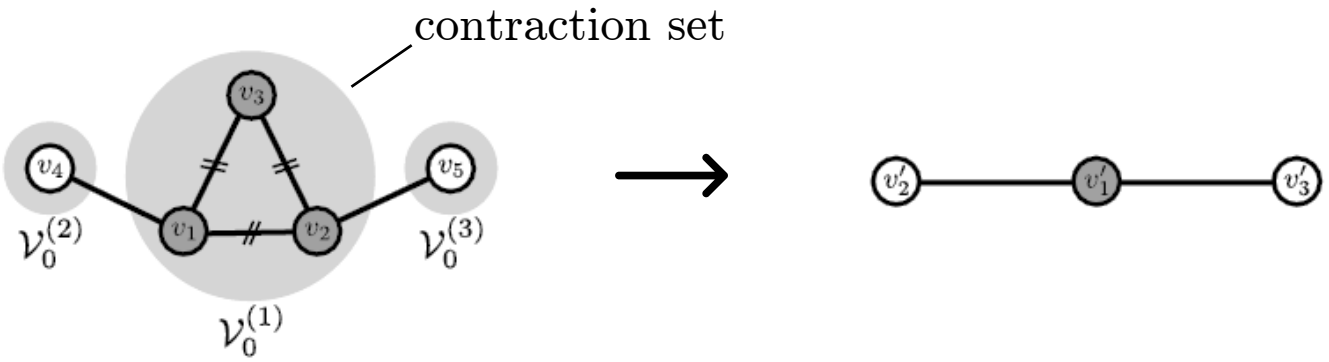


To coarsen $G_{\ell-1} = (\mathcal{V}_{\ell-1}, \mathcal{E}_{\ell-1})$ into $G_\ell = (\mathcal{V}_\ell, \mathcal{E}_\ell)$:

- Partition $G_{\ell-1}$ into N_ℓ connected subgraphs $G_{\ell-1}^{(r)} = (\mathcal{V}_{\ell-1}^{(r)}, \mathcal{E}_{\ell-1}^{(r)})$. Call $\mathcal{V}_{\ell-1}^{(r)}$ a contraction set.
- Form a vertex $v_r' \in \mathcal{V}_\ell$ for every contraction set.
- The weight of edge (v_r', v_p') is equal to $cut(\mathcal{V}_{\ell-1}^{(r)}, \mathcal{V}_{\ell-1}^{(p)})$

Toy coarsening example

L is the Laplacian

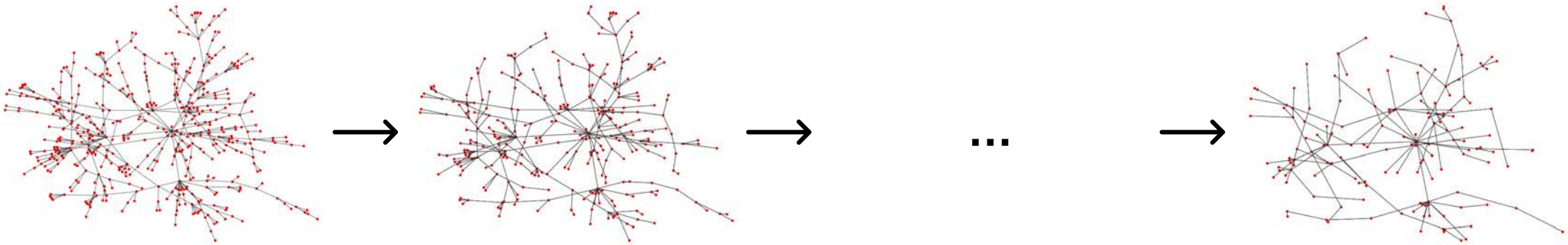


$$P_1 = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad P_1^+ = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \Pi = P_1^+ P_1 = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and coarsening results in

$$L_c = P_1^{\mp} L P_1^+ = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \quad x_c = P_1 x = \begin{bmatrix} (x(1) + x(2) + x(3))/3 \\ x(4) \\ x(5) \end{bmatrix}.$$

Benefits of multilevel coarsening



$N_0 = N$ vertices $\quad$ N_1 vertices $\quad$ $N_c = n$ vertices

We can now solve problem in $\mathbb{R}^n$ with $n \ll N$. If needed, the solution is lifted to $\mathbb{R}^N$ as $P_1^+ \cdots P_c^+ x$.

- Speed. Only $O(M)$ operations needed!! (trivial pseudo-inversion)
- Interpretability. Different from random projections, variables have a meaning.
- Simplicity. We can use the same algorithm to process G_c as with G.

Basic properties

One may express the reduced quantities in a more compact form:

$$x_c = Px, \quad L_c = P^{\mp} L P^{+} \quad \text{and} \quad \widetilde{x} = \Pi x,$$

where $P = P_c \cdots P_1 \in \mathbb{R}^{n \times N}$, $P^{+} = P_1^{+} \cdots P_c^{+} \in \mathbb{R}^{N \times n}$, and $\Pi = P^{+} P \in \mathbb{R}^{N \times N}$.

The *dimensionality reduction ratio* $r = 1 - \frac{n}{N}$ should be as large as possible.

Property 1. Π *is a projection matrix.*

Denote by λ and $\widetilde{\lambda}$, respectively, the eigenvalues of L and L_c.

Theorem 1. *The interlacing inequality* $\lambda_k \leq \widetilde{\lambda}_k$ *holds for all* $k = 1, \ldots, n$.

Interesting guarantees, but not strong enough ..

