Hardness results and an exact exponential algorithm for the spanning tree congestion problem

Yoshio Okamoto[1] Yota Otachi[2] Ryuhei Uehara[1]
Takeaki Uno[3]

[1]JAIST

[2]Tohoku Univ.

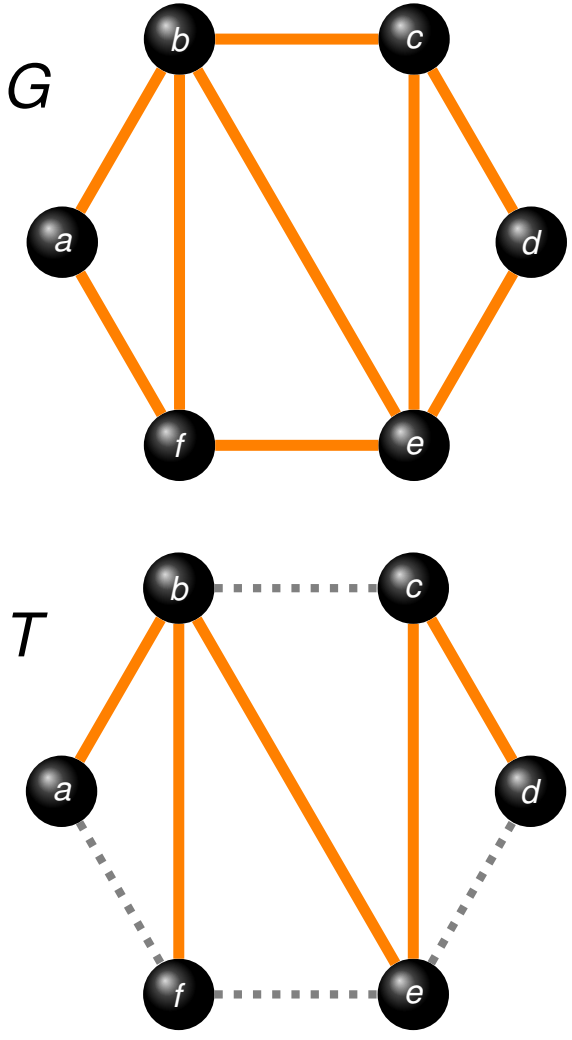
[3]NII

GROW 2011, October 29. KAIST, Korea.
(This work was presented also at TAMC 2011.)

Outline

Outline

Spanning Tree Congestion



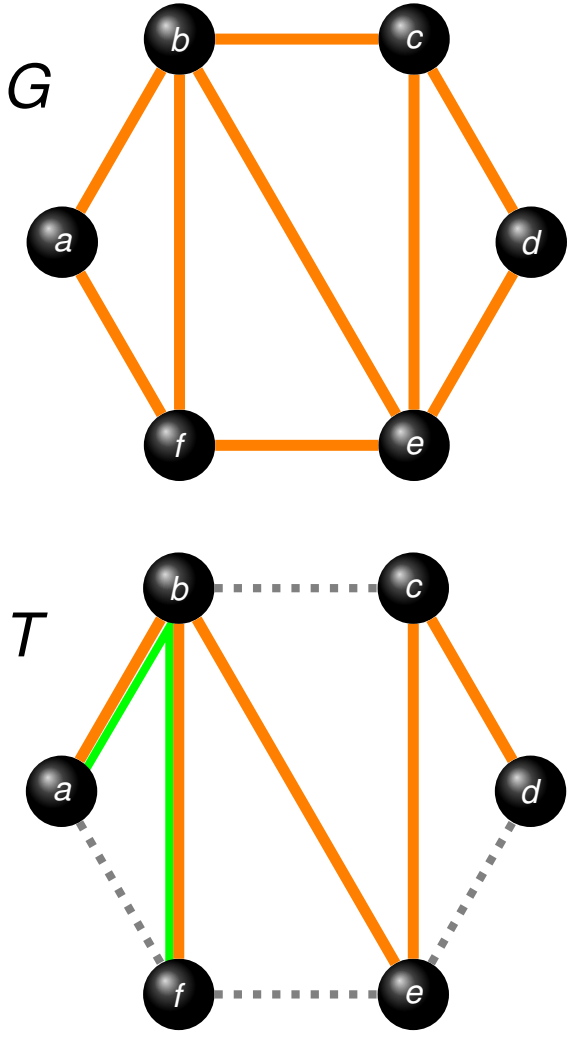
We consider connected graphs only.

Definition (Spanning Tree Congestion)

T: a spanning tree of a graph G.

- The *detour* of $\{u, v\} \in E(G)$ is the unique u–v path in T.
- The *congestion of* $e \in E(T)$, $cng_{G,T}(e)$ is the number of edges in G whose detours contain e.
- The *congestion of* T, $cng_G(T)$ is the max congestion over all its edges.
- The *spanning tree congestion* of G, $stc(G)$ is the min congestion over all its spanning trees.

Spanning Tree Congestion



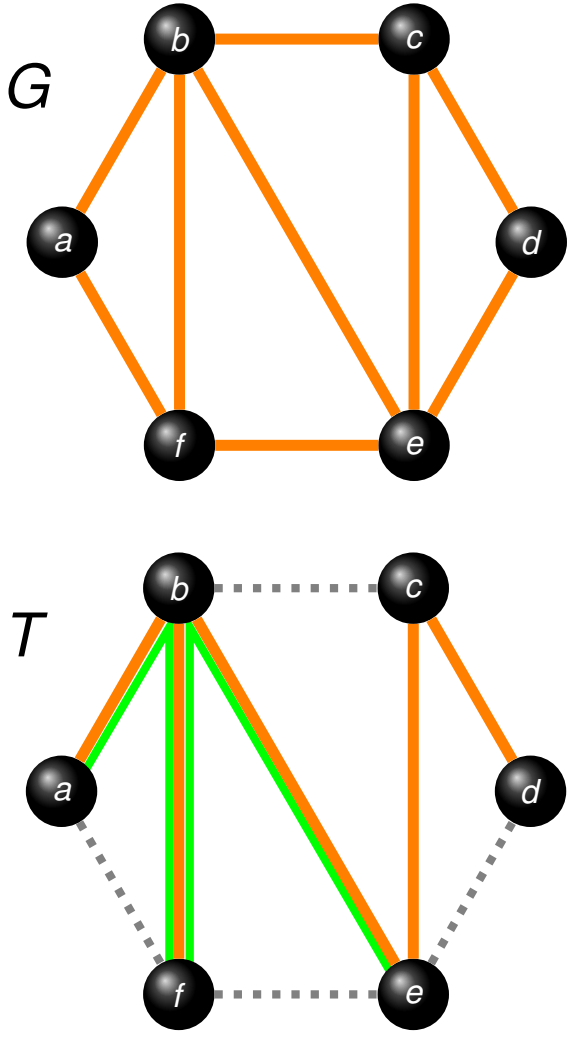
We consider connected graphs only.

Definition (Spanning Tree Congestion)

T: a spanning tree of a graph G.

- The *detour* of $\{u, v\} \in E(G)$ is the unique u–v path in T.
- The *congestion of* $e \in E(T)$, $cng_{G,T}(e)$ is the number of edges in G whose detours contain e.
- The *congestion of* T, $cng_G(T)$ is the max congestion over all its edges.
- The *spanning tree congestion* of G, $stc(G)$ is the min congestion over all its spanning trees.

