Distributed Laplacian Solving with Applications

Sebastian Forster, né Krinninger

University of Salzburg

SIROCCO 2022

Joint work with Gramoz Goranci, Yang P. Liu, Richard Peng, Xiaorui Sun, Tijn de Vos, and Mingquan Ye

PARIS LODRON UNIVERSITÄT SALZBURG

European Research Council
Established by the European Commission

Der Wissenschaftsfonds.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 947702). Supported by the Austrian Science Fund (FWF): P 32863-N

Laplacian Paradigm

- Laplacian systems
- Spectral sparsifiers
- Electrical flow
- Effective resistance
- Expander decompositions
- Continuous optimization
- Interior-point methods
- Gradient descent
- Preconditioning
- ...

Laplacian Paradigm and Distributed Computing

Observation

Laplacian paradigm often yields inherently parallelizable algorithms

Laplacian Paradigm and Distributed Computing

Observation

Laplacian paradigm often yields inherently parallelizable algorithms

Basic operation:

- Vector $\mathbf{x}$: each node represents a coordinate
- Matrix $\mathbf{A}$: each edge represents a non-zero entry
- Matrix-vector multiplication $\mathbf{Ax}$: one round

Laplacian Paradigm and Distributed Computing

Observation

Laplacian paradigm often yields inherently parallelizable algorithms

Basic operation:

- Vector $\mathbf{x}$: each node represents a coordinate
- Matrix $\mathbf{A}$: each edge represents a non-zero entry
- Matrix-vector multiplication $\mathbf{Ax}$: one round

State of the art for (approximate) single-source shortest path, maximum flow, minimum-cost flow:
[Ghaffari, Karrenbauer, Kuhn, Lenzen, Patt-Shamir '15] [Becker, F, Karrenbauer, Lenzen '17] [Zuzic '21] [Anagnostides, Themis Gouleakis, Christoph Lenzen '21] [Zuzic, Goranci, Ye, Haeupler, Sun '22] [Rozhon, Grunau, Haeupler, Zuzic, Li '22]

Laplacian Systems

Goal

Solve linear system $\mathbf{L}\mathbf{x} = \mathbf{b}$ such that $\mathbf{L}$ is a **Laplacian matrix**.

Laplacian Systems

Goal

Solve linear system $\mathbf{L}\mathbf{x} = \mathbf{b}$ such that $\mathbf{L}$ is a **Laplacian matrix**.

Definition

The **Laplacian matrix** $\mathbf{L}(G)$ of graph $G = (V, E, w)$ is defined by

$$\mathbf{L}(G)_{u,v} = \begin{cases} \sum_{(u,v')\in E} w_{u,v'} & \text{if } u = v, \\ -w_{u,v} & \text{otherwise.} \end{cases}$$

Laplacian Systems

Goal

Solve linear system $\mathbf{Lx} = \mathbf{b}$ such that $\mathbf{L}$ is a **Laplacian matrix**.

Definition

The **Laplacian matrix** $\mathbf{L}(G)$ of graph $G = (V, E, w)$ is defined by

$$\mathbf{L}(G)_{u,v} = \begin{cases} \sum_{(u,v')\in E} w_{u,v'} & \text{if } u = v, \\ -w_{u,v} & \text{otherwise.} \end{cases}$$

High-precision solver: Approximation of solution $\mathbf{x}^*$ with $\mathbf{x}$ s.t.

$$\|\mathbf{x} - \mathbf{x}^*\|_{\mathbf{L}(G)} \leq \epsilon \|\mathbf{b}\|_{\mathbf{L}(G)} .$$

Laplacian Systems

Goal

Solve linear system $\mathbf{L}\mathbf{x} = \mathbf{b}$ such that $\mathbf{L}$ is a **Laplacian matrix**.

