Protecting the Connectivity of a Graph under Non-uniform Edge Failures

Felix Hommelsheim[1], **Zhenwei Liu**[1,2], Nicole Megow[1], Guochuan Zhang[2]

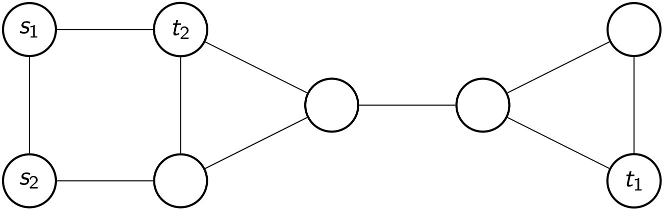
[1]University of Bremen, Germany

[2]Zhejiang University, China

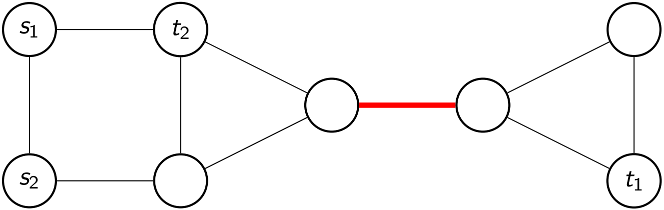
STACS 2025, Jena, Germany

March 05, 2025

Problem Definition

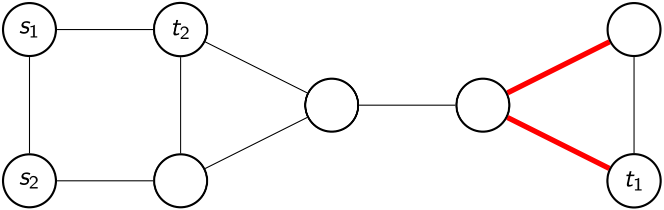


Problem Definition



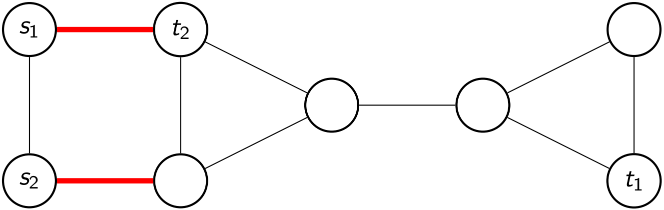
An attacker aims to disconnect some (s_i, t_i) by removing at most $q = 2$ edges

Problem Definition



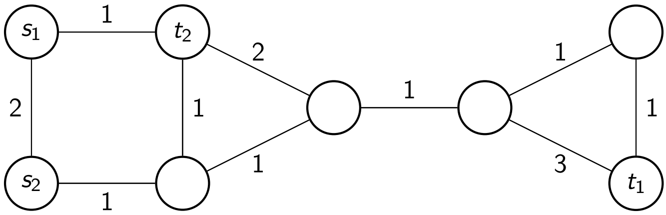
An attacker aims to disconnect some (s_i, t_i) by removing at most $q = 2$ edges

Problem Definition



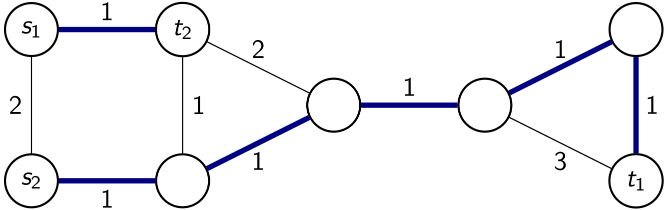
An attacker aims to disconnect some (s_i, t_i) by removing at most $q = 2$ edges

Problem Definition



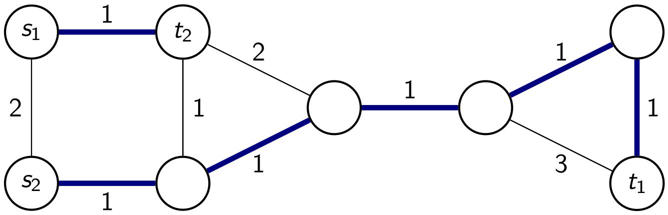
We (as the defender) protect the edges in advance by paying cost

Problem Definition



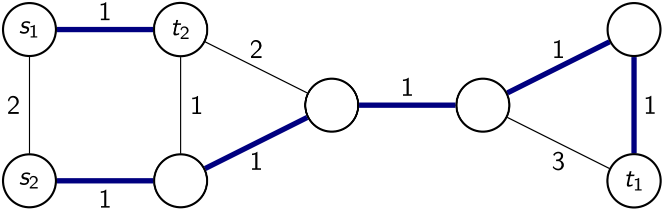
We (as the defender) protect the edges in advance by paying cost

Problem Definition



In short, we robustify the network assuming at most q edge failures

Problem Definition



In short, we robustify the network assuming at most q edge failures

Input:

- An undirected graph $G = (V, E)$ with edge cost $c : E \to \mathbb{R}_{\geq 0}$.
- The attacker's budget q.