Spanning Tree Congestion



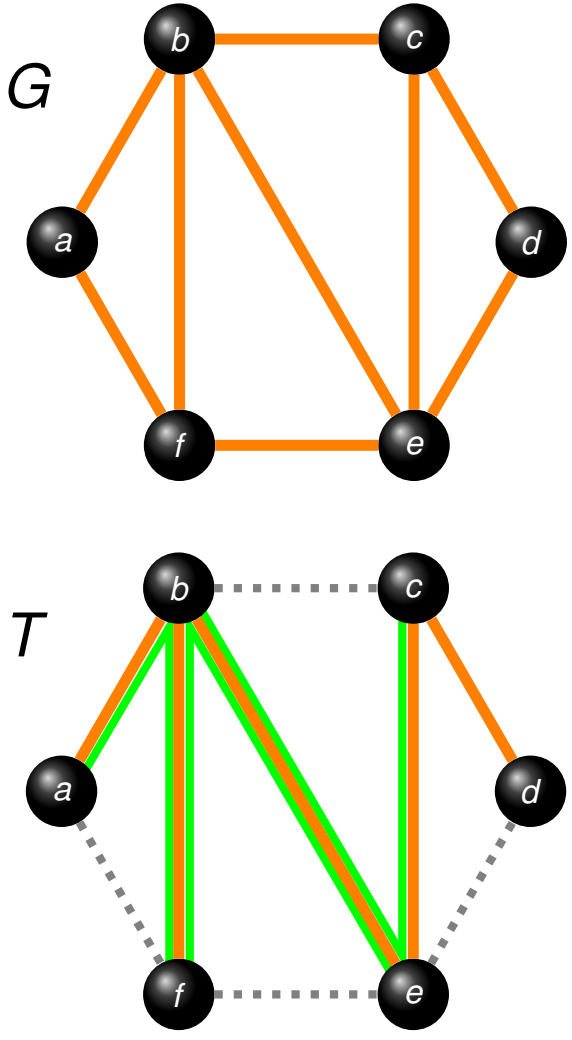
We consider connected graphs only.

Definition (Spanning Tree Congestion)

T: a spanning tree of a graph G.

- The *detour* of $\{u, v\} \in E(G)$ is the unique u–v path in T.
- The *congestion of* $e \in E(T)$, $cng_{G,T}(e)$ is the number of edges in G whose detours contain e.
- The *congestion of* T, $cng_G(T)$ is the max congestion over all its edges.
- The *spanning tree congestion* of G, $stc(G)$ is the min congestion over all its spanning trees.

Spanning Tree Congestion



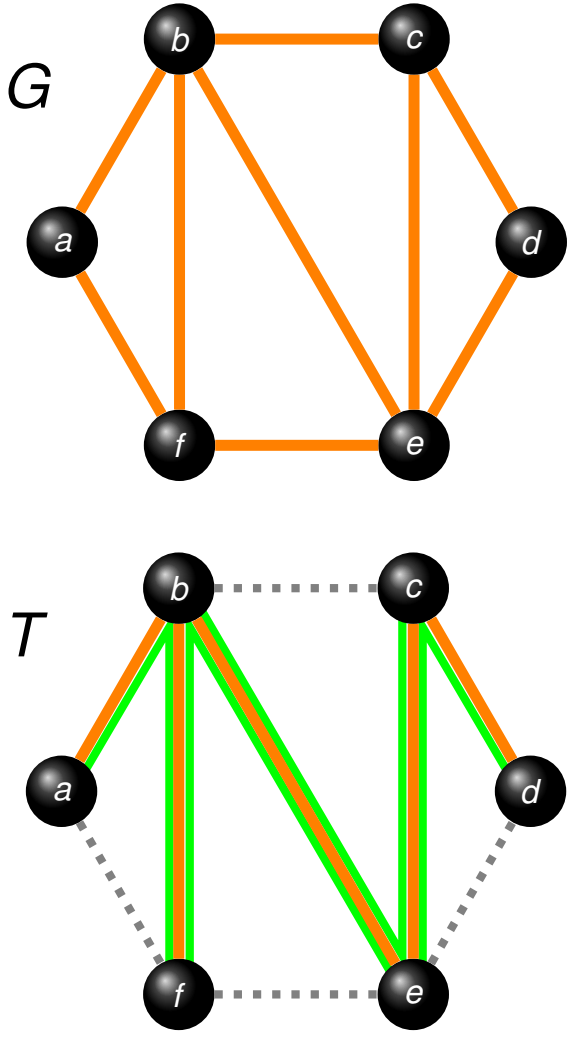
We consider connected graphs only.

Definition (Spanning Tree Congestion)

T: a spanning tree of a graph G.

- The *detour* of $\{u, v\} \in E(G)$ is the unique u–v path in T.
- The *congestion of* $e \in E(T)$, $cng_{G,T}(e)$ is the number of edges in G whose detours contain e.
- The *congestion of* T, $cng_G(T)$ is the max congestion over all its edges.
- The *spanning tree congestion* of G, $stc(G)$ is the min congestion over all its spanning trees.

Spanning Tree Congestion



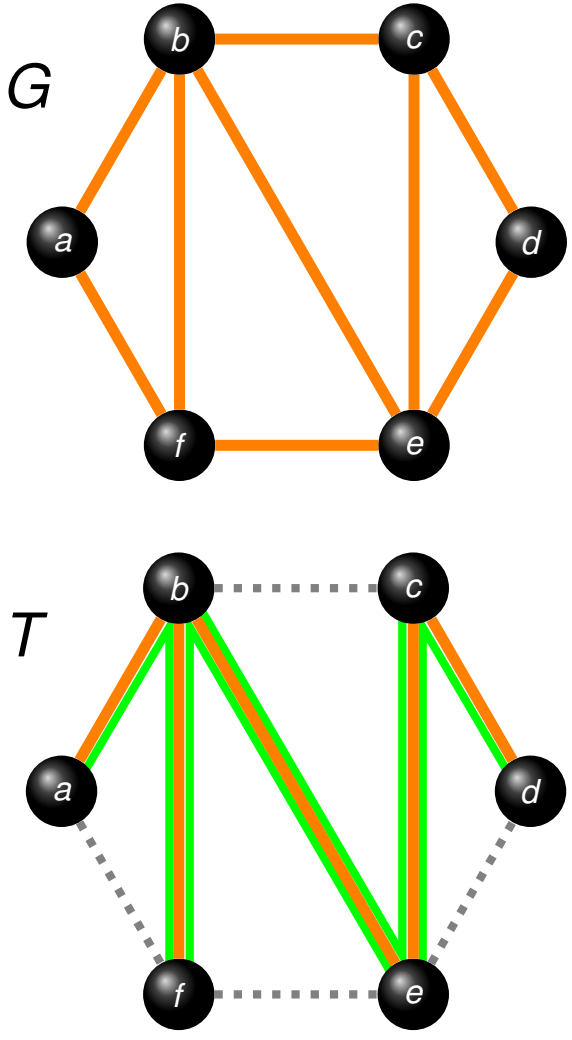
We consider connected graphs only.

Definition (Spanning Tree Congestion)

T: a spanning tree of a graph G.

- The *detour* of $\{u, v\} \in E(G)$ is the unique u–v path in T.
- The *congestion of* $e \in E(T)$, $cng_{G,T}(e)$ is the number of edges in G whose detours contain e.
- The *congestion of* T, $cng_G(T)$ is the max congestion over all its edges.
- The *spanning tree congestion* of G, $stc(G)$ is the min congestion over all its spanning trees.

Spanning Tree Congestion



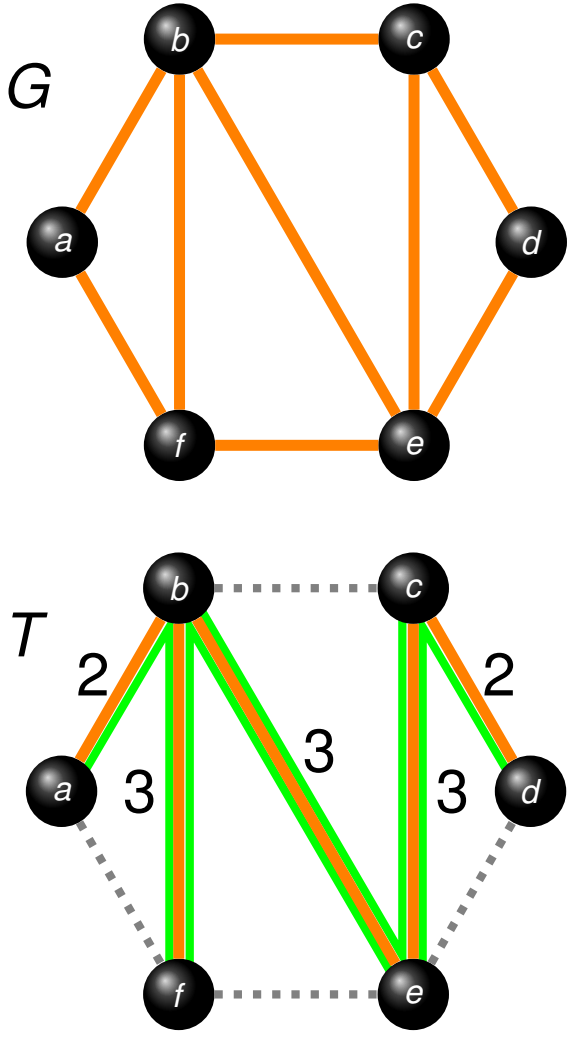
We consider connected graphs only.