Definition

The **Laplacian matrix** $\mathbf{L}(G)$ of graph $G = (V, E, w)$ is defined by

$$\mathbf{L}(G)_{u,v} = \begin{cases} \sum_{(u,v')\in E} w_{u,v'} & \text{if } u = v, \\ -w_{u,v} & \text{otherwise.} \end{cases}$$

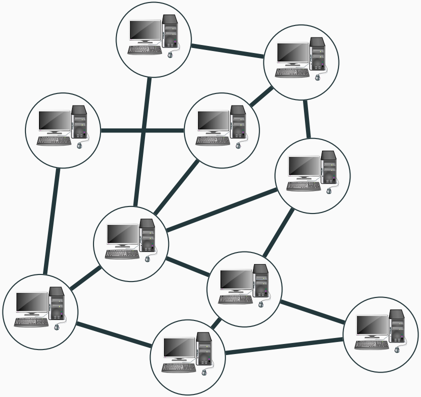
High-precision solver: Approximation of solution $\mathbf{x}^*$ with $\mathbf{x}$ s.t.

$$\|\mathbf{x} - \mathbf{x}^*\|_{\mathbf{L}(G)} \leq \epsilon \|\mathbf{b}\|_{\mathbf{L}(G)}.$$

Prior work:

- $\tilde{O}(m)$ sequential running time [Spielman, Teng '04]
- $\tilde{O}(m)$ work, polylogarithmic depth [Peng, Spielman '14]

CONGEST Model



- Edges correspond to non-zero entries of matrix
- Each node holds one row/column of matrix
- Communication over edges in synchronous rounds
- Bandwidth $O(\log n)$ per edge

Our Results for the CONGEST Model

Theorem ([F, Goranci, Liu, Peng, Sun, Ye])

In the CONGEST model, given a weighted and undirected graph G and a vector $\mathbf{b}$ on n vertices, we can in $O(n^{o(1)}(\sqrt{n}+D))$ rounds return a vector $\mathbf{x}$ such that $\|\mathbf{x}-\mathbf{x}^*\|_{\mathbf{L}(G)} \leq \epsilon\|\mathbf{b}\|_{\mathbf{L}(G)}$.

Almost matches a $\tilde{\Omega}(\sqrt{n}+D)$ lower bound

Our Results for the CONGEST Model

Theorem ([F, Goranci, Liu, Peng, Sun, Ye])

In the CONGEST model, given a weighted and undirected graph G and a vector $\mathbf{b}$ on n vertices, we can in $O(n^{o(1)}(\sqrt{n}+D))$ rounds return a vector $\mathbf{x}$ such that $\|\mathbf{x}-\mathbf{x}^*\|_{\mathbf{L}(G)} \leq \epsilon \|\mathbf{b}\|_{\mathbf{L}(G)}$.

Almost matches a $\tilde{\Omega}(\sqrt{n}+D)$ lower bound

Implications

$\tilde{O}\left(m^{3/7+o(1)}(n^{1/2}D^{1/4}+D)\right)$-round algorithms in CONGEST model for the following problems:

- Maximum flow [Mądry '16]
- Unit capacity minimum cost flow [Cohen et al. '17]
- Negative weight shortest path [Cohen et al. '17]

First $o(n)$-round algorithms for sparse, low-diameter graphs

Approximate Schur Complement

Definition (Schur complement)

For an $n \times n$ symmetric matrix $\mathbf{M}$ and a subset of *terminals* $T \subseteq [n]$, let $S = [n] \setminus T$. Permute the rows/columns of $\mathbf{M}$ to write

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{[S,S]} & \mathbf{M}_{[S,T]} \\ \mathbf{M}_{[T,S]} & \mathbf{M}_{[T,T]} \end{bmatrix}.$$

Then the *Schur complement* of $\mathbf{M}$ onto T is defined as

$\mathrm{SC}(\mathbf{M}, T) := \mathbf{M}_{[T,T]} - \mathbf{M}_{[T,S]}\mathbf{M}_{[S,S]}^{-1}\mathbf{M}_{[S,T]}$.

Result of block Gaussian elimination

Approximate Schur Complement

Definition (Schur complement)

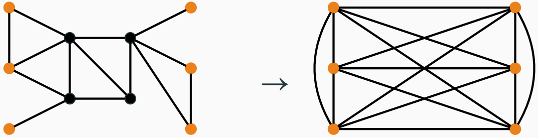
For an $n \times n$ symmetric matrix $\mathbf{M}$ and a subset of *terminals* $T \subseteq [n]$, let $S = [n] \setminus T$. Permute the rows/columns of $\mathbf{M}$ to write

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{[S,S]} & \mathbf{M}_{[S,T]} \\ \mathbf{M}_{[T,S]} & \mathbf{M}_{[T,T]} \end{bmatrix}.$$

Then the *Schur complement* of $\mathbf{M}$ onto T is defined as
$\mathrm{SC}(\mathbf{M}, T) := \mathbf{M}_{[T,T]} - \mathbf{M}_{[T,S]}\mathbf{M}_{[S,S]}^{-1}\mathbf{M}_{[S,T]}$.

