Short Cycles via Low Diameter Decomposition

Yang Liu
Stanford University

Joint work with Sushant Sachdeva, Zejun Yu
University of Toronto

Short Cycle Decomposition

Short Cycle Decomposition

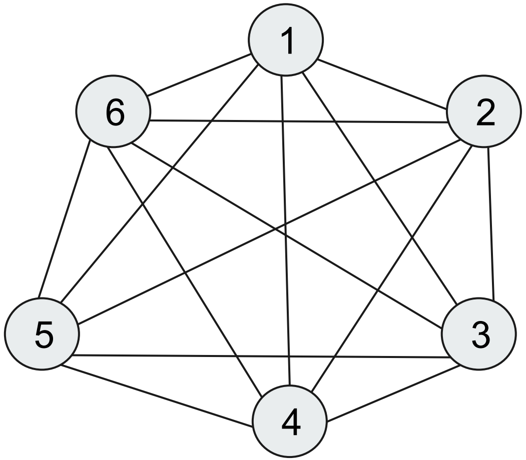
(k, L) short cycle decomposition of an undirected, unweighted graph G

Decomposition of edges of G into edge disjoint cycles of length <= L and at most k extra edges.

Short Cycle Decomposition

(k, L) short cycle decomposition of an undirected, unweighted graph G

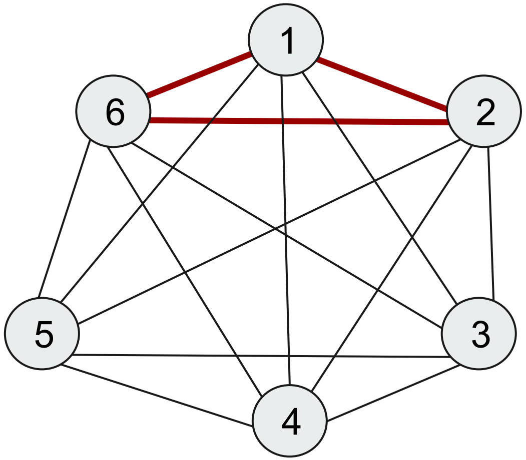
Decomposition of edges of G into edge disjoint cycles of length <= L and at most k extra edges.



Short Cycle Decomposition

(k, L) short cycle decomposition of an undirected, unweighted graph G

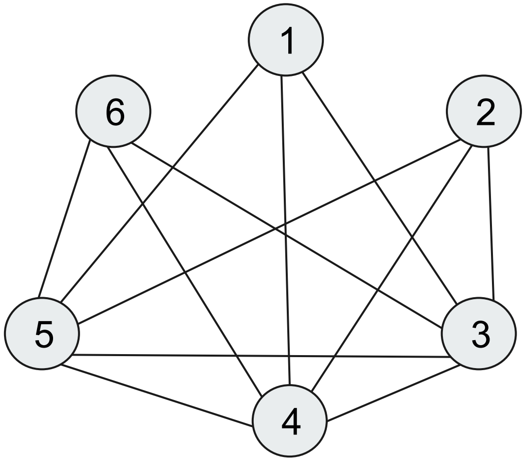
Decomposition of edges of G into edge disjoint cycles of length <= L and at most k extra edges.



Short Cycle Decomposition

(k, L) short cycle decomposition of an undirected, unweighted graph G

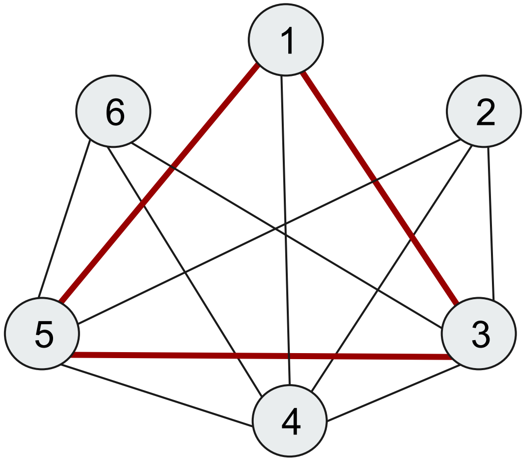
Decomposition of edges of G into edge disjoint cycles of length <= L and at most k extra edges.



Short Cycle Decomposition

(k, L) short cycle decomposition of an undirected, unweighted graph G

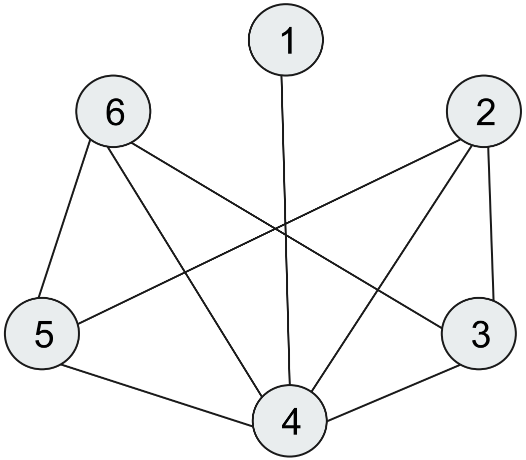
Decomposition of edges of G into edge disjoint cycles of length <= L and at most k extra edges.



Short Cycle Decomposition

(k, L) short cycle decomposition of an undirected, unweighted graph G

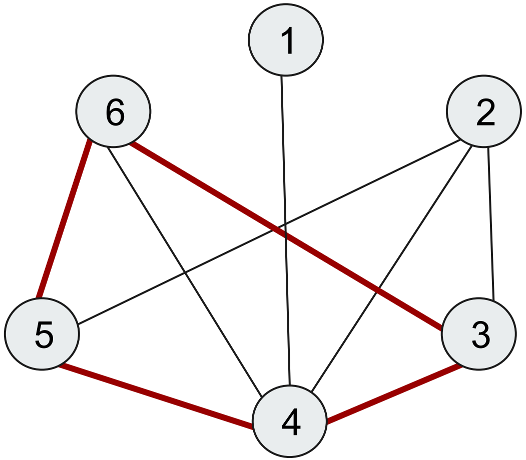
Decomposition of edges of G into edge disjoint cycles of length <= L and at most k extra edges.