Definition (Spanning Tree Congestion)

T: a spanning tree of a graph G.

- The *detour* of $\{u, v\} \in E(G)$ is the unique u–v path in T.
- The *congestion of* $e \in E(T)$, $cng_{G,T}(e)$ is the number of edges in G whose detours contain e.
- The *congestion of* T, $cng_G(T)$ is the max congestion over all its edges.
- The *spanning tree congestion* of G, $stc(G)$ is the min congestion over all its spanning trees.

Spanning Tree Congestion



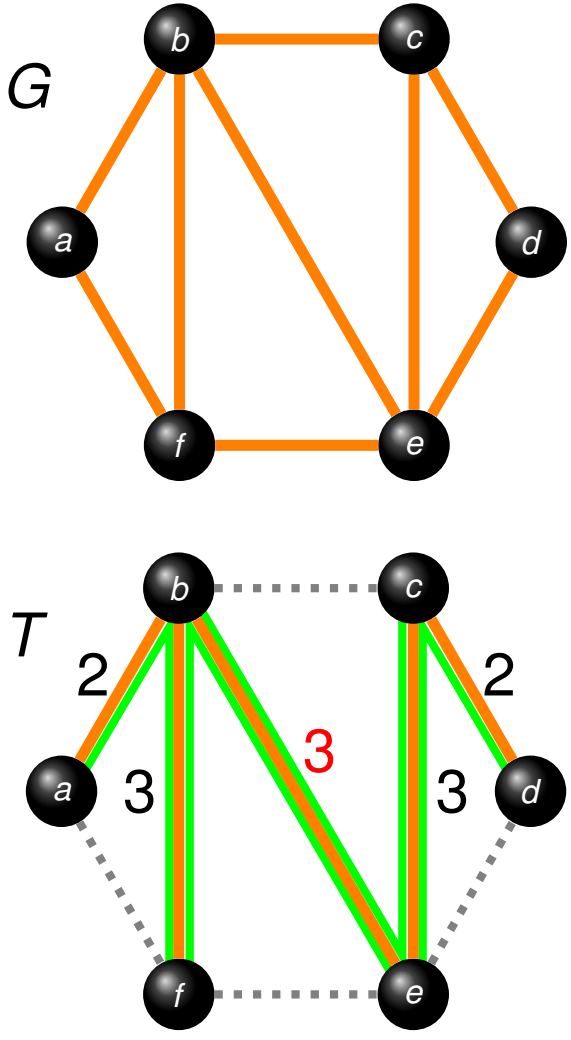
We consider connected graphs only.

Definition (Spanning Tree Congestion)

T: a spanning tree of a graph G.

- The *detour* of $\{u, v\} \in E(G)$ is the unique u–v path in T.
- The *congestion of* $e \in E(T)$, $cng_{G,T}(e)$ is the number of edges in G whose detours contain e.
- The *congestion of* T, $cng_G(T)$ is the max congestion over all its edges.
- The *spanning tree congestion* of G, $stc(G)$ is the min congestion over all its spanning trees.

Spanning Tree Congestion



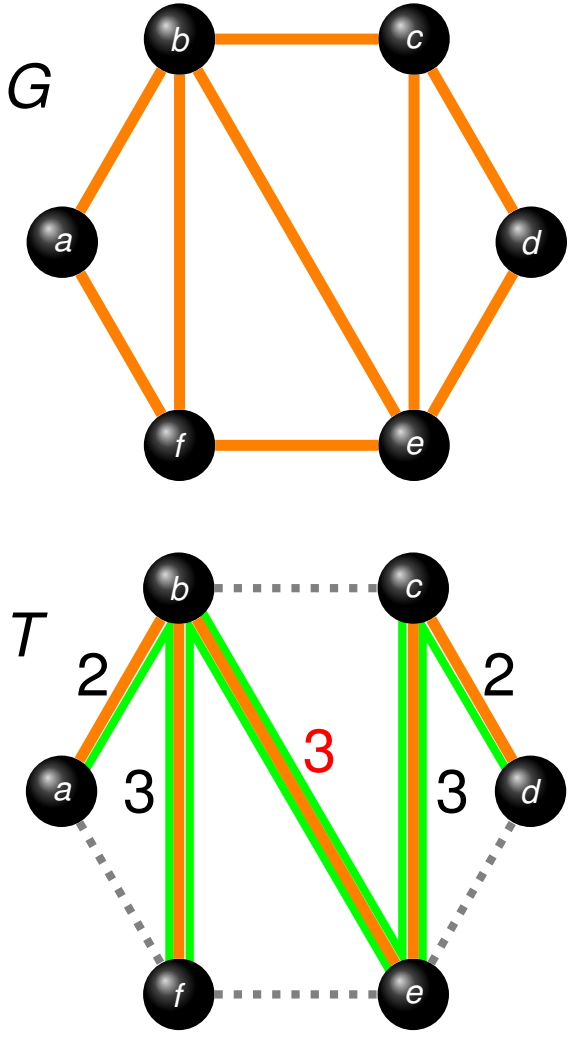
We consider connected graphs only.

Definition (Spanning Tree Congestion)

T: a spanning tree of a graph G.

- The *detour* of $\{u, v\} \in E(G)$ is the unique u–v path in T.
- The *congestion of* $e \in E(T)$, $cng_{G,T}(e)$ is the number of edges in G whose detours contain e.
- The *congestion of* T, $cng_G(T)$ is the max congestion over all its edges.
- The *spanning tree congestion* of G, $stc(G)$ is the min congestion over all its spanning trees.

Spanning Tree Congestion



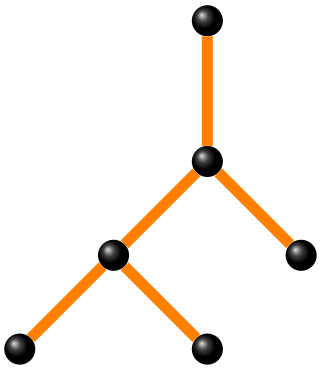
We consider connected graphs only.

Definition (Spanning Tree Congestion)

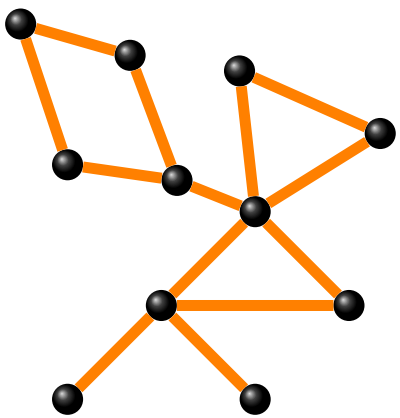
T: a spanning tree of a graph G.

- The *detour* of $\{u, v\} \in E(G)$ is the unique u–v path in T.
- The *congestion of* $e \in E(T)$, $cng_{G,T}(e)$ is the number of edges in G whose detours contain e.
- The *congestion of* T, $cng_G(T)$ is the max congestion over all its edges.
- The *spanning tree congestion* of G, $stc(G)$ is the min congestion over all its spanning trees.