Result of block Gaussian elimination

Graphical interpretation:



Input graph $\qquad$ Schur complement

Approximate Schur Complement

Definition (Schur complement)

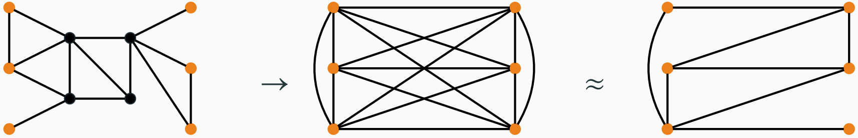
For an $n \times n$ symmetric matrix $\mathbf{M}$ and a subset of *terminals* $T \subseteq [n]$, let $S = [n] \setminus T$. Permute the rows/columns of $\mathbf{M}$ to write

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{[S,S]} & \mathbf{M}_{[S,T]} \\ \mathbf{M}_{[T,S]} & \mathbf{M}_{[T,T]} \end{bmatrix}.$$

Then the *Schur complement* of $\mathbf{M}$ onto T is defined as $\mathrm{SC}(\mathbf{M}, T) := \mathbf{M}_{[T,T]} - \mathbf{M}_{[T,S]}\mathbf{M}_{[S,S]}^{-1}\mathbf{M}_{[S,T]}$.

Result of block Gaussian elimination

Graphical interpretation:



Input graph Schur complement Sparsification

Minor Sparsifiers

Problem

Communication of edges "along" sparsifier edges my lead to too much congestion

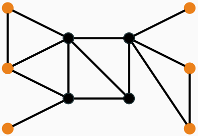
Minor Sparsifiers

Problem

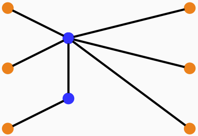
Communication of edges “along” sparsifier edges my lead to too much congestion

Solution

Vertex sparsifiers as **minors** of the communication graph



→



Minor Sparsifiers

Problem

Communication of edges “along” sparsifier edges my lead to too much congestion

Solution

Vertex sparsifiers as **minors** of the communication graph

Lemma

Matrix-vector multiplication involving minor sparsifier takes $\tilde{O}(\sqrt{n} + D)$ rounds.

Minor Sparsifiers

Problem

Communication of edges “along” sparsifier edges my lead to too much congestion

Solution

Vertex sparsifiers as **minors** of the communication graph

Lemma

Matrix-vector multiplication involving minor sparsifier takes $\tilde{O}(\sqrt{n} + D)$ rounds.

Key contribution: Parallel variant of [Li Schild ’18]

Technical Details

- From [Kyng, Lee, Peng, Sachdeva, Spielman '16]: Repeated elimination of **almost independent sets** yields vertex sparsifier "chain" with recursion depth $d = O(\log n)$

Technical Details

- From [Kyng, Lee, Peng, Sachdeva, Spielman '16]: Repeated elimination of **almost independent sets** yields vertex sparsifier "chain" with recursion depth $d = O(\log n)$
  - Fast computation of inverse of submatrix of eliminated nodes using iterative method

Technical Details

- From [Kyng, Lee, Peng, Sachdeva, Spielman '16]: Repeated elimination of **almost independent sets** yields vertex sparsifier "chain" with recursion depth $d = O(\log n)$
  - Fast computation of inverse of submatrix of eliminated nodes using iterative method
- In addition to Schur complement itself, we need to compute further information (linear operators)

Technical Details

- From [Kyng, Lee, Peng, Sachdeva, Spielman '16]: Repeated elimination of **almost independent sets** yields vertex sparsifier "chain" with recursion depth $d = O(\log n)$
  - Fast computation of inverse of submatrix of eliminated nodes using iterative method
- In addition to Schur complement itself, we need to compute further information (linear operators)
  - Obtain Schur complement from sampling random walks