Short Cycle Decomposition

(k, L) short cycle decomposition of an undirected, unweighted graph G

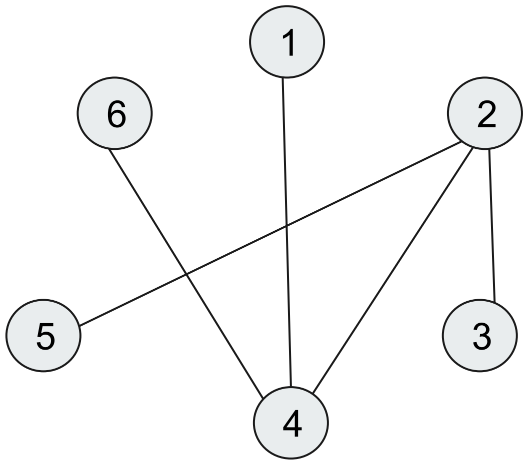
Decomposition of edges of G into edge disjoint cycles of length <= L and at most k extra edges.



Short Cycle Decomposition

(k, L) short cycle decomposition of an undirected, unweighted graph G

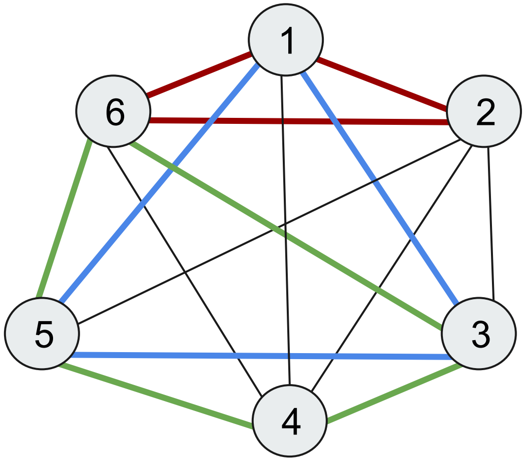
Decomposition of edges of G into edge disjoint cycles of length <= L and at most k extra edges.



Short Cycle Decomposition

(k, L) short cycle decomposition of an undirected, unweighted graph G

Decomposition of edges of G into edge disjoint cycles of length <= L and at most k extra edges.



Sparsification

Sparsification

Approximate some property of a graph G with sparse subgraph H.

Sparsification

Approximate some property of a graph G with sparse subgraph H.

Cut Sparsifier [BK96]**:** for any set $S, \mathrm{cut}_G(S) \approx_\epsilon \mathrm{cut}_H(S)$

Sparsification

Approximate some property of a graph G with sparse subgraph H.

Cut Sparsifier [BK96]**:** for any set $S, \mathrm{cut}_G(S) \approx_\epsilon \mathrm{cut}_H(S)$

Spanner [Che89]**:** for any pair of vertices u, v we have $d_G(u,v) \leq \alpha \cdot d_H(u,v)$

Sparsification

Approximate some property of a graph G with sparse subgraph H.

Sparsification

Approximate some property of a graph G with sparse subgraph H.

Spectral: $\forall x \in \mathbb{R}^n, (1-\epsilon)x^T L_G x \leq x^T L_H x \leq (1+\epsilon)x^T L_G x$

Sparsification

Approximate some property of a graph G with sparse subgraph H.

Spectral: $\forall x \in \mathbb{R}^n, (1-\epsilon)x^T L_G x \leq x^T L_H x \leq (1+\epsilon)x^T L_G x$

Laplacian: $L_G = D_G - A_G$

$$x^T L_G x = \sum_{(u,v)\in E(G)} w_{uv}(x_u - x_v)^2$$

Applications of Spectral Sparsification

Nearly Linear time Laplacian Solvers [ST04, ST14, KMP14, KMP11]

Cut and flow approximation algorithms [She09, She13, CKM+11, KLOS13, Peng16]

Random spanning tree generation [DKP+17]

Estimating determinants + spanning tree counts [DPPR17]

Spectral Sparsification: What's Known

Graph G with n vertices and m edges

Spectral Sparsification: What's Known

Graph G with n vertices and m edges

Nearly linear time spectral sparsifier H with $\tilde{O}(n\epsilon^{-2})$ edges [ST11, SS11]

Spectral Sparsification: What's Known

Graph G with n vertices and m edges

Nearly linear time spectral sparsifier H with $\tilde{O}(n\epsilon^{-2})$ edges [ST11, SS11]

Construction of spectral sparsifier H with $O(n\epsilon^{-2})$ edges [BSS09, BSS12]

Spectral Sparsification: What's Known

Graph G with n vertices and m edges

Nearly linear time spectral sparsifier H with $\tilde{O}(n\epsilon^{-2})$ edges [ST11, SS11]

Construction of spectral sparsifier H with $O(n\epsilon^{-2})$ edges [BSS09, BSS12]

$\Omega(n\epsilon^{-2})$ is optimal, even for arbitrary data structures with cut size queries [BSS12, CKST17]

Spectral Sparsification: new directions

Spectral Sparsification: new directions

Spectral Sketches: for an unknown fixed $x \in \mathbb{R}^n$, $x^T L_G x \approx_\epsilon x^T L_H x$ [ACK+16]

Spectral Sparsification: new directions

Spectral Sketches: for an unknown fixed $x \in \mathbb{R}^n, x^T L_G x \approx_\epsilon x^T L_H x$ [ACK+16]

[JS18]: Data structure with $\tilde{O}(n\epsilon^{-1})$ size

Spectral Sparsification: new directions

Spectral Sketches: for an unknown fixed $x \in \mathbb{R}^n, x^T L_G x \approx_\epsilon x^T L_H x$ [ACK+16]

Spectral Sparsification: new directions

Spectral Sketches: for an unknown fixed $x \in \mathbb{R}^n, x^T L_G x \approx_\epsilon x^T L_H x$ [ACK+16]