A different perspective: restricted similarity

Define the following matrix induced semi-norm:

$$\|x\|_L = \sqrt{x^\top L x}$$

Ideally, one would hope that

$$(1-\epsilon)\ \|x\|_L \leq \|x_c\|_{L_c} \leq (1+\epsilon)\ \|x\|_L \quad \forall x \in \mathbb{R}^N,$$

in which case we say that L_c and L are called ϵ-similar.

Definition 1 (Restricted similarity). *Let* $\mathbf{R}$ *be a* k*-dimensional subspace of* $\mathbb{R}^N$ *such that* $k \leq n \leq N$. *Matrices* L *and* L_c *are* $(\mathbf{R}, \epsilon)$*-similar if there exists an* $\epsilon \geq 0$ *such that*

$$\|x - \tilde{x}\|_L \leq \epsilon \|x\|_L \quad \forall x \in \mathbf{R}.$$

Implies $(1-\epsilon)\ \|x\|_L \leq \|x_c\|_{L_c} \leq (1+\epsilon)\ \|x\|_L \quad \forall x \in \mathbf{R}.$

Implications: spectrum is preserved

Consider the matrix of k first eigenvectors:

$$U_k \in \mathbb{R}^{N\times k} = [u_1, u_2, \ldots, u_k] \quad \text{and} \quad \mathbf{U}_k = \text{span}(U_k)$$

Theorem 1. *If L_c and L are $(\mathbf{U}_k, \epsilon_k)$-similar and $P^\top = P^+$, then*

$$\lambda_2 \leq \widetilde{\lambda}_2 \leq \frac{(1+\epsilon_2)^2}{1-\epsilon_2}\lambda_2 \quad \textit{and} \quad \lambda_k \leq \widetilde{\lambda}_k \leq \frac{(1+\epsilon_k)^2}{1-\|\Pi^\perp U_k\|_2^2}\lambda_k.$$

The angle [Davis and Kahan 1970] between eigenspaces U_k and $\widetilde{U}_k$ of L and $\widetilde{L}$, respectively is:

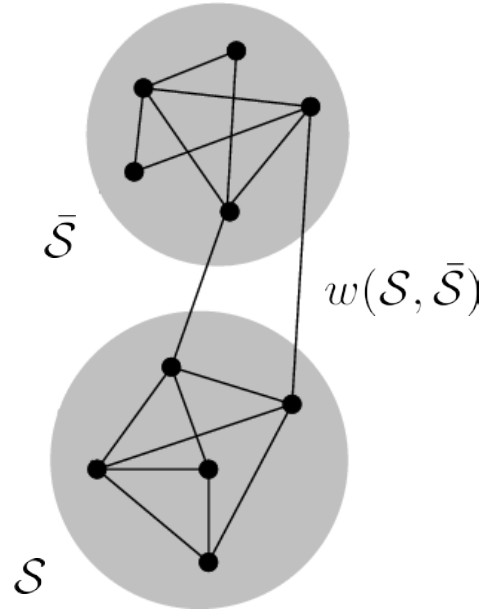
Theorem 2. *If L_c and L are $(\mathbf{U}_k, \epsilon_k)$-similar then*

$$\|\sin\Theta\left(U_k, P^\top\widetilde{U}_k\right)\|_F^2 \leq \sum_{i=1}^{k}\frac{\epsilon_i(2+\epsilon_i)\lambda_i + \lambda_k\epsilon_i}{\lambda_{k+1}-\lambda_k}.$$

Implications: good cuts are preserved

The conductance of G is defined as

$$\phi_2(G) = \min_{\mathcal{S} \subset \mathcal{V}} \frac{w(\mathcal{S}, \bar{\mathcal{S}})}{\min\{w(\mathcal{S}), w(\bar{\mathcal{S}})\}}$$



- $\bar{\mathcal{S}} = \mathcal{V} \setminus \mathcal{S}$ is the complement set,
- $w(\mathcal{S}, \bar{\mathcal{S}}) = \sum_{v_i \in \mathcal{S}, v_j \in \bar{\mathcal{S}}} w_{ij}$ is the weight of the cut and
- $w(\mathcal{S}) = \sum_{v_i \in \mathcal{S}} \sum_{v_j \in \mathcal{V}} w_{ij}$ is the volume of $\mathcal{S}$.

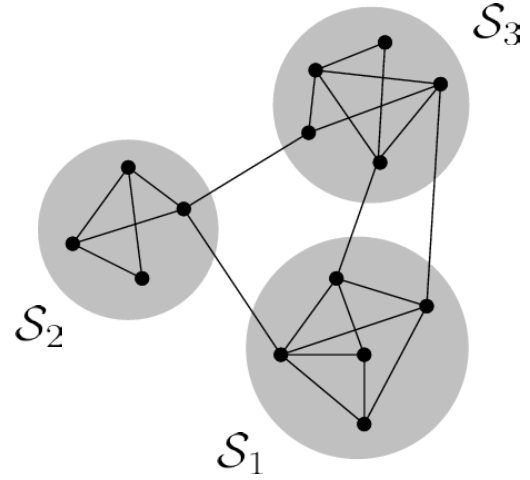
Corollary 1. *If L_c and L are $(\mathbf{U}_2, \epsilon_2)$-similar and $P^+ = P^\top$ then*

$$\phi_2(G) \leq \phi_2(G_c) \leq 2 \left(\frac{1 + \epsilon_2}{\sqrt{1 - \epsilon_2}} \right) \sqrt{\phi_2(G)}.$$

Implications: good multi-way cuts are preserved

The k-conductance is defined as

$$\phi_k(G) = \min_{\mathcal{S}_1,\ldots,\mathcal{S}_k} \max_i \phi_2(\mathcal{S}_i)$$