Examples of spanning tree congestion



Tree



Cactus

Examples

- $stc(G) = 1 \iff G$ is a tree
- $stc(G) \leq 2 \iff G$ is a cactus
- $stc(G) \leq 3 \implies G$ is planar

Note: stc is not closed under edge/vertex deletions nor edge contractions.

Problems

Problem: STC

Instance: Connected graph G, positive integer k.

Question: $stc(G) \leq k$?

Problem: k-STC

Instance: Connected graph G.

Question: $stc(G) \leq k$?

Note: k is a fixed constant.

Outline

Known results

Positive (Bodlaender et al. '11)

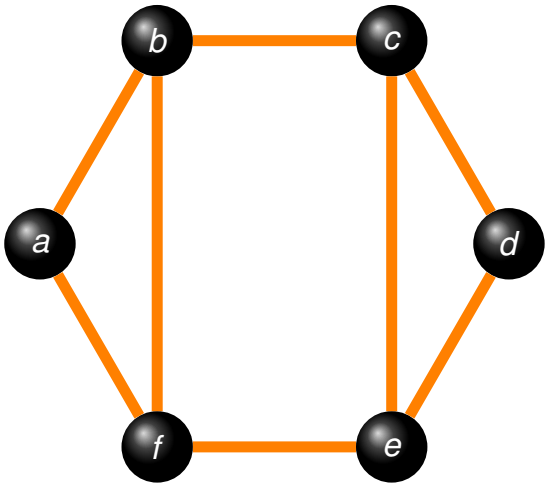
- STC is linear time solvable for outerplanar graphs.
  - Maybe for bounded treewidth graphs, in $O(n^{f(tw)})$ time?
- k-STC is linear time solvable for the following cases:
  - $k \leq 3$,
  - bounded degree graphs,
  - apex-minor-free graphs.

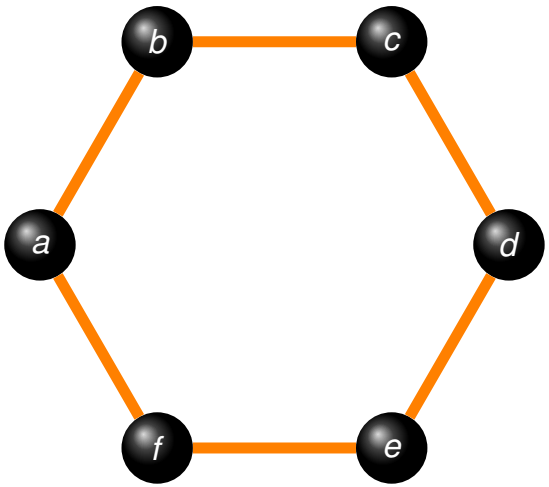
Negative (Bodlaender et al. '11)

- STC is NP-complete for planar graphs.
- For $k \geq 8$, k-STC is NP-complete even for K_6-minor-free graphs with only one vertex of unbounded degree.

Known results are restricted to *sparse* graphs.

Chordal graphs and Chordal bipartite graphs





Two important classes of dense graphs.

Definition (Chordal graphs)

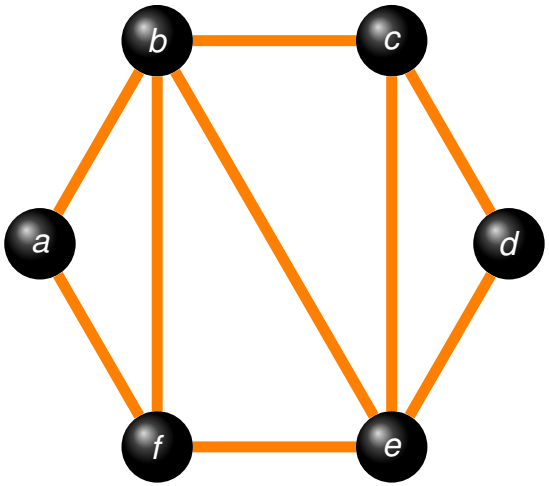
A graph is *chordal* if every induced cycle is of length 3.

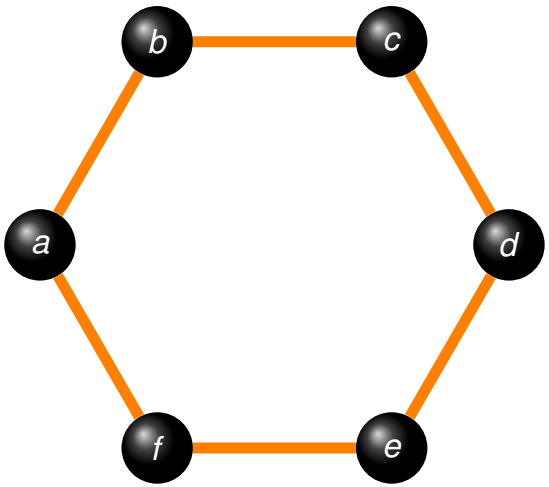
Definition (Chordal bipartite graphs)

A bigraph is *chordal bipartite* if every induced cycle is of length 4.

Chordal and chordal bipartite graphs are *dense* (or *non-sparse*) in some sense. They can contain large (bi)cliques.

Chordal graphs and Chordal bipartite graphs





Two important classes of dense graphs.

Definition (Chordal graphs)

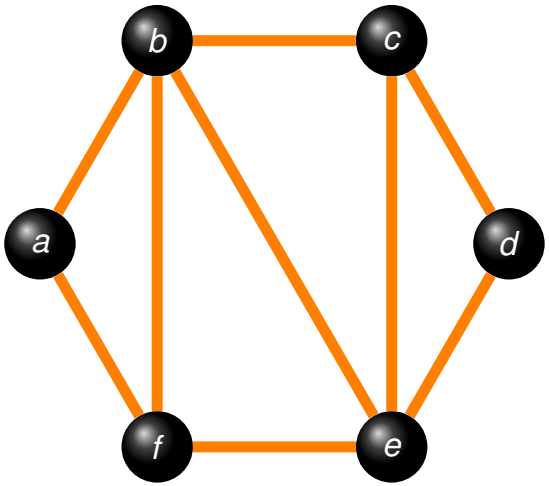
A graph is *chordal* if every induced cycle is of length 3.

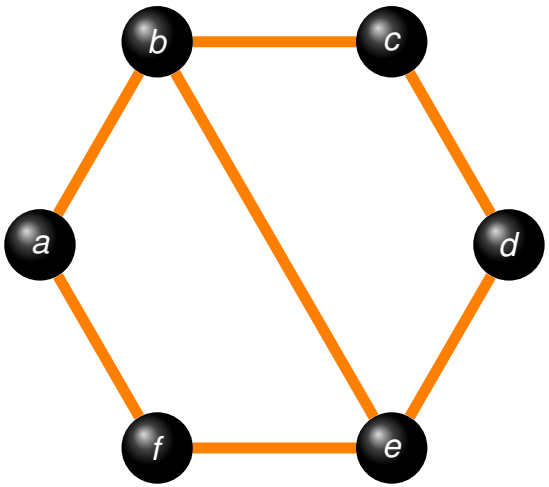
Definition (Chordal bipartite graphs)

A bigraph is *chordal bipartite* if every induced cycle is of length 4.

Chordal and chordal bipartite graphs are *dense* (or *non-sparse*) in some sense. They can contain large (bi)cliques.

Chordal graphs and Chordal bipartite graphs





Two important classes of dense graphs.

Definition (Chordal graphs)

A graph is *chordal* if every induced cycle is of length 3.

Definition (Chordal bipartite graphs)

A bigraph is *chordal bipartite* if every induced cycle is of length 4.

Chordal and chordal bipartite graphs are *dense* (or *non-sparse*) in some sense. They can contain large (bi)cliques.

STC for Non-sparse graphs

STC for non-sparse graphs

- Complete graphs (easy)
- Complete bigraphs (easy)
- Chordal graphs (unknown)
- Chordal bipartite graphs (unknown)

Non-sparseness make the problem easy?