Resistance sparsifiers: for all vertices u, v, $\mathrm{Reff}_G(u,v) \approx_\epsilon \mathrm{Reff}_H(u,v)$ [DKW15]

Spectral Sparsification: new directions

Spectral Sketches: for an unknown fixed $x \in \mathbb{R}^n$, $x^T L_G x \approx_\epsilon x^T L_H x$ [ACK+16]

Resistance sparsifiers: for all vertices u, v, $\mathrm{Reff}_G(u,v) \approx_\epsilon \mathrm{Reff}_H(u,v)$ [DKW15]

Effective resistance (Reff) is quadratic form wrt Laplacian pseudoinverse

Spectral Sparsification: new directions

Spectral Sketches: for an unknown fixed $x \in \mathbb{R}^n$, $x^T L_G x \approx_\epsilon x^T L_H x$ [ACK+16]

Resistance sparsifiers: for all vertices u, v, $\mathrm{Reff}_G(u,v) \approx_\epsilon \mathrm{Reff}_H(u,v)$ [DKW15]

Spectral Sparsification: new directions

Spectral Sketches: for an unknown fixed $x \in \mathbb{R}^n, x^T L_G x \approx_\epsilon x^T L_H x$ [ACK+16]

Resistance sparsifiers: for all vertices u, v, $\text{Reff}_G(u, v) \approx_\epsilon \text{Reff}_H(u, v)$ [DKW15]

[DKW15]: Conjecture that H only needs $\tilde{O}(n\epsilon^{-1})$ edges

Spectral Sparsification: new directions

Spectral Sketches: for an unknown fixed $x \in \mathbb{R}^n, x^T L_G x \approx_\epsilon x^T L_H x$ [ACK+16]

Resistance sparsifiers: for all vertices u, v, $\mathrm{Reff}_G(u, v) \approx_\epsilon \mathrm{Reff}_H(u, v)$ [DKW15]

Spectral Sparsification: new directions

Spectral Sketches: for an unknown fixed $x \in \mathbb{R}^n$, $x^T L_G x \approx_\epsilon x^T L_H x$ [ACK+16]

Resistance sparsifiers: for all vertices u, v, $\mathrm{Reff}_G(u,v) \approx_\epsilon \mathrm{Reff}_H(u,v)$ [DKW15]

Sparsifying Eulerian directed graphs (directed graphs where all vertices have equal weighted in/outdegree) [CKP+17]

Spectral Sparsification: new directions

Spectral Sketches: for an unknown fixed $x \in \mathbb{R}^n$, $x^T L_G x \approx_\epsilon x^T L_H x$ [ACK+16]

Resistance sparsifiers: for all vertices u, v, $\mathrm{Reff}_G(u, v) \approx_\epsilon \mathrm{Reff}_H(u, v)$ [DKW15]

Sparsifying Eulerian directed graphs (directed graphs where all vertices have equal weighted in/outdegree) [CKP+17]

[CKP+17]: Applications to Laplacian solvers for directed graphs

Spectral Sparsification: new directions

Spectral Sketches: for an unknown fixed $x \in \mathbb{R}^n$, $x^T L_G x \approx_\epsilon x^T L_H x$ [ACK+16]

Resistance sparsifiers: for all vertices u, v, $\mathrm{Reff}_G(u, v) \approx_\epsilon \mathrm{Reff}_H(u, v)$ [DKW15]

Sparsifying Eulerian directed graphs (directed graphs where all vertices have equal weighted in/outdegree) [CKP+17]

Spectral Sparsification: new directions

Spectral Sketches: for an unknown fixed $x \in \mathbb{R}^n, x^T L_G x \approx_\epsilon x^T L_H x$ [ACK+16]

Resistance sparsifiers: for all vertices u, v, $\text{Reff}_G(u,v) \approx_\epsilon \text{Reff}_H(u,v)$ [DKW15]

Sparsifying Eulerian directed graphs (directed graphs where all vertices have equal weighted in/outdegree) [CKP+17]

Short Cycle Decomposition was introduced in [CGP+18] to make progress on problems such as the above

Short Cycle Decomposition: What's known

Short Cycle Decomposition: What's known

Theorem [CGP+18]: There is an algorithm running in time $m \cdot \exp(O(\log n)^{\frac{3}{4}})$ which produces a $\left(n \cdot \exp(O(\log n)^{\frac{1}{2}}, \exp(O(\log n)^{\frac{3}{4}})\right)$ short cycle decomposition.

Short Cycle Decomposition: What's known

Theorem [CGP+18]: There is an algorithm running in time $m \cdot \exp(O(\log n)^{\frac{3}{4}})$ which produces a $\left(n \cdot \exp(O(\log n)^{\frac{1}{2}}, \exp(O(\log n)^{\frac{3}{4}})\right)$ short cycle decomposition.

Extra edges

Cycle length

Short Cycle Decomposition: What's known

Theorem [CGP+18]: There is an algorithm running in time $m \cdot \exp(O(\log n)^{\frac{3}{4}})$ which produces a $\left(n \cdot \exp(O(\log n)^{\frac{1}{2}}, \exp(O(\log n)^{\frac{3}{4}})\right)$ short cycle decomposition.