Theorem 3. *For any integer $2 \leq k \leq n$ and $t > 0$ such that $k' = (1+t)k \leq n$, if L_c and L are $(\mathbf{U}_{k'}, \epsilon_{k'})$-similar combinatorial Laplacian matrices and $P^+ = P^\top$ then*

$$\phi_k(G) \leq \phi_k(G_c) = O\left(\frac{1+\epsilon_{k'}}{\|\Pi U_{k'}\|_2}\, poly\,(1/t)\, \sqrt{\phi_{k'}(G)\log k}\right).$$

Implications: spectral clustering works

Spectral clustering. Partition G into $\mathcal{P} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\}$ [Shi and Malik]:

- Search for the partitioning $\mathcal{P}^*$ that minimizes the k-means cost:

$$\mathcal{F}(U_k, \mathcal{P}) = \left[\sum_{r=1}^{k} \sum_{v_i, v_j \in \mathcal{S}_r} \frac{\|U_k(i,:) - U_k(j,:)\|_2^2}{2\,|\mathcal{S}_r|} \right]^{\frac{1}{2}}$$

Spectral clustering + coarsening. [Karypis and Kumar] proposed the following scheme:

- Hierarchically coarsen G until the latter reaches a target size;
- Solve partitioning problem in the small dimension;
- Lift the solution back to the original domain, while performing some fast refinement.

Implications: spectral clustering works

Suppose that

$$\mathcal{P}^* = \arg\min_{\mathcal{P}} \mathcal{F}(U_k, \mathcal{P}) \quad \text{and} \quad \widetilde{\mathcal{P}}^* = \arg\min_{\mathcal{P}} \mathcal{F}(P^\top \widetilde{U}_k, \mathcal{P})$$

are the two identified clustering assignments.

[Boutsidis et al. 2015]: If $|\mathcal{F}(U_k, \mathcal{P}^*) - \mathcal{F}(U_k, \widetilde{\mathcal{P}}^*)|$ is small then $\widetilde{\mathcal{P}}^*$ and $\mathcal{P}^*$ are of the same quality.

Corollary 2. *If L_c and L are $(\mathbf{U}_k, \epsilon_k)$-similar then*

$$|\mathcal{F}(U_k, \widetilde{\mathcal{P}}^*) - \mathcal{F}(U_k, \mathcal{P}^*)| \leq 2\sqrt{2} \sum_{i=1}^{k} \frac{\epsilon_i(2+\epsilon_i)\lambda_i + \lambda_k \epsilon_i}{\widetilde{\lambda}_{k+1} - \lambda_k}$$

even without refinement.

Bisection (simple case):

- $|\mathcal{F}(U_2, \widetilde{\mathcal{P}}^*) - \mathcal{F}(U_2, \mathcal{P}^*)| = O(\epsilon_2 \frac{\lambda_2}{\lambda_3 - \lambda_2})$

