On Breaking k-Trusses

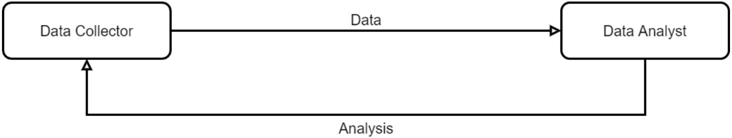
Dutch Seminar on Optimization

Huiping Chen[1], Alessio Conte[2], Roberto Grossi[2], Grigorios Loukides[1], Solon P. Pissis[3,4,5], **Michelle Sweering[4]**

[1]King's College London
[2]University of Pisa
[3]ERABLE Team, INRIA, Lyon
[4]CWI, Amsterdam
[5]Vrije Universiteit, Amsterdam

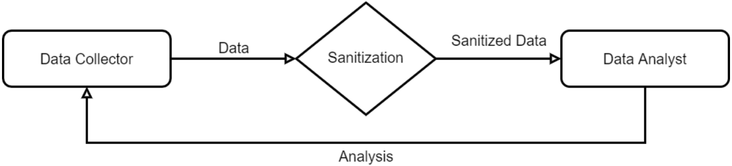
25th March 2021

Data Sanitization

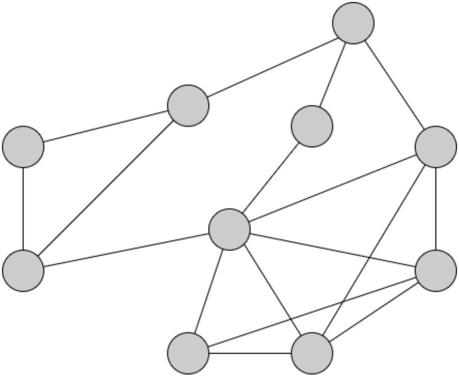


Data Sanitization

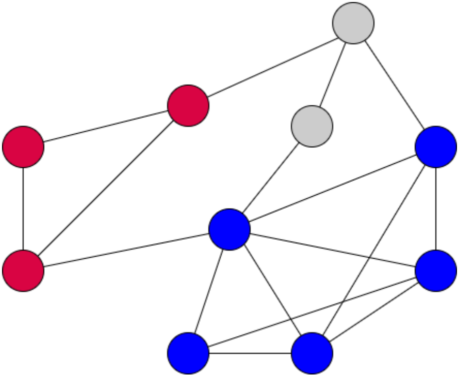
- Privacy Constraints
- Utility Properties



Communities



Communities



Community Breaking

- Maintaining communities in social networks

Community Breaking

- Maintaining communities in social networks
- Assessing resilience to attacks and errors in communication networks

Community Breaking

- Maintaining communities in social networks
- Assessing resilience to attacks and errors in communication networks
- Hiding membership to communities in social networks

Community Breaking

- Maintaining communities in social networks
- Assessing resilience to attacks and errors in communication networks
- Hiding membership to communities in social networks
- Preventing detection of confidential communities

k-Trusses

Definition (k-Truss)

A k-truss is a subgraph in which each edge is contained in at least $k-2$ triangles of the subgraph.

k-Trusses

Definition (k-Truss)

A k-truss is a subgraph in which each edge is contained in at least $k-2$ triangles of the subgraph.

Problem (MIN-k-TBS)

Find a minimum set of edges to delete to remove all k-trusses.

k-Trusses

Definition (k-Truss)

A k-truss is a subgraph in which each edge is contained in at least $k - 2$ triangles of the subgraph.

Problem (MIN-k-TBS)

Find a minimum set of edges to delete to remove all k-trusses.

Problem (MIN-k-CBS)

Find a minimum set of edges incident to U such that no node in U is in a k-truss.

NP-hardness

Theorem

The problems MIN-k-TBS and MIN-k-CBS are NP-hard.

NP-hardness

Theorem

The problems MIN-k-TBS and MIN-k-CBS are NP-hard.

Theorem

For $\delta > 0$, both MIN-k-TBS and MIN-k-CBS cannot be approximated within an additive term of $|V|^{2-\delta}$, unless $P = NP$.

NP-hardness

Theorem

The problems MIN-k-TBS and MIN-k-CBS are NP-hard.

Theorem

For $\delta > 0$, both MIN-k-TBS and MIN-k-CBS cannot be approximated within an additive term of $|V|^{2-\delta}$, unless $P = NP$.