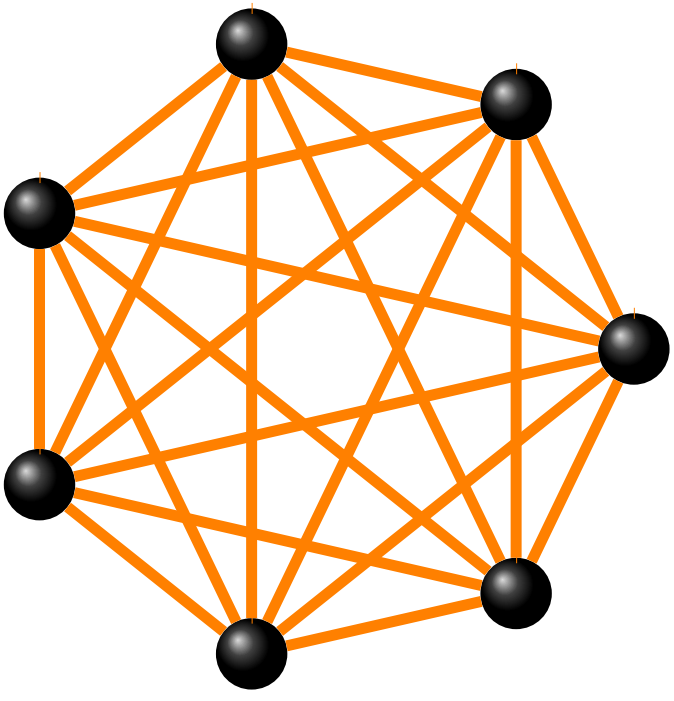
Observation: k-STC is linear time solvable for chordal graphs.

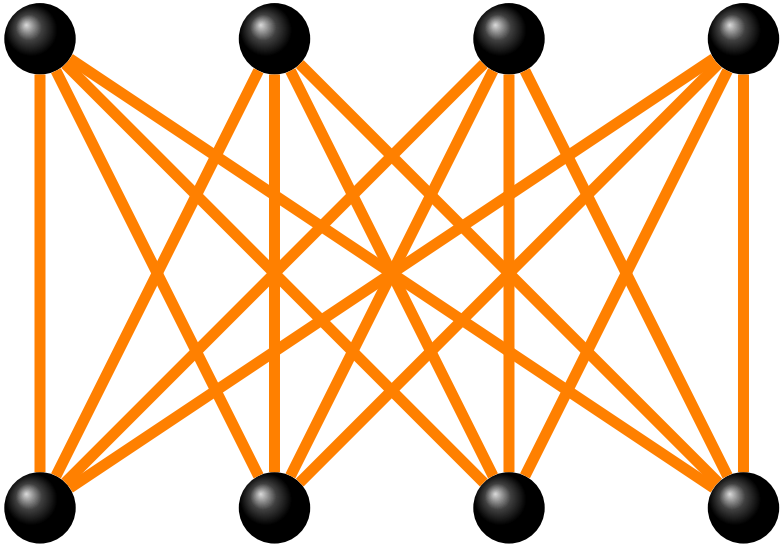
- for chordal graphs, large tw $\Longrightarrow$ large stc
  - large tw $\Longleftrightarrow$ large clique
  - large clique $\Longrightarrow$ large stc
- k-STC is linear time solvable for bounded tw graphs

More on non-sparse graphs

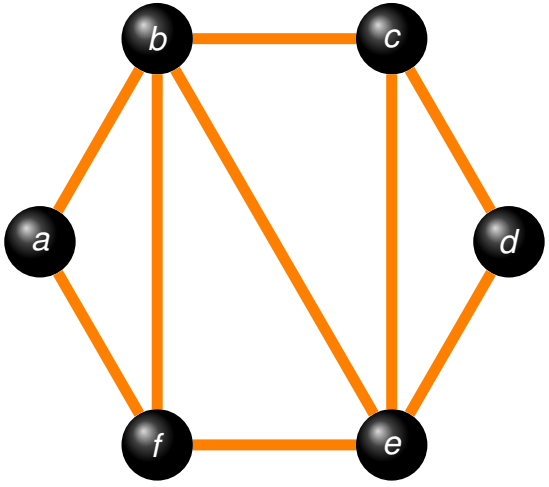
Non-sparse graphs

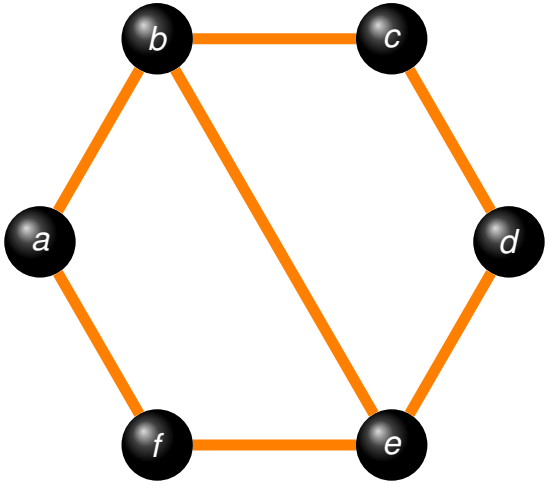
- Complete graphs $\subsetneq$ Split graphs $\subsetneq$ Chordal graphs
- Complete bigraphs $\subsetneq$ Chain graphs $\subsetneq$ Chordal bipartite





Split graphs and Chain graphs





Definition (Split graphs)

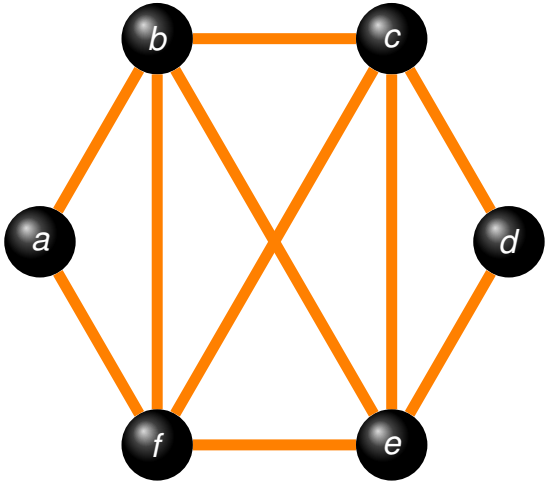
A graph is a *split graph* if its vertex set can be partitioned into an independent set and a clique.

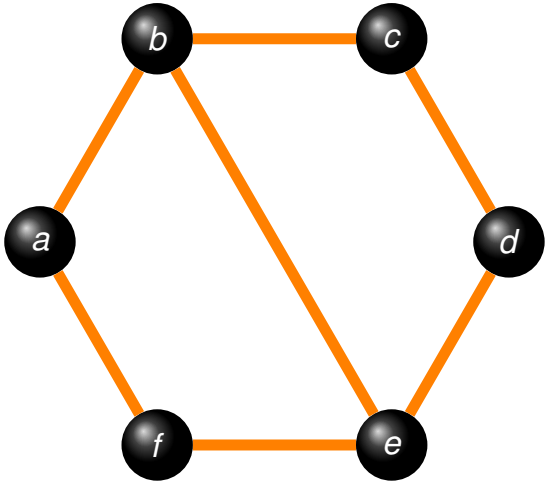
Definition (Chain graphs)

A bigraph is a *chain graph* if it has no independent edge pair.

We prove that STC is NP-complete even for these graphs (thus for chordal and chordal bipartite graphs).