Technical Details

- From [Kyng, Lee, Peng, Sachdeva, Spielman ’16]: Repeated elimination of **almost independent sets** yields vertex sparsifier “chain” with recursion depth $d = O(\log n)$
  - Fast computation of inverse of submatrix of eliminated nodes using iterative method
- In addition to Schur complement itself, we need to compute further information (linear operators)
  - Obtain Schur complement from sampling random walks
  - Algorithmically: estimate of *congestion* in random walks

Technical Details

- From [Kyng, Lee, Peng, Sachdeva, Spielman '16]: Repeated elimination of **almost independent sets** yields vertex sparsifier "chain" with recursion depth $d = O(\log n)$
  - Fast computation of inverse of submatrix of eliminated nodes using iterative method
- In addition to Schur complement itself, we need to compute further information (linear operators)
  - Obtain Schur complement from sampling random walks
  - Algorithmically: estimate of *congestion* in random walks
  - Computation introduces round-overhead of $(\log^c n)^d$; can only work with $d = O(\log \log n) \rightarrow$ sophisticated recursion

Technical Details

- From [Kyng, Lee, Peng, Sachdeva, Spielman ’16]: Repeated elimination of **almost independent sets** yields vertex sparsifier “chain” with recursion depth $d = O(\log n)$
    - Fast computation of inverse of submatrix of eliminated nodes using iterative method
- In addition to Schur complement itself, we need to compute further information (linear operators)
    - Obtain Schur complement from sampling random walks
    - Algorithmically: estimate of *congestion* in random walks
    - Computation introduces round-overhead of $(\log^c n)^d$; can only work with $d = O(\log \log n) \rightarrow$ sophisticated recursion
- Minor sparsifiers: avoid sequential sampling of [Li Schild ’18]

Technical Details

- From [Kyng, Lee, Peng, Sachdeva, Spielman '16]: Repeated elimination of **almost independent sets** yields vertex sparsifier "chain" with recursion depth $d = O(\log n)$
  - Fast computation of inverse of submatrix of eliminated nodes using iterative method
- In addition to Schur complement itself, we need to compute further information (linear operators)
  - Obtain Schur complement from sampling random walks
  - Algorithmically: estimate of *congestion* in random walks
  - Computation introduces round-overhead of $(\log^c n)^d$; can only work with $d = O(\log \log n) \rightarrow$ sophisticated recursion
- Minor sparsifiers: avoid sequential sampling of [Li Schild '18]
  - Identify "steady" edges that can be sampled independently
  - Requires recursive solution of linear system: edge reduction via ultra-sparsifiers

Technical Details

- From [Kyng, Lee, Peng, Sachdeva, Spielman '16]: Repeated elimination of **almost independent sets** yields vertex sparsifier "chain" with recursion depth $d = O(\log n)$
  - Fast computation of inverse of submatrix of eliminated nodes using iterative method
- In addition to Schur complement itself, we need to compute further information (linear operators)
  - Obtain Schur complement from sampling random walks
  - Algorithmically: estimate of *congestion* in random walks
  - Computation introduces round-overhead of $(\log^c n)^d$; can only work with $d = O(\log \log n) \rightarrow$ sophisticated recursion
- Minor sparsifiers: avoid sequential sampling of [Li Schild '18]
  - Identify "steady" edges that can be sampled independently
  - Requires recursive solution of linear system: edge reduction via ultra-sparsifiers
  - Distortion of minor property in recursive calls

What Next?

Implications

$\tilde{O}\left(m^{3/7+o(1)}(n^{1/2}D^{1/4} + D)\right)$-round algorithms in CONGEST model for the following problems:

- Maximum flow [Mądry '16]
- Unit capacity minimum cost flow [Cohen et al. '17]
- Negative weight shortest path: [Cohen et al. '17]

What Next?

Implications

$\tilde{O}\left(m^{3/7+o(1)}(n^{1/2}D^{1/4}+D)\right)$-round algorithms in CONGEST model for the following problems:

- Maximum flow [Mądry '16]
- Unit capacity minimum cost flow [Cohen et al. '17]
- Negative weight shortest path: [Cohen et al. '17]

Question

Sublinear #rounds in dense graphs?

What Next?