Theorem

For $\epsilon > 0$, both MIN-k-TBS and MIN-k-CBS cannot be approximated within a multiplicative factor of $(k - 2 - \epsilon)$, assuming the unique games conjecture.

NP-hardness

Theorem

For all $k \geq 3$, MIN-k-TBS is NP-hard.

NP-hardness

Theorem

For all $k \geq 3$, MIN-k-TBS is NP-hard.

Proof.

For $k = 3$, we want to break all triangles. This is known to be NP-hard.

NP-hardness

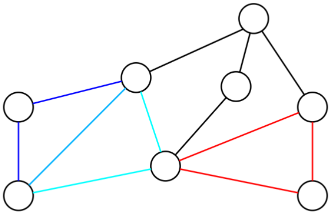
Theorem

For all $k \geq 3$, MIN-k-TBS is NP-hard.

Proof.

For $k = 3$, we want to break all triangles. This is known to be NP-hard.

- Let G be an instance of MIN-3-TBS.



NP-hardness

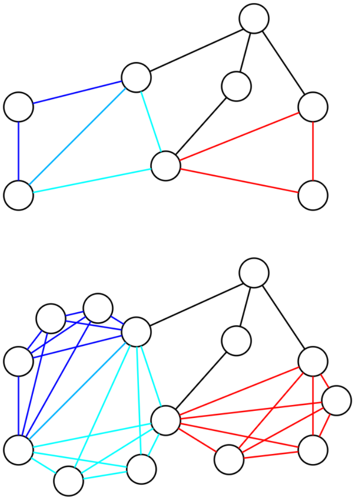
Theorem

For all $k \geq 3$, MIN-k-TBS is NP-hard.

Proof.

For $k = 3$, we want to break all triangles. This is known to be NP-hard.

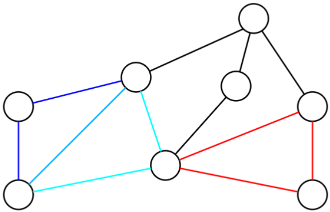
- Let G be an instance of MIN-3-TBS.
- Turn each triangle in G into a k-clique by adding $k - 3$ vertices to obtain G'.



NP-hardness

Theorem

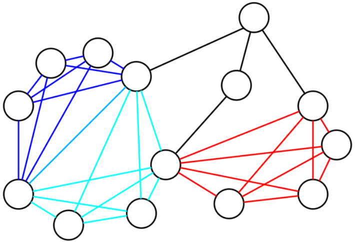
For all $k \geq 3$, MIN-k-TBS is NP-hard.



Proof.

For $k = 3$, we want to break all triangles. This is known to be NP-hard.

- Let G be an instance of MIN-3-TBS.
- Turn each triangle in G into a k-clique by adding $k - 3$ vertices to obtain G'.
- MIN-k-TBS in G' is equivalent to MIN-3-TBS in G.



NP-hardness

Theorem

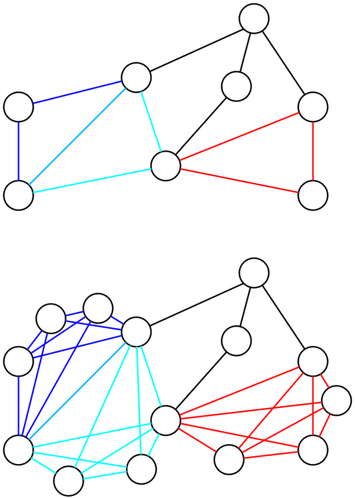
For all $k \geq 3$, MIN-k-TBS is NP-hard.

Proof.

For $k = 3$, we want to break all triangles. This is known to be NP-hard.

- Let G be an instance of MIN-3-TBS.
- Turn each triangle in G into a k-clique by adding $k - 3$ vertices to obtain G'.
- MIN-k-TBS in G' is equivalent to MIN-3-TBS in G.

Therefore MIN-k-TBS is NP-hard. $\square$



Algorithms

- Exact Algorithm

Algorithms

- Exact Algorithm
- Heuristics

Truss Decomposition

Definition (k-Truss)

A k-truss is a subgraph in which each edge is contained in at least $k-2$ triangles of the subgraph.

Algorithm

$k = 2$ (every graph G is a 2-truss)
while G non-empty:

- $k = k + 1$
- while there is an edge in less than $k-2$ triangles:
  - remove that edge
- G is currently the maximum k-truss

Dynamic Truss Update for Edge Deletion

Definition (Triangle trussness)

A triangle has trussness k if it appears in a k-truss, i.e. all three of its edges have trussness k.