Short Cycle Decomposition: Applications [CGP+18]

Short Cycle Decomposition: Applications [CGP+18]

Graphical spectral sketches and Resistance Sparsifiers

Short Cycle Decomposition: Applications [CGP+18]

Graphical spectral sketches and Resistance Sparsifiers

Estimating effective resistances

Short Cycle Decomposition: Applications [CGP+18]

Graphical spectral sketches and Resistance Sparsifiers

Estimating effective resistances

Degree-preserving sparsifiers

Short Cycle Decomposition: Applications [CGP+18]

Graphical spectral sketches and Resistance Sparsifiers

Estimating effective resistances

Degree-preserving sparsifiers

Eulerian directed graph sparsifiers

Short Cycle Decomposition: Applications [CGP+18]

Graphical spectral sketches and Resistance Sparsifiers

Estimating effective resistances

Degree-preserving sparsifiers

Eulerian directed graph sparsifiers

Improvements to the short cycle decomposition algorithm in [CGP+18] give immediate improvements for these applications

Our Results

Our Results

Theorem [LSY19]: For any constant $\delta \leq \frac{1}{2}$, algorithm running in time $O(mn^{\delta})$ for $\left(O(n), O(\log n)^{\frac{1}{\delta}-1}\right)$ short cycle decomposition

Our Results

Theorem [LSY19]: For any constant $\delta \leq \frac{1}{2}$, algorithm running in time $O(mn^{\delta})$ for $\left(O(n), O(\log n)^{\frac{1}{\delta}-1}\right)$ short cycle decomposition

Theorem [LSY19]: Algorithm running in time $m \cdot \exp(O(\log n)^{\frac{1}{2}})$ for $\left(O(n), \exp(O(\log n)^{\frac{1}{2}})\right)$ short cycle decomposition

Our Results

Our Results

Improvements to all of:

Our Results

Improvements to all of:

Graphical spectral sketches and Resistance Sparsifiers

Estimating effective resistances

Degree-preserving sparsifiers

Eulerian directed graph sparsifiers

Our Results

Our Results

In our opinion, algorithm is simpler

Our Results

In our opinion, algorithm is simpler

Uses low diameter decomposition [LS90], instead of expander decomposition (used in [CGP+18])

Our Results

In our opinion, algorithm is simpler

Uses low diameter decomposition [LS90], instead of expander decomposition (used in [CGP+18])

First almost linear time Eulerian graph sparsification algorithm *without expander decomposition*.

Naive Short Cycle Decomposition

Naive Short Cycle Decomposition

Quadratic time algorithm for $(2n, 2\log n)$ short cycle decomposition

Naive Short Cycle Decomposition

Quadratic time algorithm for $(2n, 2\log n)$ short cycle decomposition

Intuition: low depth spanning trees

Naive Short Cycle Decomposition

Quadratic time algorithm for $(2n, 2\log n)$
short cycle decomposition

Intuition: low depth spanning trees

Delete vertices until all vertices have degree >= 3

Naive Short Cycle Decomposition

Quadratic time algorithm for $(2n, 2\log n)$ short cycle decomposition

Intuition: low depth spanning trees

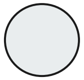
Delete vertices until all vertices have degree >= 3