Local variation algorithms

Finally, something practical

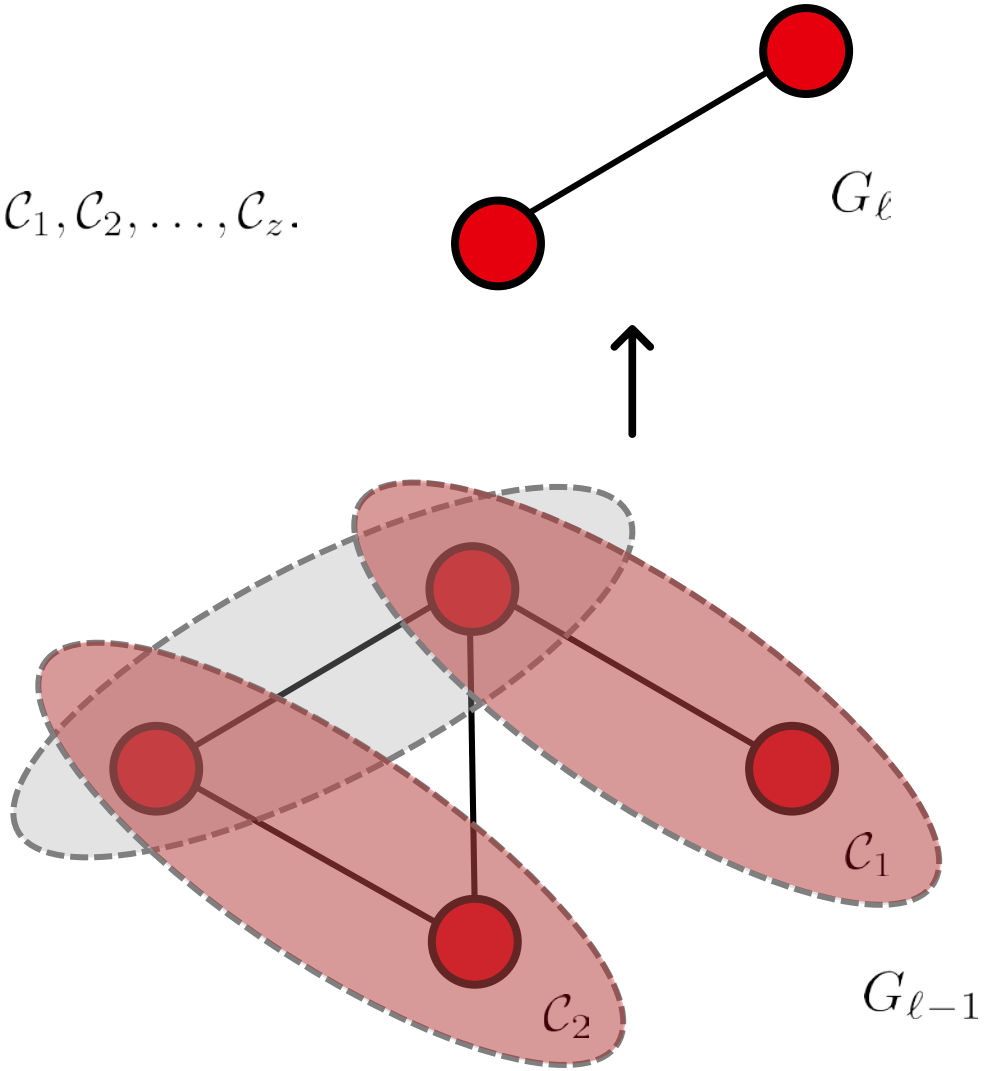
Idea

To go from level $\ell - 1$ to ℓ:

1. Find many, small, possibly overlapping candidate sets $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_z$.
2. Compute local variation cost $\sigma_\ell(G_{\mathcal{C}_i}, \mathbf{R})$ for each.
3. Select as many non-overlapping sets as you can, while $\sqrt{\sum_{\mathcal{C} \in \mathcal{P}_\ell} \sigma_\ell(G_{\mathcal{C}}, \mathbf{R})^2} \leq \sigma_\ell$
4. Contract and proceed to next level.

Theorem 1. *Matrices L_c and L are $(\mathbf{R}, \epsilon)$-similar with*

$$\epsilon = \prod_{\ell=1}^{c} (1 + \sigma_\ell) - 1.$$



Example algorithms

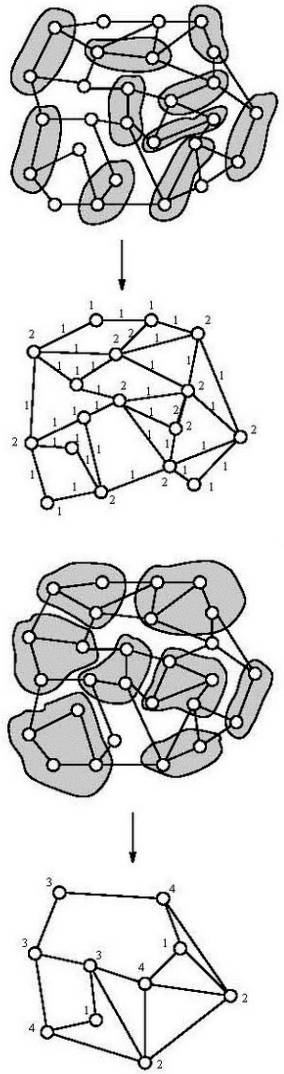
To go from $G_{\ell-1} = (\mathcal{V}_{\ell-1}, \mathcal{E}_{\ell-1})$ to $G_\ell = (\mathcal{V}_\ell, \mathcal{E}_\ell)$:

Edge-based local variation algorithm – takes $\tilde{O}(ckM + ck^3 + k^2N)$ time

- Compute the local variation cost $\sigma_\ell(e_{ij})$ of each edge $e_{ij} = (v_i, v_j) \in \mathcal{E}_{\ell-1}$.
- Find a minimum weight matching (each vertex belongs to one contraction set)
- Contract every edge in the matching.

Generally, we define any candidate family of sets (neighborhoods, cliques, ..) and apply the same idea:

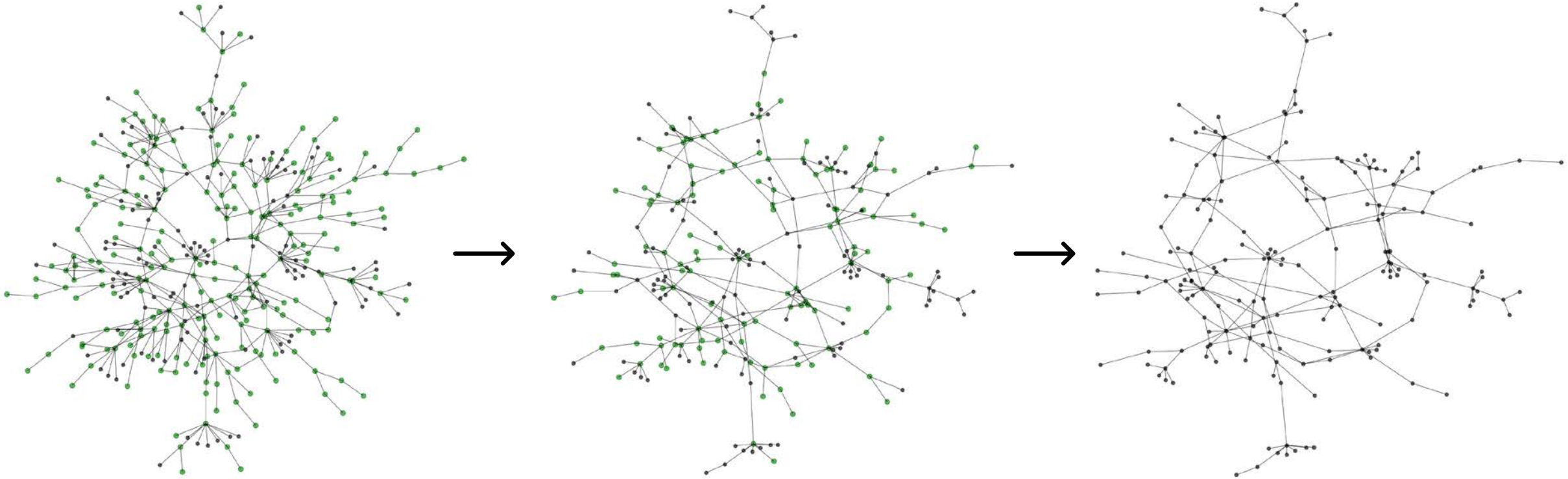
- Careful to keep the complexity small.
- **Neighborhood-based** algorithm takes $\tilde{O}(cM(k^2d + kd^2))$ time.