Dynamic Truss Update for Edge Deletion

Definition (Triangle trussness)

A triangle has trussness k if it appears in a k-truss, i.e. all three of its edges have trussness k.

Update Algorithm

- Similar Algorithm
- Store the triangles each edge is in.
- Store the number of triangles of trussness k each edge is in.
- Time is proportional to the updated number of triangles.
- Propagation can cause $O(\mathcal{T}_G)$ time per update.
- Very good amortized time complexity $O(t(G) \cdot \mathcal{T}_G)$

Exact Algorithm

- We want to find a minimum E' which intersects all k-trusses

Exact Algorithm

- We want to find a minimum E' which intersects all k-trusses
- Every k-truss contains a minimal k-truss

Exact Algorithm

- We want to find a minimum E' which intersects all k-trusses
- Every k-truss contains a minimal k-truss
- $\implies$ we want to find E' which intersects all minimal k-trusses

Exact Algorithm

- We want to find a minimum E' which intersects all k-trusses
- Every k-truss contains a minimal k-truss
- $\implies$ we want to find E' which intersects all minimal k-trusses

Idea

- List all minimal k-trusses

Exact Algorithm

- We want to find a minimum E' which intersects all k-trusses
- Every k-truss contains a minimal k-truss
- $\implies$ we want to find E' which intersects all minimal k-trusses

Idea

- List all minimal k-trusses
- Find a minimum hypergraph transversal

Max-Truss Breaking Heuristic (MBH)

- Let k' be the highest number such that G contains a k'-truss.
- Let M be the maximal k'-truss.
- Let $|\mathrm{TRI}_{\geq k'}(M, e)|$ be the number of triangles in M containing edge e.

Max-Truss Breaking Heuristic (MBH)

- Let k' be the highest number such that G contains a k'-truss.
- Let M be the maximal k'-truss.
- Let $|\mathrm{TRI}_{\geq k'}(M, e)|$ be the number of triangles in M containing edge e.

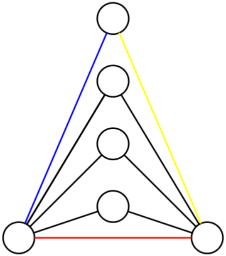
While $k' \geq k$:

MBH_S Delete the edge with the highest $|\mathrm{TRI}_{\geq k}(M, e)|$

MBH_C Delete the edge in the highest $|\mathrm{TRI}_{\geq k}(M, e)|/|\mathrm{TRI}_{<k}(M, e)|$

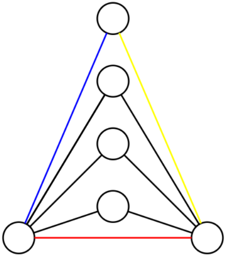
Save the Neighbours Heuristic (SNH)

- Give more weight to edges which are in many triangles
- Give lower weight to triangles whose edges have high support



Save the Neighbours Heuristic (SNH)

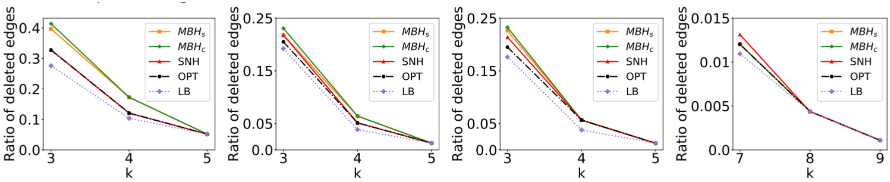
- Give more weight to edges which are in many triangles
- Give lower weight to triangles whose edges have high support



Delete edge with minimal

$$\sum_{\{e,f,g\}\in \mathrm{TRI}_{\geq k}} \frac{|\mathrm{TRI}_{\geq k}(M,e)|}{\max(|\mathrm{TRI}_{\geq k}(M,f)|-k+2,1)}+\frac{|\mathrm{TRI}_{\geq k}(M,e)|}{\max(|\mathrm{TRI}_{\geq k}(M,g)|-k+2,1)}$$

Small Networks



(a) TRIBES (b) KARATE (c) DOLPHINS (d) NETSCIENCE

Benchmarking Large Networks

Problem

How can we compare the accuracy of these heuristics on large data?

Benchmarking Large Networks

Problem

How can we compare the accuracy of these heuristics on large data?

Idea

Partition the graph into cliques.
Find a lower bound for each clique.