BFS from anywhere to find a short cycle

Naive Short Cycle Decomposition

Quadratic time algorithm for $(2n, 2\log n)$
short cycle decomposition

Intuition: low depth spanning trees



Delete vertices until all vertices have degree >= 3

BFS from anywhere to find a short cycle

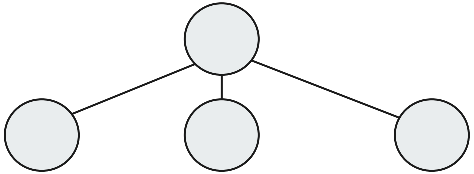
Naive Short Cycle Decomposition

Quadratic time algorithm for $(2n, 2\log n)$ short cycle decomposition

Intuition: low depth spanning trees

Delete vertices until all vertices have degree >= 3

BFS from anywhere to find a short cycle



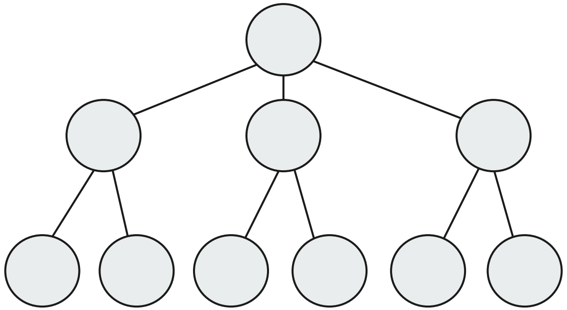
Naive Short Cycle Decomposition

Quadratic time algorithm for $(2n, 2\log n)$ short cycle decomposition

Intuition: low depth spanning trees

Delete vertices until all vertices have degree >= 3

BFS from anywhere to find a short cycle



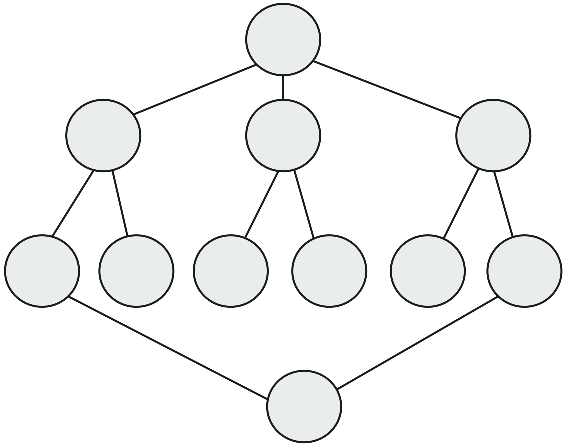
Naive Short Cycle Decomposition

Quadratic time algorithm for $(2n, 2\log n)$ short cycle decomposition

Intuition: low depth spanning trees

Delete vertices until all vertices have degree >= 3

BFS from anywhere to find a short cycle



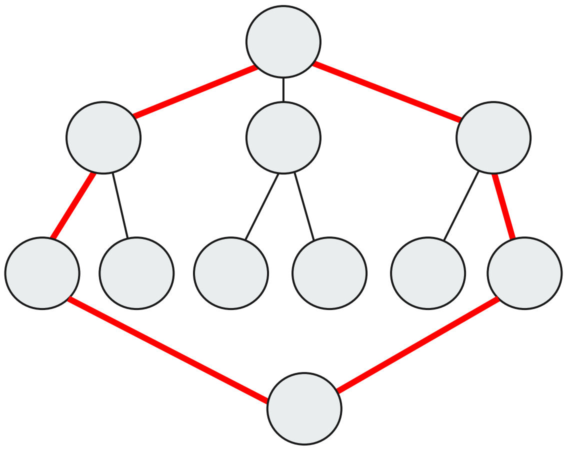
Naive Short Cycle Decomposition

Quadratic time algorithm for $(2n, 2\log n)$ short cycle decomposition

Intuition: low depth spanning trees

Delete vertices until all vertices have degree >= 3

BFS from anywhere to find a short cycle



Naive Short Cycle Decomposition

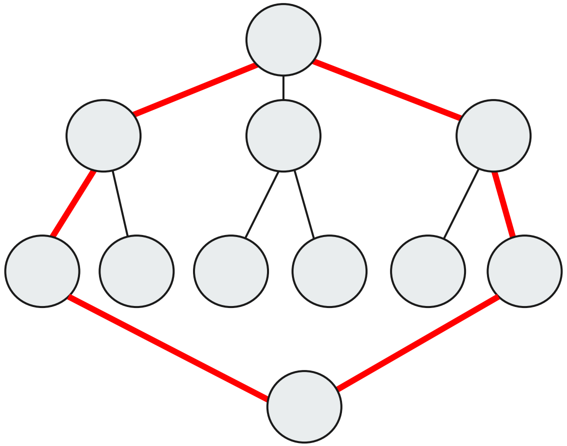
Quadratic time algorithm for $(2n, 2\log n)$ short cycle decomposition

Intuition: low depth spanning trees

Delete vertices until all vertices have degree >= 3

BFS from anywhere to find a short cycle

Delete cycle and repeat



Reduction to sparse, bounded degree graphs

Reduction to sparse, bounded degree graphs

Graph G with n vertices and m edges

Reduction to sparse, bounded degree graphs

Graph G with n vertices and m edges

$m = 10n$

Reduction to sparse, bounded degree graphs

Graph G with n vertices and m edges

m = 10n

Maximum degree is bounded (say at most 40).

Reduction to sparse, bounded degree graphs

Graph G with n vertices and m edges

m = 10n

Maximum degree is bounded (say at most 40).

Lemma (informal): If we can do $(O(n), O(\log n)^c)$ short cycle decomposition on such graphs efficiently, we can do it on all graphs.

Main Theorem

Main Theorem

G has n vertices, $m = 10n$ edges.

Main Theorem

G has n vertices, $m = 10n$ edges.

Maximum degree Δ

Main Theorem

G has n vertices, m = 10n edges.

Maximum degree Δ

[LSY19]: For any integer $c \geq 1$ we can find in time $O(500^c m n^{\frac{1}{c+1}})$ *vertex-disjoint* cycles of length $O(\log n)^c$ containing $\frac{m}{10\Delta}$ total vertices.

Preliminaries to algorithm

Preliminaries to algorithm

Vertex-disjoint analogue of naive cycle decomposition

Preliminaries to algorithm

Vertex-disjoint analogue of naive cycle decomposition

In time $O(n^2)$ we can find vertex-disjoint cycles of length $O(\log n)$ containing $\frac{m}{10\Delta}$ vertices.

Preliminaries to algorithm

Vertex-disjoint analogue of naive cycle decomposition

In time $O(n^{\frac{3}{2}})$ we can find vertex-disjoint cycles of length $O(\log n)$ containing $\frac{m}{10\Delta}$ vertices.

Preliminaries to algorithm

Vertex-disjoint analogue of naive cycle decomposition

Preliminaries to algorithm

Vertex-disjoint analogue of naive cycle decomposition

Low diameter decomposition [LS90] (analogue to low depth tree in naive cycle decomposition)

Preliminaries to algorithm

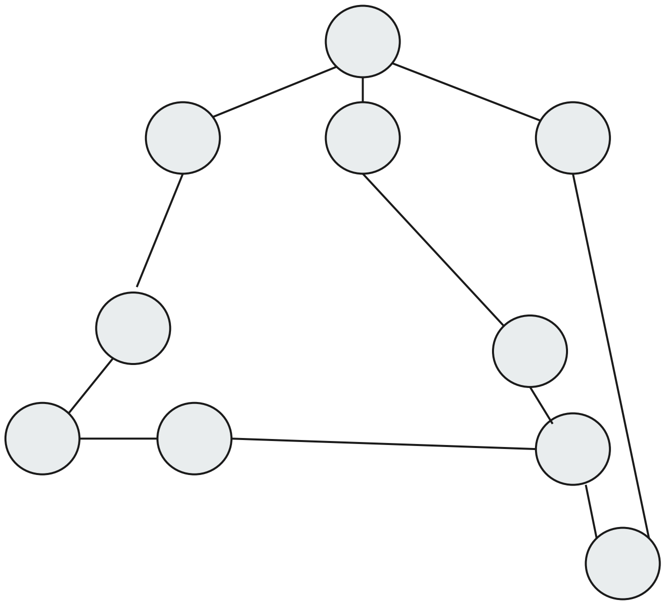
Vertex-disjoint analogue of naive cycle decomposition

Low diameter decomposition [LS90] (analogue to low depth tree in naive cycle decomposition)

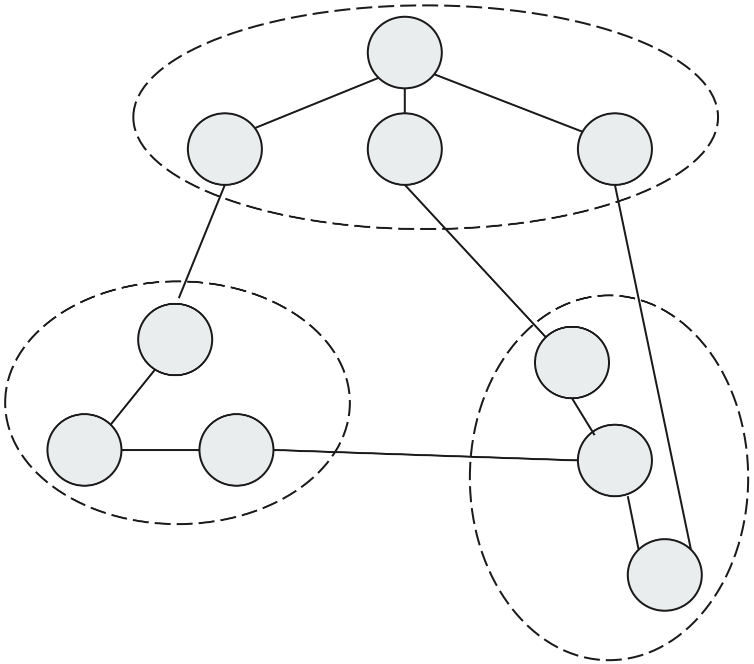
[MPX13]: For any parameter β we can remove βm edges to make each remaining connected component have diameter $O(\beta^{-1} \log n)$