Split graphs and Chain graphs





Definition (Split graphs)

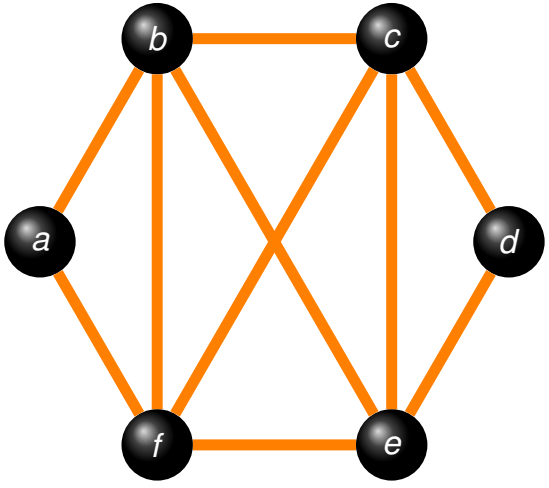
A graph is a *split graph* if its vertex set can be partitioned into an independent set and a clique.

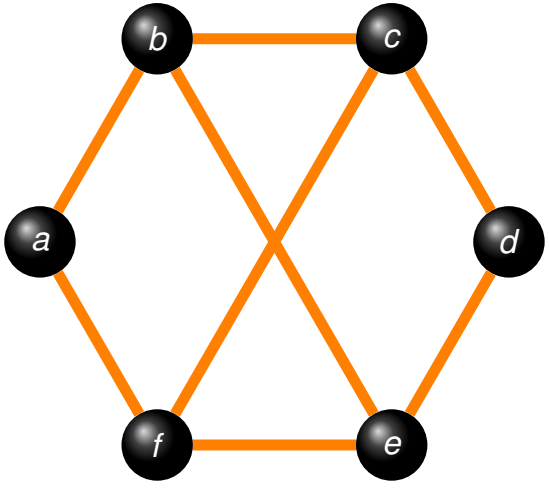
Definition (Chain graphs)

A bigraph is a *chain graph* if it has no independent edge pair.

We prove that STC is NP-complete even for these graphs (thus for chordal and chordal bipartite graphs).

Split graphs and Chain graphs





Definition (Split graphs)

A graph is a *split graph* if its vertex set can be partitioned into an independent set and a clique.

Definition (Chain graphs)

A bigraph is a *chain graph* if it has no independent edge pair.

We prove that STC is NP-complete even for these graphs (thus for chordal and chordal bipartite graphs).

How to cope with the hardness?

Too difficult!

Computing the spanning tree congestion is NP-hard even for very restricted graphs.

Solution

We present a fast exponential-time algorithm.

Outline

Our results

Negative

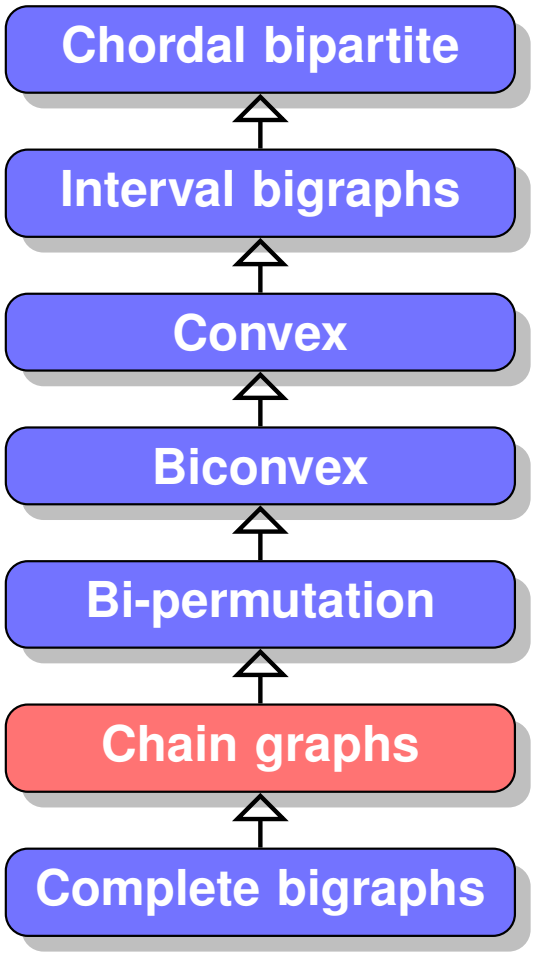
STC is NP-complete for

- split graphs (thus for chordal graphs) and
- chain graphs (thus for chordal bipartite graphs).

Positive

For any connected graph G with n vertices, $stc(G)$ can be determined in $O^*(2^n)$ time.

Remark on our result



Hardness for chain graphs is surprising.

Why?

- Many problems are in P for superclasses.
- Any chain graph has clique-width ≤ 3.
- No graph parameter problem is known to be NP-hard for unweighted chain graphs.*

* To the best of my knowledge. I'm eager to know counter examples.

Outline

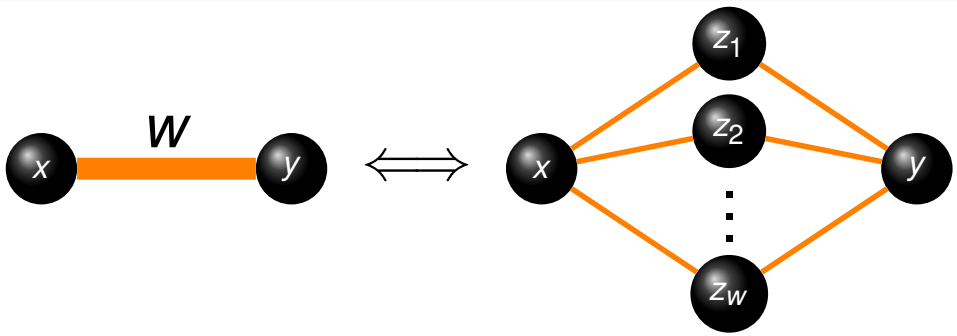
Introducing weighted edges

STC for edge weighted graphs is defined naturally.

- the congestion of e is the sum of the weights of edges whose detours contain e.

Lemma

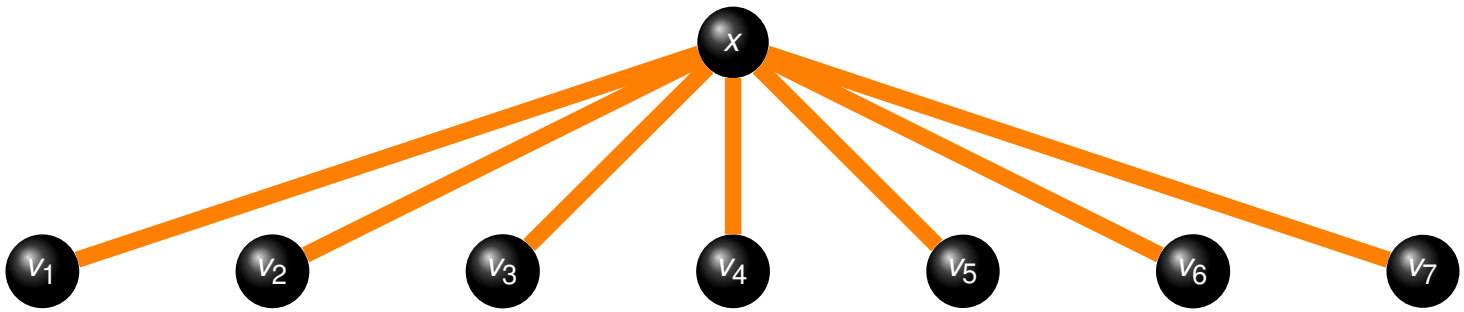
A weighted edge $\{u, v\}$ of weight $w \in \mathbb{N}$ can be simulated by w parallel u–v paths.



Forcing edges

Lemma

Assume $cng_G(T) \leq k$. If all edges incident to x has weight $> k/2$, then all those edges should be in T.