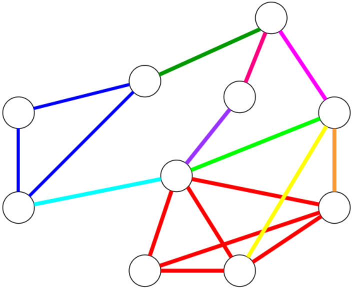
Benchmarking Large Networks

Problem

How can we compare the accuracy of these heuristics on large data?

Idea

Partition the graph into cliques.
Find a lower bound for each clique.



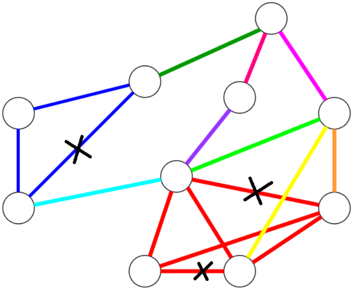
Benchmarking Large Networks

Problem

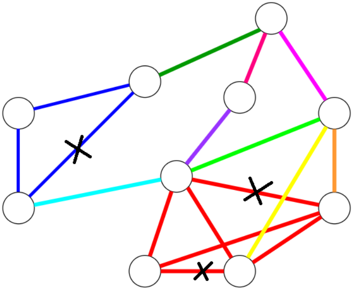
How can we compare the accuracy of these heuristics on large data?

Idea

Partition the graph into cliques.
Find a lower bound for each clique.



Benchmarking Large Networks



Problem

How can we compare the accuracy of these heuristics on large data?

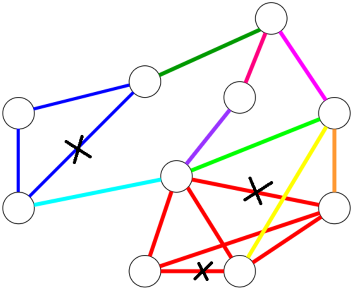
Idea

Partition the graph into cliques.
Find a lower bound for each clique.

Theorem (Turán)

A graph on n vertices and more than $\frac{(k-2)}{(k-1)} \cdot \frac{n^2}{2}$ edges contains a k-clique.

Benchmarking Large Networks



Problem

How can we compare the accuracy of these heuristics on large data?

Idea

Partition the graph into cliques.
Find a lower bound for each clique.

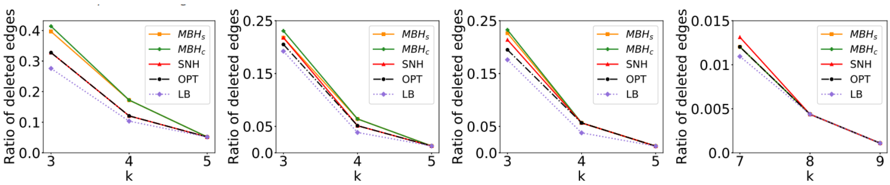
Theorem (Turán)

A graph on n vertices and more than $\frac{(k-2)}{(k-1)} \cdot \frac{n^2}{2}$ edges contains a k-clique.

Theorem (Conte et al.)

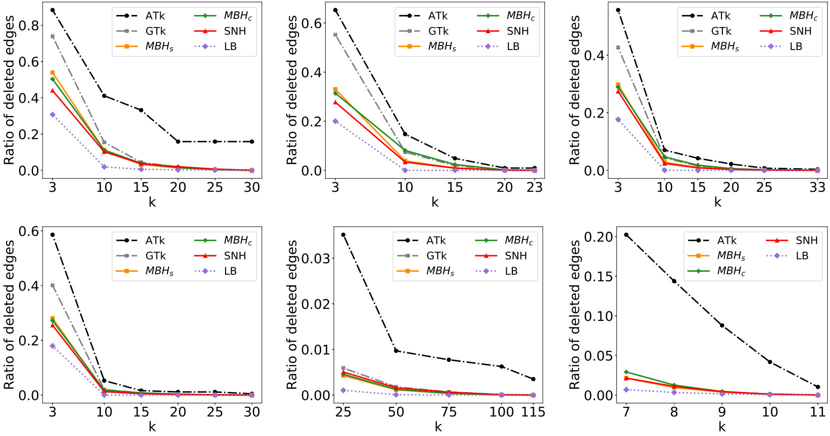
A graph with m edges and $\mathcal{T}_G$ triangles has trussness at least $\mathcal{T}_G/m + 1$.

Small Networks



(a) TRIBES **(b) KARATE** **(c) DOLPHINS** **(d) NETSCIENCE**

Benchmarking Large Networks



Benchmarking Large Networks