Implications

$\tilde{O}\left(m^{3/7+o(1)}(n^{1/2}D^{1/4} + D)\right)$-round algorithms in CONGEST model for the following problems:

- Maximum flow [Mądry '16]
- Unit capacity minimum cost flow [Cohen et al. '17]
- Negative weight shortest path: [Cohen et al. '17]

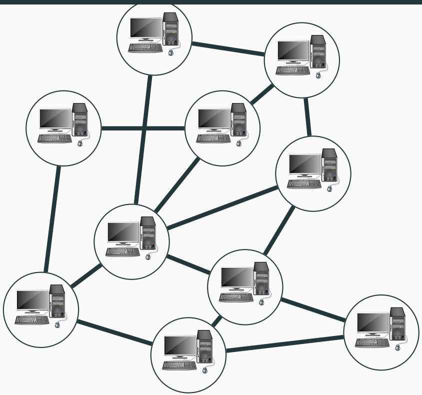
Question

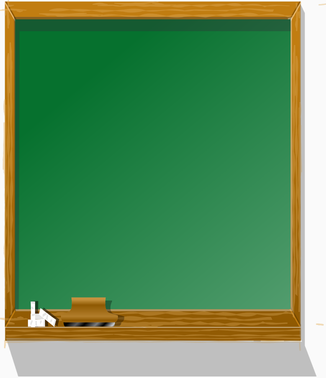
Sublinear #rounds in dense graphs?

Easier Question

Sublinear #rounds on the Broadcast Congested Clique?

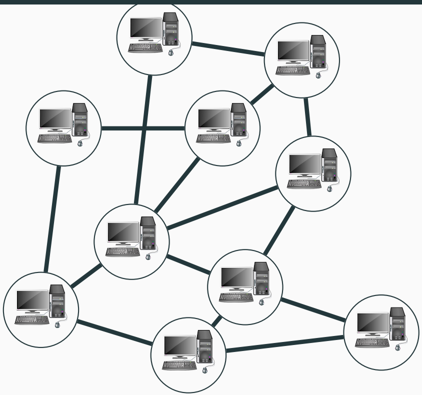
Broadcast Congested Clique

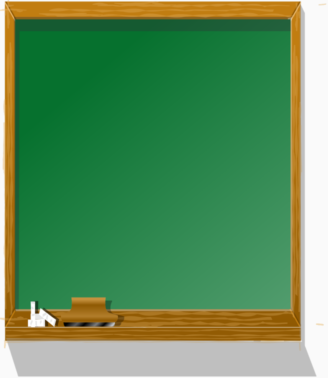




- Nodes can communicate with all other nodes [Lotker et al. ’05]
- Broadcast *the same* message to all nodes [Drucker, Kuhn, Oshman ’12]

Broadcast Congested Clique





- Nodes can communicate with all other nodes [Lotker et al. ’05]
- Broadcast *the same* message to all nodes [Drucker, Kuhn, Oshman ’12]
- For many problems: only “trivialization” of CONGEST model upper bounds with $D = 1$ is known

Our Results for the BCC

Theorem ([F, de Vos '22])

On the Broadcast Congested Clique, the minimum cost flow problem can be solved in $\tilde{O}(\sqrt{n})$ rounds.

Our Results for the BCC

Theorem ([F, de Vos '22])

On the Broadcast Congested Clique, the minimum cost flow problem can be solved in $\tilde{O}(\sqrt{n})$ rounds.

Other Results:

- On the Broadcast Congested Clique, a spectral sparsifier of quality $1 \pm \epsilon$ and size $\tilde{O}(n/\epsilon^2)$ can be computed in $\tilde{O}(1/\epsilon^2)$ rounds
- On the Broadcast Congested Clique, a Laplacian system can be solved up to high accuracy in $\tilde{O}(\log^2(1/\epsilon))$ rounds
- On the Broadcast Congested Clique, certain Linear Programs can be solved in $\tilde{O}(\sqrt{n})$ rounds.