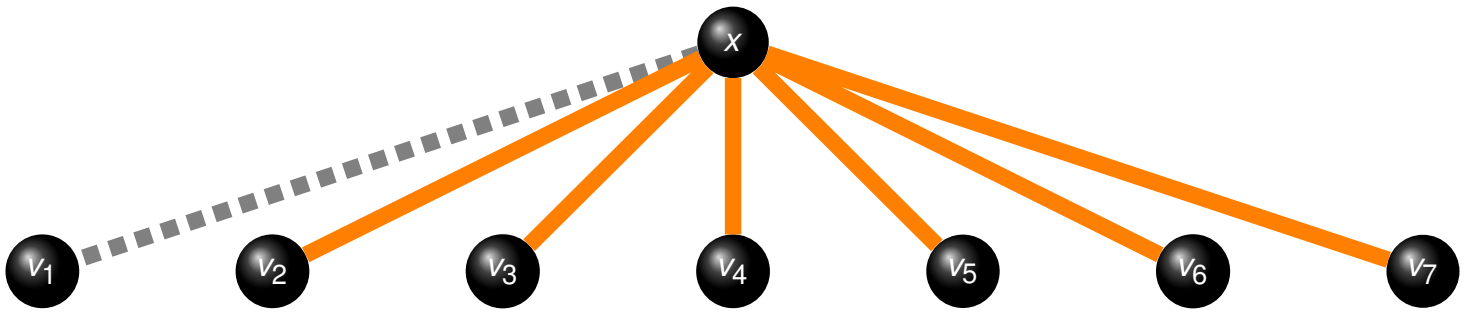
Proof.

If T does not contain an edge $\{x, v_i\}$, then the detour for $\{x, v_i\}$ go through some edge $\{x, v_j\}$ with $i \neq j$. Clearly, the congestion of the edge $\{x, v_j\}$ is more than k. □

Forcing edges

Lemma

Assume $cng_G(T) \leq k$. If all edges incident to x has weight $> k/2$, then all those edges should be in T.



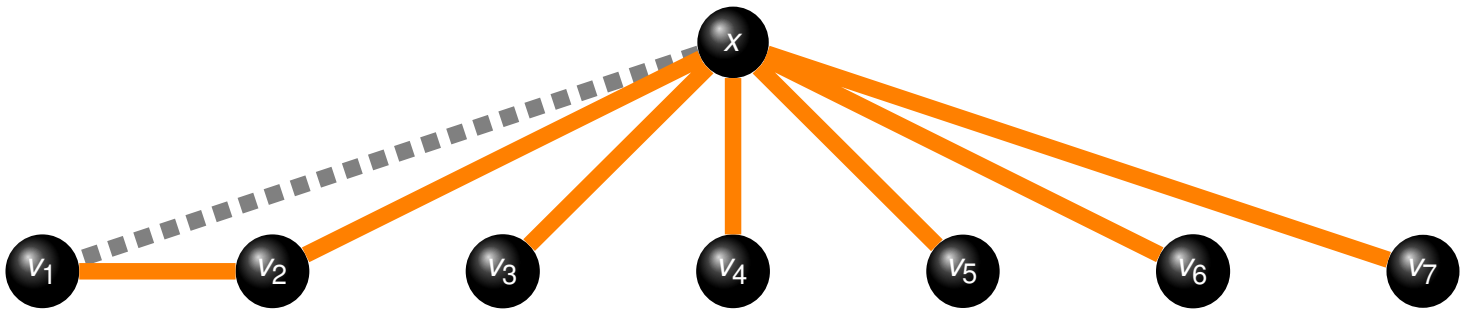
Proof.

If T does not contain an edge $\{x, v_i\}$, then the detour for $\{x, v_i\}$ go through some edge $\{x, v_j\}$ with $i \neq j$. Clearly, the congestion of the edge $\{x, v_j\}$ is more than k. □

Forcing edges

Lemma

Assume $cng_G(T) \leq k$. If all edges incident to x has weight $> k/2$, then all those edges should be in T.



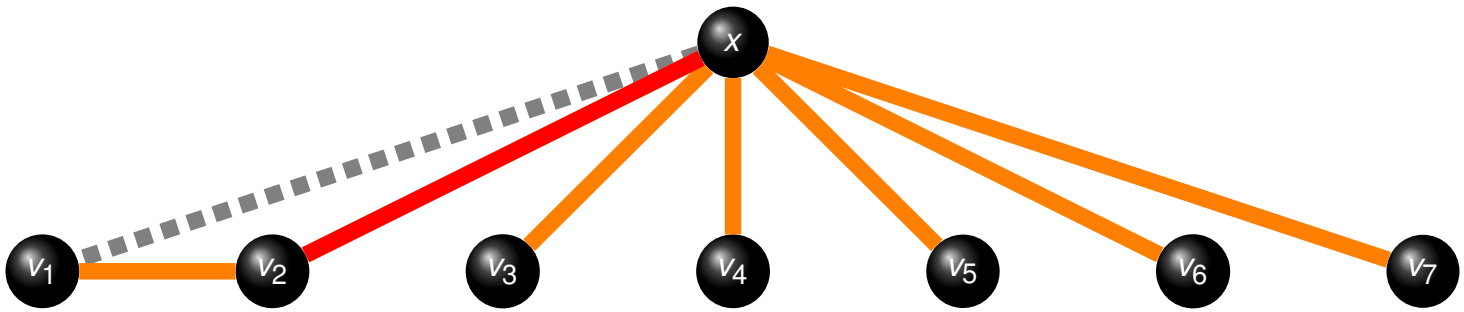
Proof.

If T does not contain an edge $\{x, v_i\}$, then the detour for $\{x, v_i\}$ go through some edge $\{x, v_j\}$ with $i \neq j$. Clearly, the congestion of the edge $\{x, v_j\}$ is more than k. □

Forcing edges

Lemma

Assume $cng_G(T) \leq k$. If all edges incident to x has weight $> k/2$, then all those edges should be in T.



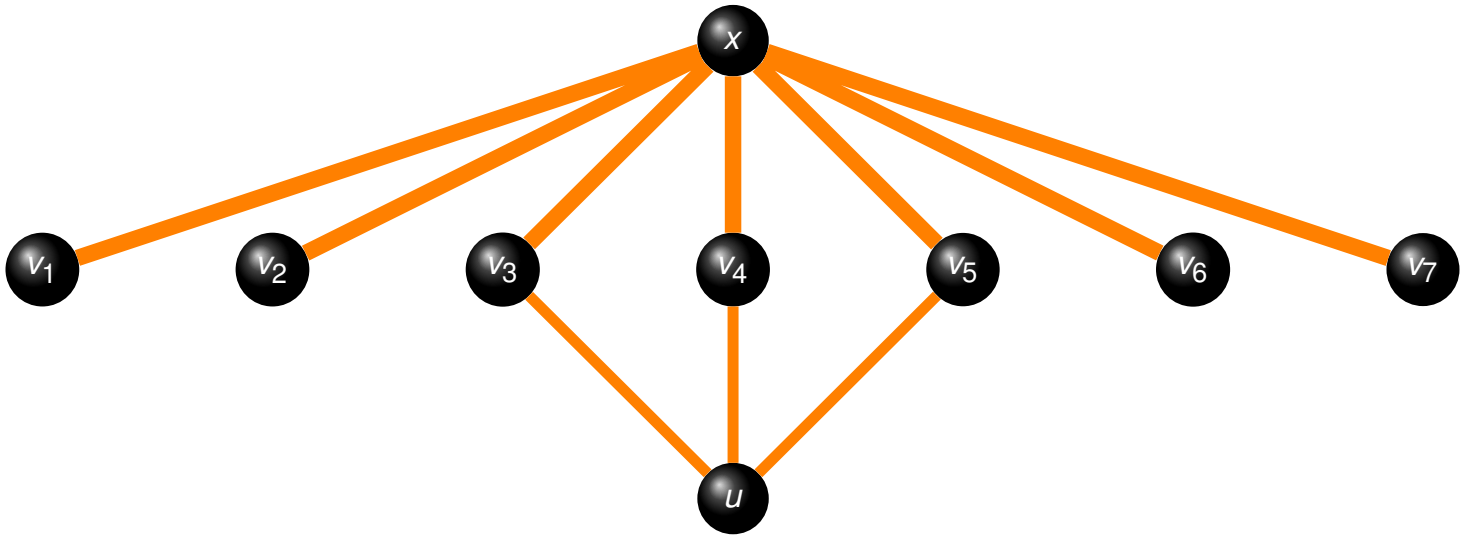
Proof.