Contraction

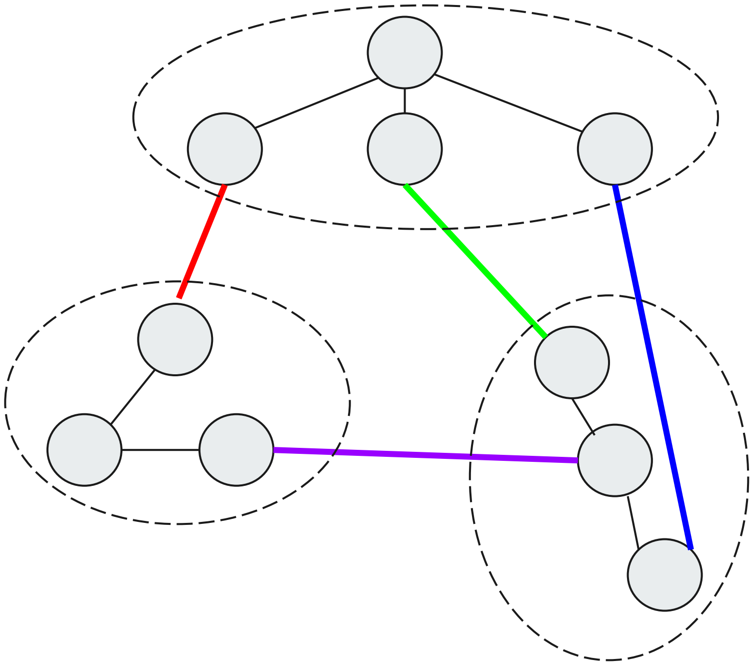
Contraction



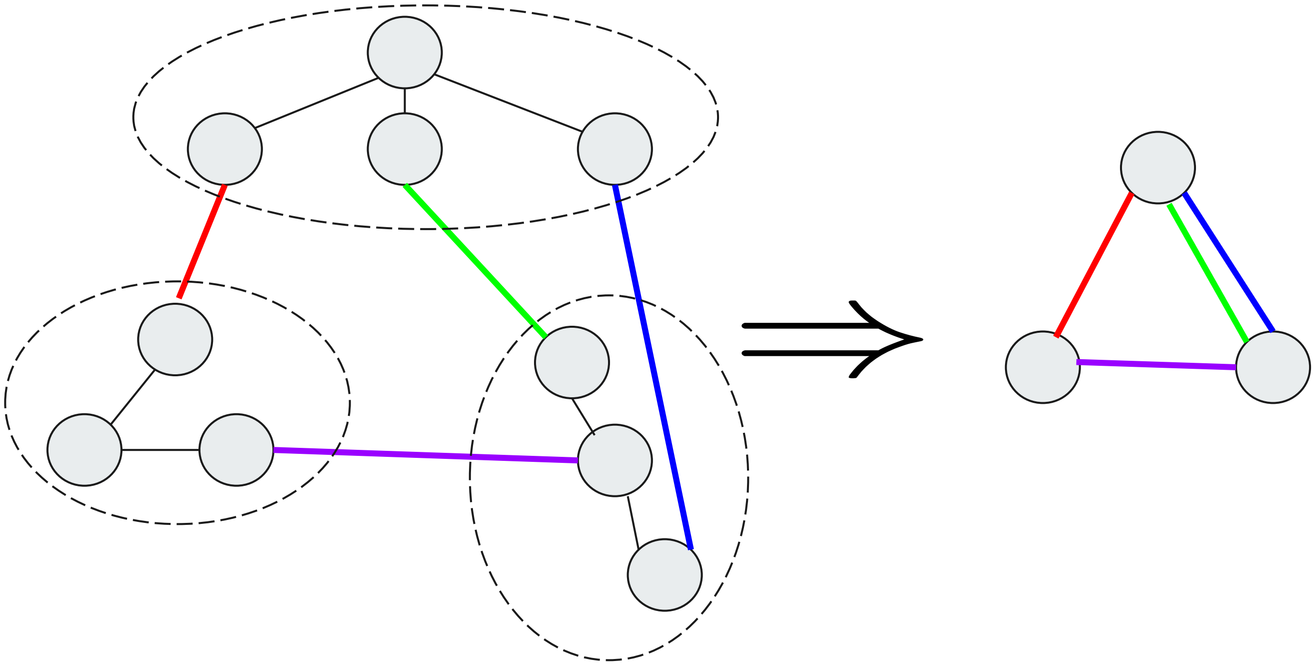
Contraction



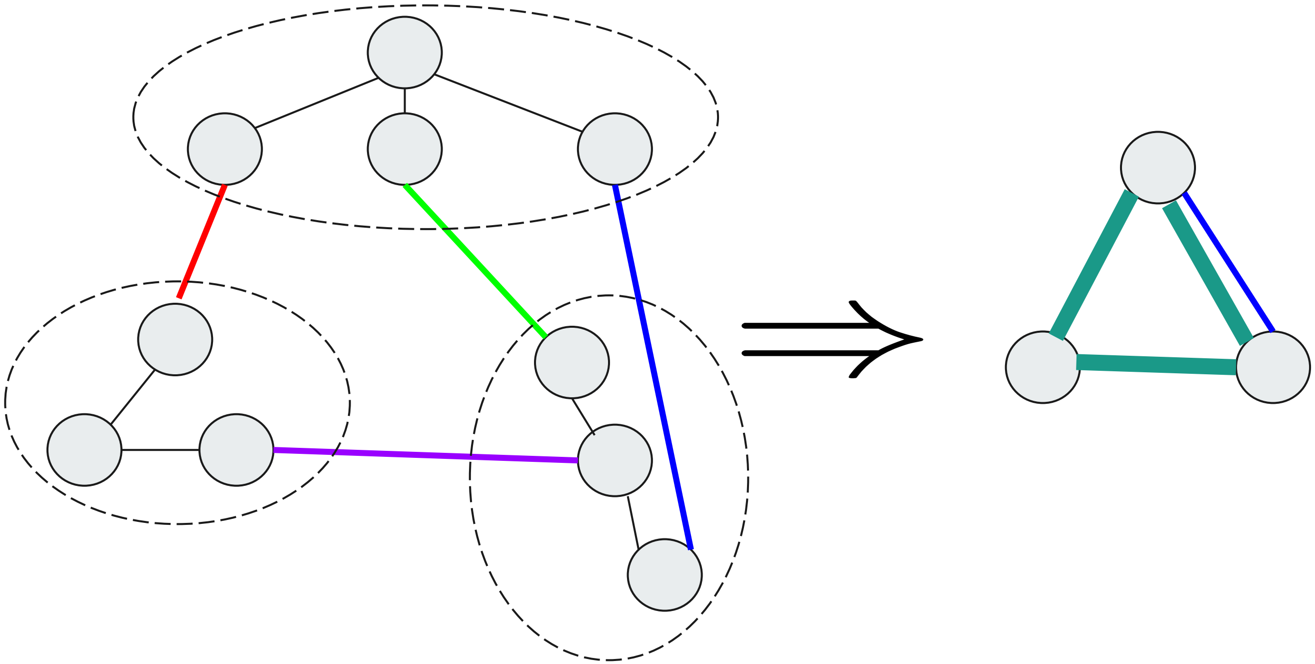
Contraction



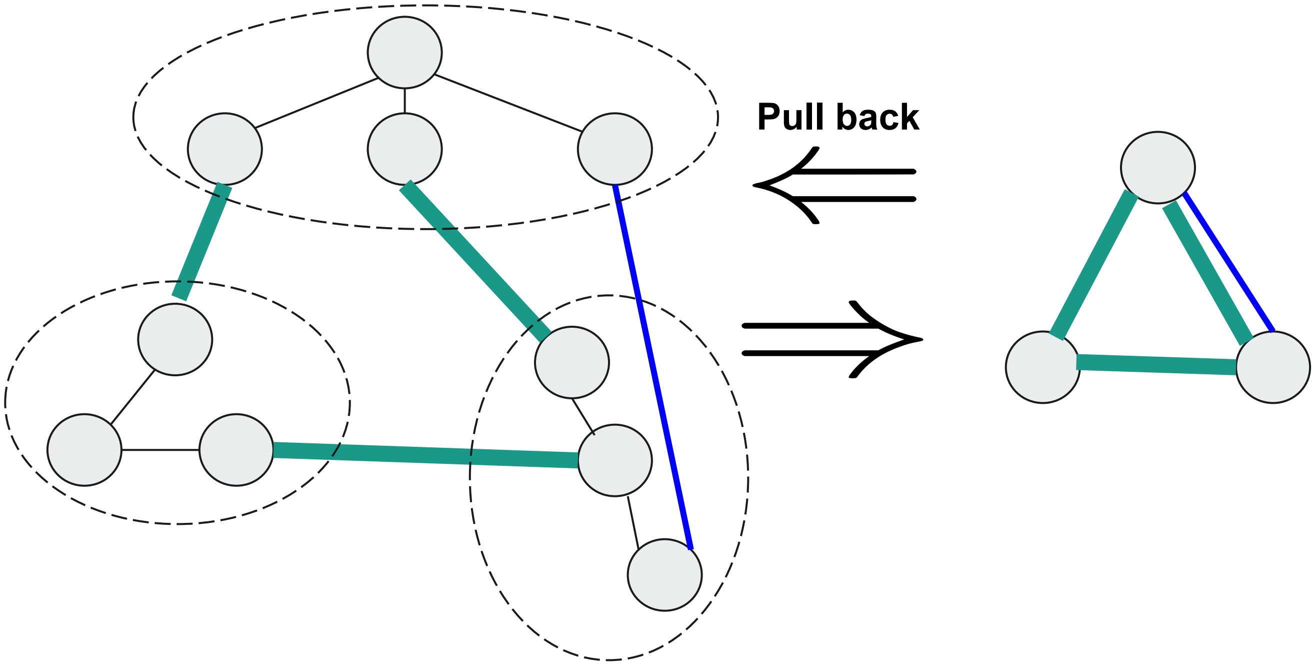
Contraction



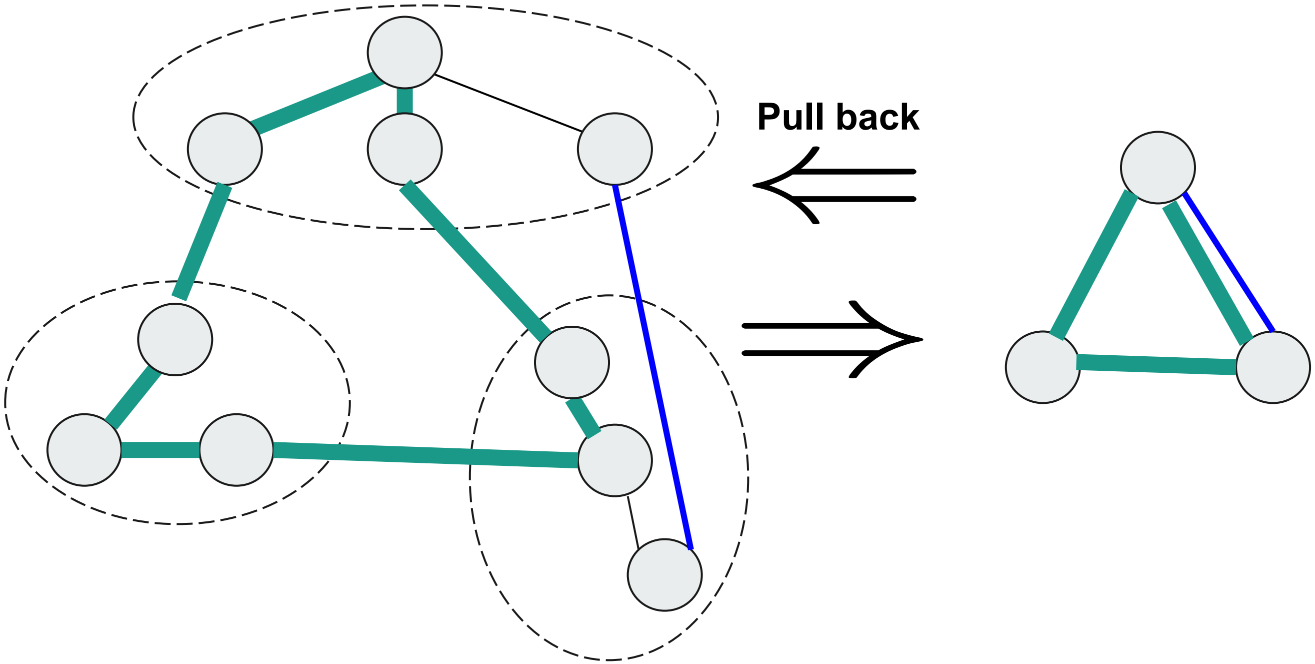
Contraction



Contraction



Contraction



Algorithm description

Algorithm description

Fix a constant $k = n^{\frac{1}{c+1}}$

Algorithm description

Fix a constant $k = n^{\frac{1}{c+1}}$

Partition G into components G_1, G_2, …, G_t, each with diameter $O(\log n)$ and around k vertices

Algorithm description

Fix a constant $k = n^{\frac{1}{c+1}}$

Partition G into components $G_1, G_2, \ldots, G_t$, each with diameter $O(\log n)$ and around k vertices

Contract the components, and recurse on the new graph H (which has $\frac{n}{k}$ vertices)

Algorithm description

Fix a constant $k = n^{\frac{1}{c+1}}$

Partition G into components $G_1, G_2, \ldots, G_t$, each with diameter $O(\log n)$ and around k vertices

Contract the components, and recurse on the new graph H (which has $\frac{n}{k}$ vertices)

Pull cycles in H up to G

Partitioning the vertices

Partitioning the vertices

Partition G into components G_1, G_2, ..., G_t, each with diameter $O(\log n)$ and around k vertices

Partitioning the vertices

Partition G into components G_1, G_2, ..., G_t, each with diameter $O(\log n)$ and around k vertices

Algorithm

Partitioning the vertices