Visual & numerical results

It works!

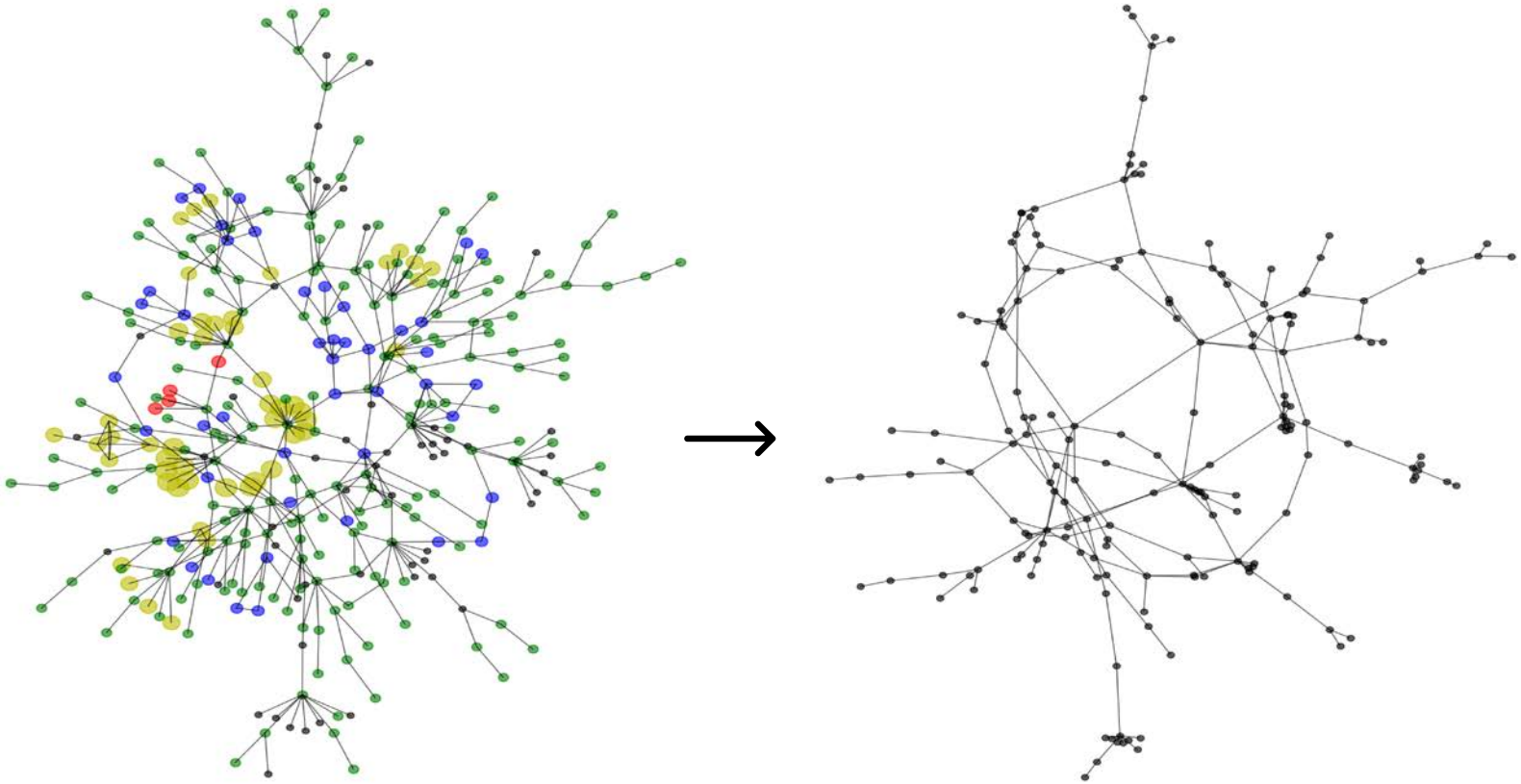
Yeast PPI network: edge-based local variation