If T does not contain an edge $\{x, v_i\}$, then the detour for $\{x, v_i\}$ go through some edge $\{x, v_j\}$ with $i \neq j$. Clearly, the congestion of the edge $\{x, v_j\}$ is more than k. □

Embedding assignment problems

Lemma

If all edges incident to x are forced in T and $N_G(u) \subseteq N_G(x)$, then u must be a leaf of T.

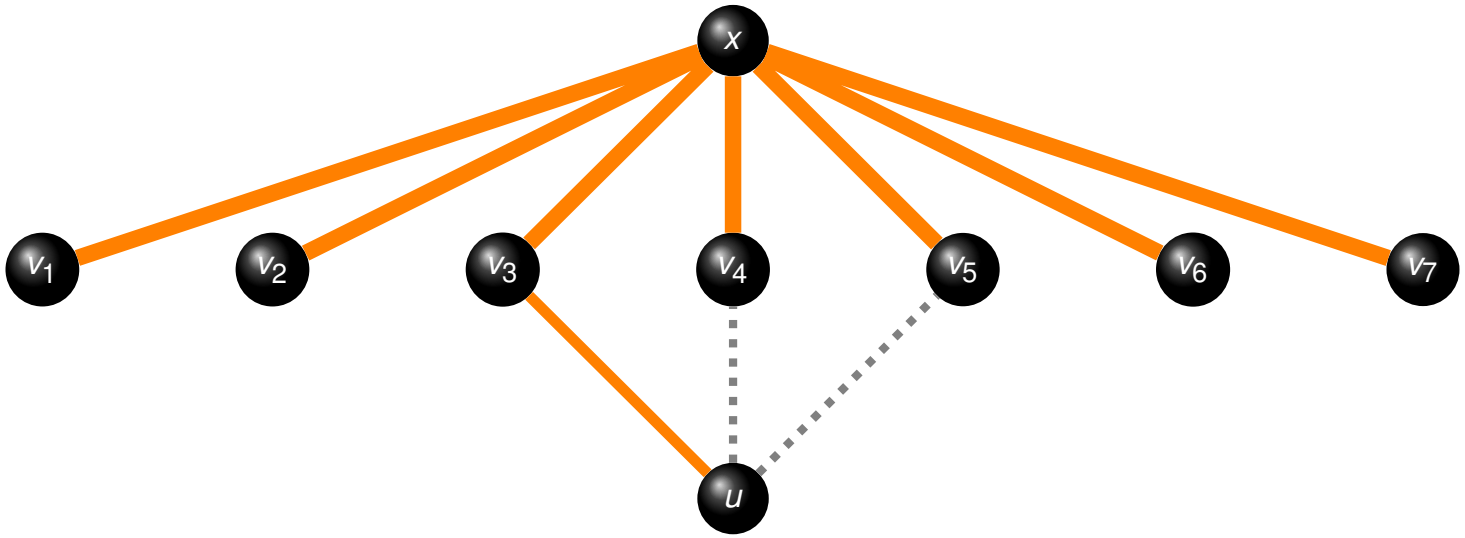


We can *embed* an instance of assignment problems (such as 3-PARTITION).

Embedding assignment problems

Lemma

If all edges incident to x are forced in T and $N_G(u) \subseteq N_G(x)$, then u must be a leaf of T.



We can *embed* an instance of assignment problems (such as 3-PARTITION).

NP-hardness for split graphs and chain graphs

- We can reduce 3-PARTITION to STC for split graphs and chordal bipartite graphs, by carefully constructing a graph based on the ideas.
- We cannot use weighted edges for chain graphs: By introducing weighted edges, we also introduce independent edges in the corresponding unweighted graph.
- We need a new idea to force edges for chain graphs.
  - Embedding 2 copies of an instance of 3-PARTITION.
  - We can force some edges in one of the copies of the instance.

NP-hardness for split graphs and chain graphs

- We can reduce 3-PARTITION to STC for split graphs and chordal bipartite graphs, by carefully constructing a graph based on the ideas.
- We cannot use weighted edges for chain graphs: By introducing weighted edges, we also introduce independent edges in the corresponding unweighted graph.
- We need a new idea to force edges for chain graphs.
  - Embedding 2 copies of an instance of 3-PARTITION.
  - We can force some edges in one of the copies of the instance.

NP-hardness for split graphs and chain graphs

- We can reduce 3-PARTITION to STC for split graphs and chordal bipartite graphs, by carefully constructing a graph based on the ideas.
- We cannot use weighted edges for chain graphs: By introducing weighted edges, we also introduce independent edges in the corresponding unweighted graph.
- We need a new idea to force edges for chain graphs.
  - Embedding 2 copies of an instance of 3-PARTITION.
  - We can force some edges in one of the copies of the instance.

Outline

Idea of our algorithm

A naive approach is to examine all spanning trees.
This runs in $O^*(2^m)$ or $O^*(n^n)$ time.

Too slow. We can do better!

Main idea

Instead of all spanning trees, we examine all possible combinations of vertex subsets.

The number of pairs of disjoint vertex subsets is 3^n. A dynamic programming approach gives an $O^*(3^n)$-time algorithm.

Improving the running time

The *fast subset convolution* method gives an $O^*(2^n)$-time algorithm.

Outline

Conclusion

Our results

- NP-hardness for very restricted dense graphs
- $O^*(2^n)$-time exact algorithm for general graphs

Outline

Open problems

- STC for cographs (graphs of clique-width ≤ 2)
  - 3-approximation is easy
  - easy for chordal cographs ($=$ trivially perfect graphs)
- STC for interval graphs
  - poper interval graphs, cochain graphs
- Is STC in FPT when parameterized by tw?
  - Maybe in XP (by standard DP?)
  - If stc is also a parameter, then FPT.
- Approximation
  - There is no PTAS unless P $=$ NP.
- k-STC for $4 \leq k \leq 7$

Open problems

- STC for cographs (graphs of clique-width ≤ 2)
  - 3-approximation is easy
  - easy for chordal cographs ($=$ trivially perfect graphs)
- STC for interval graphs
  - poper interval graphs, cochain graphs
- Is STC in FPT when parameterized by tw?
  - Maybe in XP (by standard DP?)
  - If stc is also a parameter, then FPT.
- Approximation
  - There is no PTAS unless P $=$ NP.
- k-STC for $4 \leq k \leq 7$

Open problems

- STC for cographs (graphs of clique-width ≤ 2)
  - 3-approximation is easy
  - easy for chordal cographs ($=$ trivially perfect graphs)
- STC for interval graphs
  - poper interval graphs, cochain graphs
- Is STC in FPT when parameterized by tw?
  - Maybe in XP (by standard DP?)
  - If stc is also a parameter, then FPT.
- Approximation
  - There is no PTAS unless P $=$ NP.
- k-STC for $4 \leq k \leq 7$

Open problems

- STC for cographs (graphs of clique-width ≤ 2)
  - 3-approximation is easy
  - easy for chordal cographs ($=$ trivially perfect graphs)
- STC for interval graphs
  - poper interval graphs, cochain graphs
- Is STC in FPT when parameterized by tw?
  - Maybe in XP (by standard DP?)
  - If stc is also a parameter, then FPT.
- Approximation
  - There is no PTAS unless P $=$ NP.
- k-STC for $4 \leq k \leq 7$

Open problems

- STC for cographs (graphs of clique-width ≤ 2)
  - 3-approximation is easy
  - easy for chordal cographs ($=$ trivially perfect graphs)
- STC for interval graphs
  - poper interval graphs, cochain graphs
- Is STC in FPT when parameterized by tw?
  - Maybe in XP (by standard DP?)
  - If stc is also a parameter, then FPT.
- Approximation
  - There is no PTAS unless P $=$ NP.
- k-STC for $4 \leq k \leq 7$