Partition G into components G_1, G_2, ..., G_t, each with diameter $O(\log n)$ and around k vertices

Algorithm

Do a low diameter decomposition; components might have > k vertices

Partitioning the vertices

Partition G into components G_1, G_2, ..., G_t, each with diameter $O(\log n)$ and around k vertices

Algorithm

Do a low diameter decomposition; components might have > k vertices

Build a low diameter spanning tree on each component

Partitioning the vertices

Partition G into components G_1, G_2, ..., G_t, each with diameter $O(\log n)$ and around k vertices

Algorithm

Do a low diameter decomposition; components might have > k vertices

Build a low diameter spanning tree on each component

Partition the spanning tree into components of size approximately k

Partitioning the vertices

Partition G into components G_1, G_2, ..., G_t, each with diameter $O(\log n)$ and around k vertices

Algorithm

Do a low diameter decomposition; components might have > k vertices

Build a low diameter spanning tree on each component

Partition the spanning tree into components of size approximately k

Works because graph has bounded degree

Recursion details

Recursion details

Contract the components, and recurse on the new graph H (which has $\frac{n}{k}$ vertices)

Recursion details

Contract the components, and recurse on the new graph H (which has $\frac{n}{k}$ vertices)

Issue: H still has m edges

Recursion details

Contract the components, and recurse on the new graph H (which has $\frac{n}{k}$ vertices)