$G = G_0$ G_1 $G_c = G_2$

Graph reduction by local variation

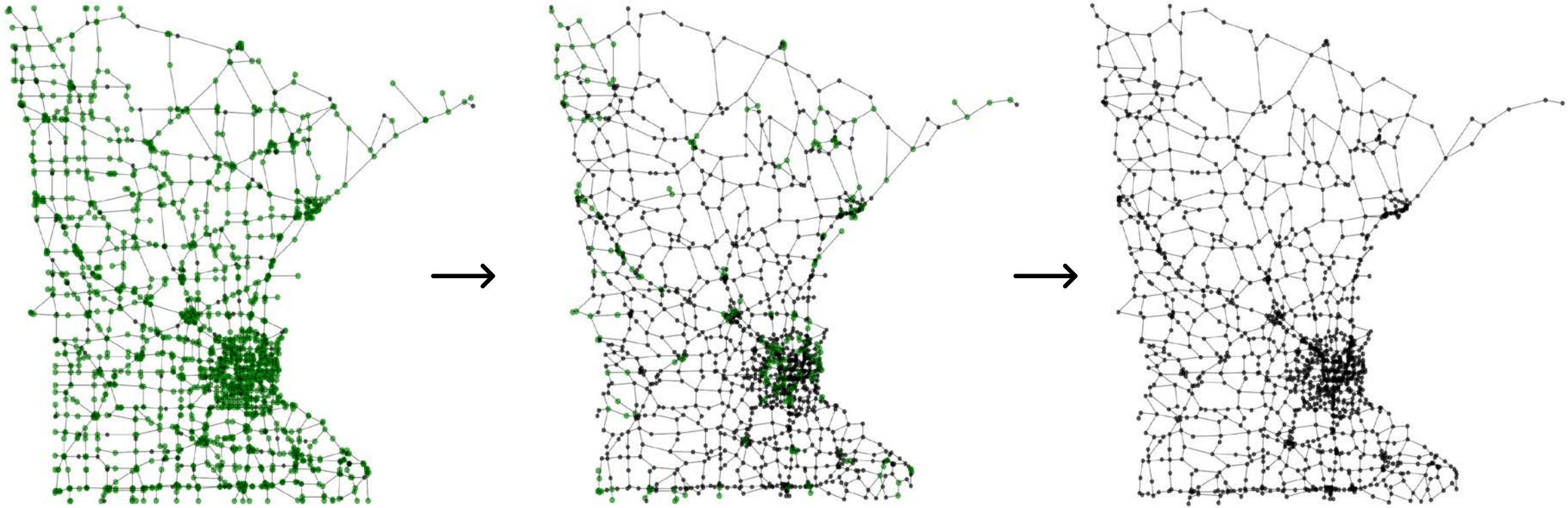
Yeast PPI network: neighborhood-based local variation