Main Idea and Challenges

Linear Programming

Minimize $\mathbf{c}^\top \mathbf{x}$ subject to $\mathbf{A}\mathbf{x} = \mathbf{b}$

Main Idea and Challenges

Linear Programming

Minimize $\mathbf{c}^\top \mathbf{x}$ subject to $\mathbf{A}\mathbf{x} = \mathbf{b}$

Implementation of [Lee, Sidford '14]:

- Interior point method with $\tilde{O}(\sqrt{\text{rank}})$ iterations
- One linear system solve per iteration

Main Idea and Challenges

Linear Programming

Minimize $\mathbf{c}^\top \mathbf{x}$ subject to $\mathbf{A}\mathbf{x} = \mathbf{b}$

Implementation of [Lee, Sidford ’14]:

- Interior point method with $\tilde{O}(\sqrt{\text{rank}})$ iterations
- One linear system solve per iteration

Minimum cost flow:

- Rank = #nodes
- Linear system has Laplacian matrix

Key Contribution

Iterative computation of spectral sparsifier [Koutis, Xu '16]:

- Compute a spanner
- Sample non-spanner edges with constant probability

Key Contribution

Iterative computation of spectral sparsifier [Koutis, Xu ’16]:

- Compute a spanner
- Sample non-spanner edges with constant probability

Problem

On Broadcast Congested Clique, nodes cannot easily coordinate with neighbors on sampling incident edges

Key Contribution

Iterative computation of spectral sparsifier [Koutis, Xu '16]:

- Compute a spanner
- Sample non-spanner edges with constant probability

Problem

On Broadcast Congested Clique, nodes cannot easily coordinate with neighbors on sampling incident edges

Solution:

- Compute spanner on "probabilistic" graph
- Sample individual edges ad-hoc when needed
- Modification of spanner algorithm of [Baswana, Sen '07]

Optimization vs. Data Structures

[Lee, Sidford '14]:
#iterations: $\tilde{O}(\sqrt{n})$
Time per iteration: $\tilde{O}(m)$

[Chen et al. '22]:
#iterations: $m^{1+o(1)}$
Time per iteration: $m^{o(1)}$

Optimization vs. Data Structures

[Lee, Sidford '14]:
#iterations: $\tilde{O}(\sqrt{n})$
Time per iteration: $\tilde{O}(m)$

Iteration count carries over to round complexity

[Chen et al. '22]:
#iterations: $m^{1+o(1)}$
Time per iteration: $m^{o(1)}$

Running time improvement does not improve round complexity

Optimization vs. Data Structures

[Lee, Sidford '14]:
#iterations: $\tilde{O}(\sqrt{n})$
Time per iteration: $\tilde{O}(m)$

Iteration count carries over to round complexity

[Chen et al. '22]:
#iterations: $m^{1+o(1)}$
Time per iteration: $m^{o(1)}$

Running time improvement does not improve round complexity

Question

Is $\tilde{\Theta}(\sqrt{n})$ the right iteration count for min-cost flow LP?

Open Problem

Question

Is $\tilde{\Theta}(\sqrt{n})$ the right round complexity for min-cost flow in the BCC?

Open Problem

Question

Is $\tilde{\Theta}(\sqrt{n})$ the right round complexity for min-cost flow in the BCC?

- Lower bounds in BCC at least not hopeless [Frischknecht, Holzer, Wattenhofer '12] [Drucker, Kuhn, Oshman '14] [Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, Suomela] [Holzer, Pinsker '15] [Becker, Montealegre, Rapaport, Todinca '18]

Open Problem

Question

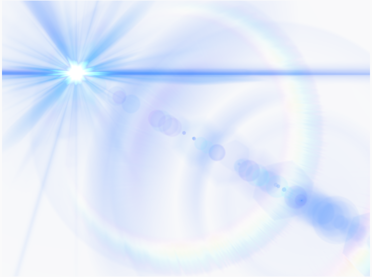
Is $\tilde{\Theta}(\sqrt{n})$ the right round complexity for min-cost flow in the BCC?

- Lower bounds in BCC at least not hopeless [Frischknecht, Holzer, Wattenhofer '12] [Drucker, Kuhn, Oshman '14] [Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, Suomela] [Holzer, Pinsker '15] [Becker, Montealegre, Rapaport, Todinca '18]
- Better upper bound already interesting for single-source reachability

Conclusion

Almost optimal Laplacian solvers

Conclusion



Almost optimal Laplacian solvers

Broadcast Congested Clique is an interesting "burning glass"