Issue: H still has m edges

Fix: Recall that we are looking for cycles containing $\frac{m}{10\Delta}$ vertices

Recursion details

Contract the components, and recurse on the new graph H (which has $\frac{n}{k}$ vertices)

Issue: H still has m edges

Fix: Recall that we are looking for cycles containing $\frac{m}{10\Delta}$ vertices

Sparsify H, proportionally reducing m and Δ

Recursion details

Contract the components, and recurse on the new graph H (which has $\frac{n}{k}$ vertices)

Issue: H still has m edges

Fix: Recall that we are looking for cycles containing $\frac{m}{10\Delta}$ vertices

Sparsify H, proportionally reducing m and Δ

This way, the maximum degree of H isn’t much larger than that of G

Conclusion

Conclusion

We give new algorithms for short cycle decomposition

Conclusion

We give new algorithms for short cycle decomposition

Improves over previous work ([CGP+18]) in terms of runtime, number of remaining edges, and cycle length

Conclusion

We give new algorithms for short cycle decomposition

Improves over previous work ([CGP+18]) in terms of runtime, number of remaining edges, and cycle length

Improvements to all of:

Graphical spectral sketches and Resistance Sparsifiers

Estimating effective resistances

Degree-preserving sparsifiers and Eulerian directed graph sparsifiers

Recent work

Recent work

Parter and Yogev have an upcoming result which gets $(O(n \log n), O(\log^2 n))$ short cycle decomposition in time $n^{1+o(1)}$.

Thanks for listening!