$G = G_0$ $G_c = G_1$

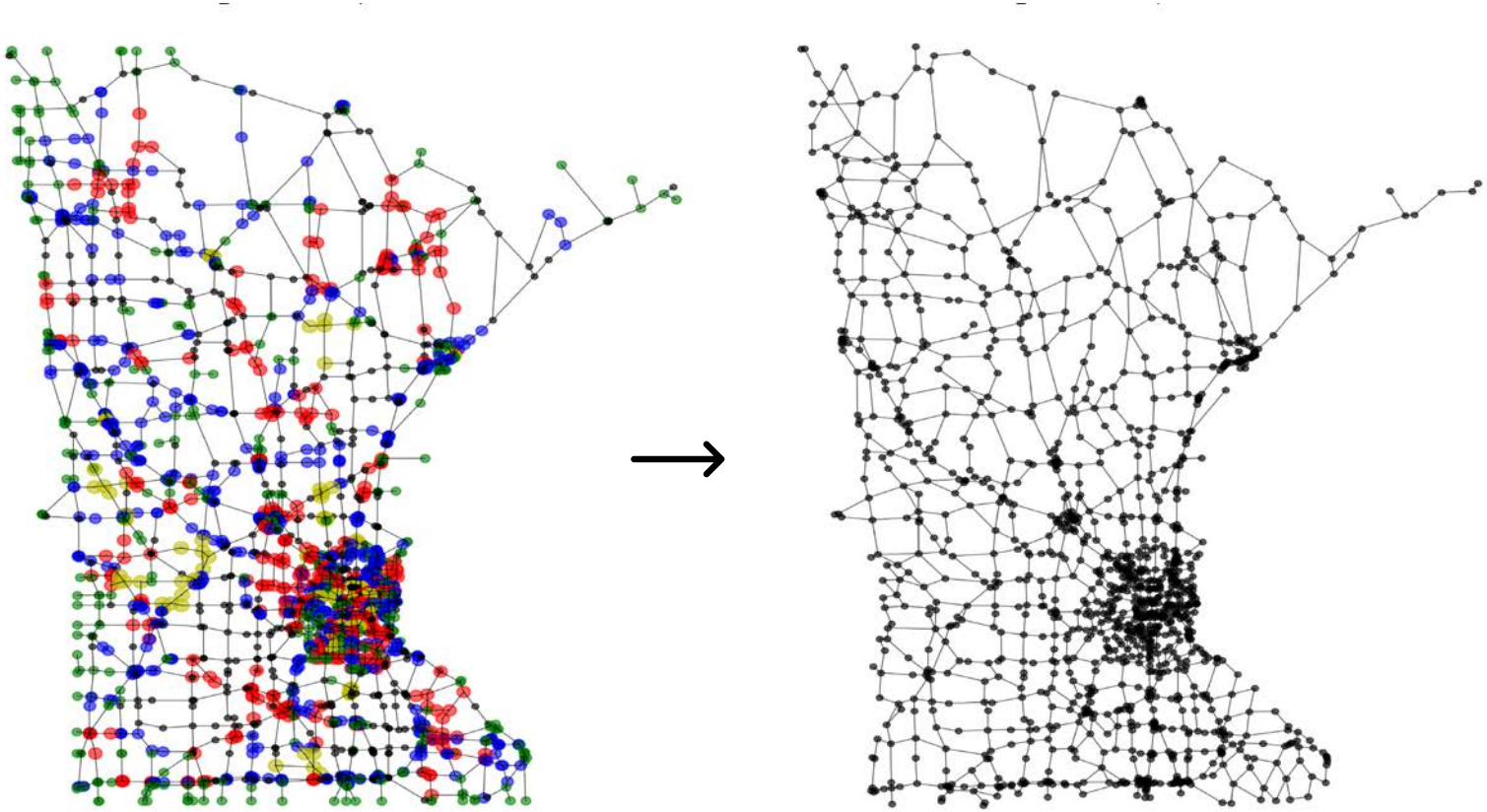
Graph reduction by local variation

Minnesota network: edge-based local variation



$G = G_0$ G_1 $G_c = G_2$

Minnesota network: neighborhood-based local variation



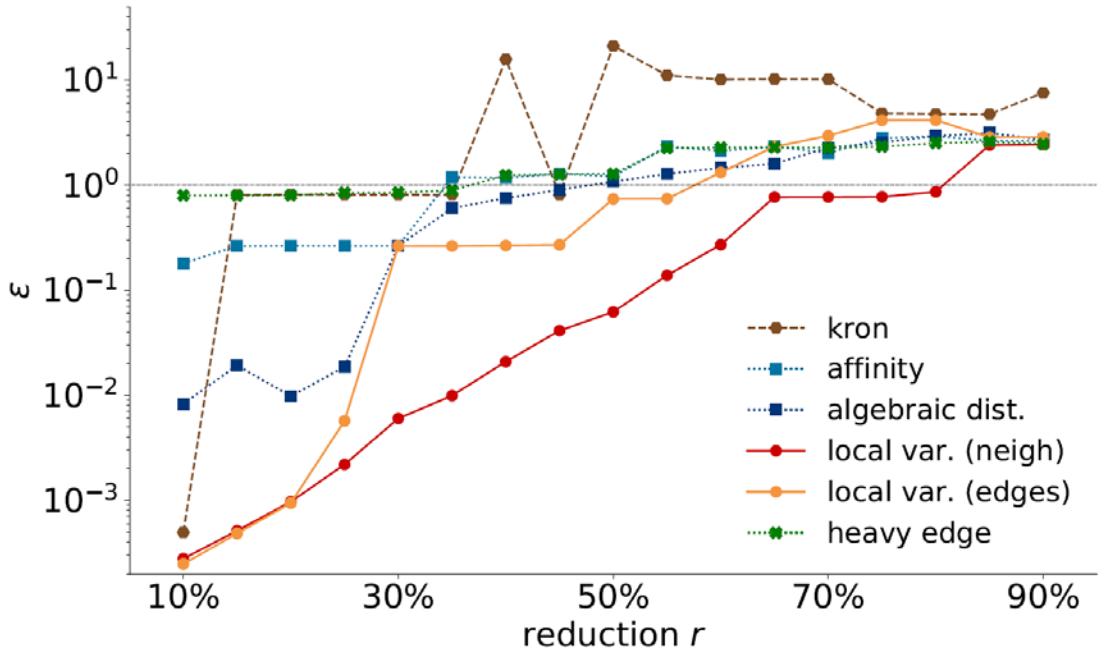
$G = G_0$ $G_c = G_1$

Graph reduction by local variation

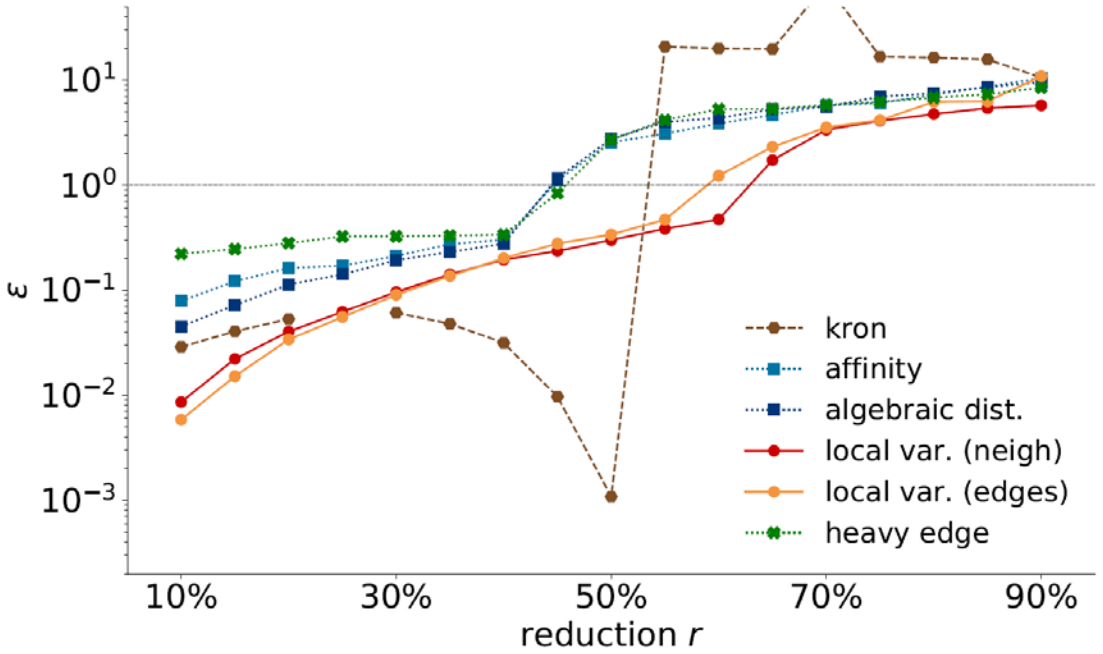
Restricted similarity experiment

Focusing on U_{10}

Yeast PPI



Minnesota network



- Up to 70% reduction feasible, results consistent for larger k

Spectrum approximation experiment

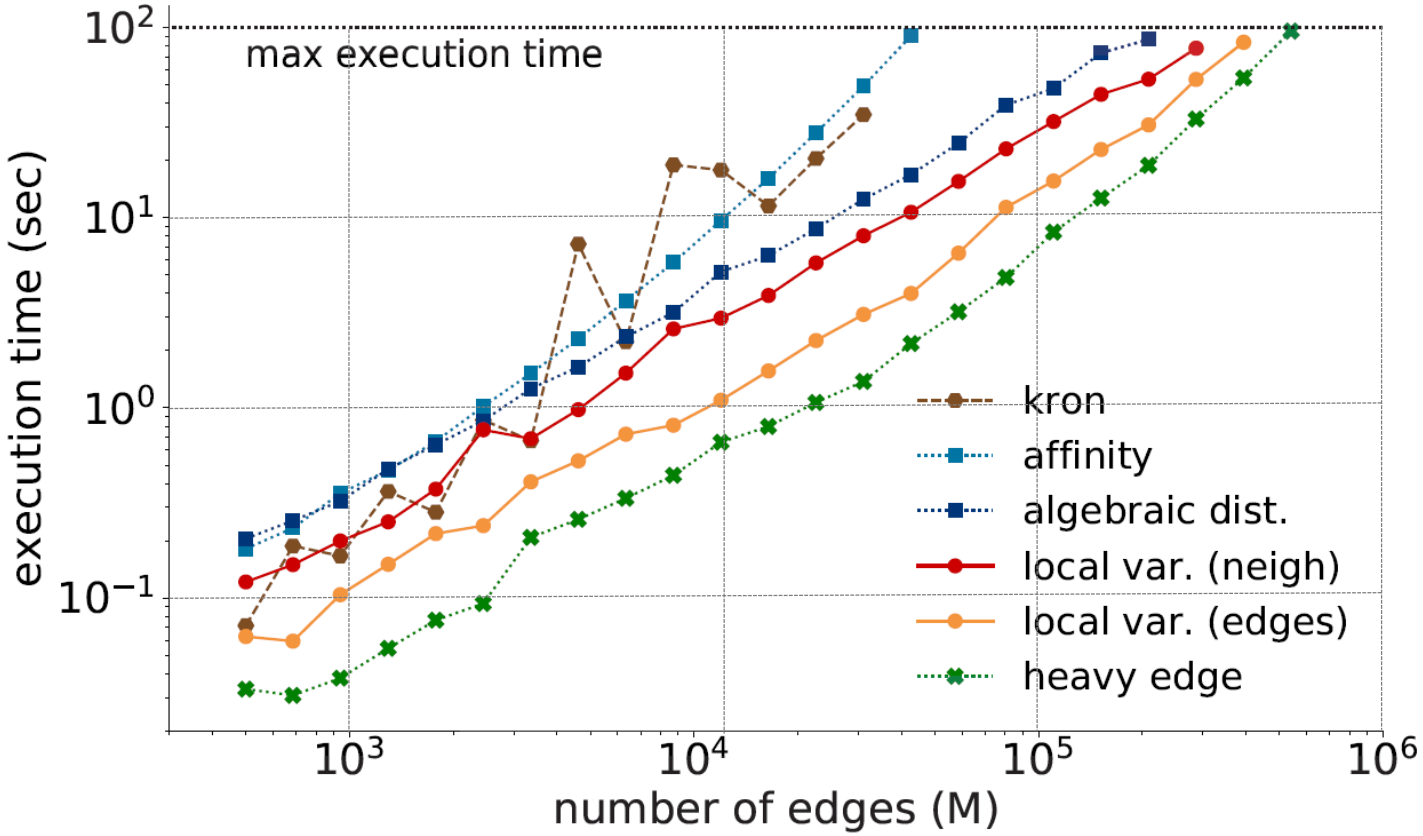
Average *eigenvalue error* for first 40 eigenvalues

	r	heavy edge	local var. (edges)	local var. (neigh.)	algebraic distance	affinity	Kron reduction
yeast	30%	0.328	0.113	**0.023**	0.094	0.195	0.120
	50%	0.879	0.430	**0.130**	0.517	0.769	1.196
	70%	2.498	2.182	**0.451**	2.560	2.229	1.946
airfoil	30%	0.277	**0.095**	0.181	0.189	0.267	0.368
	50%	0.549	**0.325**	0.349	0.698	0.862	0.960
	70%	2.268	0.872	**0.839**	2.373	2.531	2.078
bunny	30%	0.023	**0.008**	0.085	0.205	0.052	0.294
	50%	0.066	**0.058**	0.181	0.346	0.089	0.660
	70%	0.128	**0.098**	0.299	0.509	0.202	1.192
minnesota	30%	0.353	0.118	**0.115**	0.209	0.306	0.337
	50%	1.259	0.468	**0.383**	1.342	1.264	0.933
	70%	4.162	2.111	**1.612**	4.145	4.185	2.090

$$\text{error} = \frac{1}{40} \sum_{i \leq 40} \frac{\tilde{\lambda}_i - \lambda_i}{\lambda_i}$$

- Same trends as previous experiment
- On average 2.6x/3.9x smaller error compared to the second best method (w/wo Kron)

Scalability experiment



- Almost as fast as naïve heavy-edge matching

Summary

State of the art

- Graph reduction is commonly used in modern graph processing pipelines (METIS, GEPHI).
- Nevertheless, convincing analysis/algorithms is lacking.

Contributions

- Study a generic reduction scheme, featuring coarsening as a special case.
- Strong spectral and cut guarantees.
- Analysis holds for multiple levels of reduction.
- First non-heuristic algorithms for graph coarsening.

Open questions

- Lower bounds: what are the limits of reduction?
- Algorithms: how close can we achieve this limit in nearly-linear time?

Extra slides

Details are important