Output: Min-cost protected edge set $F \subseteq E$ s.t. G retains certain connectivity properties after removals of arbitrary $\leq q$ unprotected edges.

Problem Definition

Input:

- An undirected graph $G = (V, E)$ with edge cost $c : E \to \mathbb{R}_+$.
- The attacker's budget q.

Output: Min-cost protected edge set $F \subseteq E$ s.t. G retains certain connectivity properties after removals of arbitrary $\leq q$ unprotected edges.

Problem Definition

Input:

- An undirected graph $G = (V, E)$ with edge cost $c : E \to \mathbb{R}_+$.
- The attacker's budget q.

Output: Min-cost protected edge set $F \subseteq E$ s.t. G retains certain connectivity properties after removals of arbitrary $\leq q$ unprotected edges.

Different connectivity properties: p-edge-connectivity

- between multiple terminal-pairs $(s_1, t_1), \ldots, (s_k, t_k)$: (p, q)-SCP.

Problem Definition

Input:

- An undirected graph $G = (V, E)$ with edge cost $c : E \to \mathbb{R}_+$.
- The attacker's budget q.

Output: Min-cost protected edge set $F \subseteq E$ s.t. G retains certain connectivity properties after removals of arbitrary $\leq q$ unprotected edges.

Different connectivity properties: p-edge-connectivity

- between multiple terminal-pairs $(s_1, t_1), \ldots, (s_k, t_k)$: (p, q)-SCP.
- over the entire graph: (p, q)-GCP.
- between given (s, t): (p, q)-stCP.

Problem Definition

Input:

- An undirected graph $G = (V, E)$ with edge cost $c : E \to \mathbb{R}_+$.
- The attacker's budget q.

Output: Min-cost protected edge set $F \subseteq E$ s.t. G retains certain connectivity properties after removals of arbitrary $\leq q$ unprotected edges.

Different connectivity properties: p-edge-connectivity

- between multiple terminal-pairs $(s_1, t_1), \ldots, (s_k, t_k)$: (p, q)-SCP.
- over the entire graph: (p, q)-GCP.
- between given (s, t): (p, q)-stCP. $\to$ a slightly more general model Preventing Small (s, t)-Cut [Grüttemeier, Komusiewicz, Morawietz and Sommer, WG21]

Related network design problems

Consider $q \geq |E|$, i.e., the attacker can remove all unprotected edges.

Related network design problems

Consider $q \geq |E|$, i.e., the attacker can remove all unprotected edges.

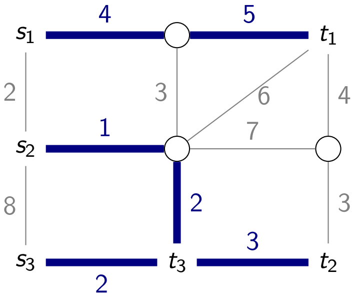
Survivable Network Design: Select a minimum-cost spanning subgraph from a given graph, satisfying certain connectivity requirements.

Related network design problems

Consider $q \geq |E|$, i.e., the attacker can remove all unprotected edges.

Survivable Network Design: Select a minimum-cost spanning subgraph from a given graph, satisfying certain connectivity requirements.

- Minimum Steiner Tree (Forest) [(Bern and Plassmann, IPL 1989)].

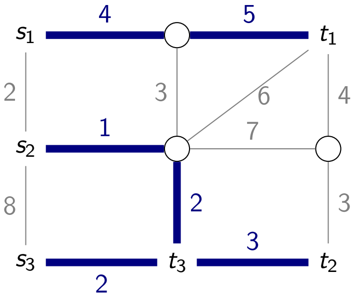


Related network design problems

Consider $q \geq |E|$, i.e., the attacker can remove all unprotected edges.

Survivable Network Design: Select a minimum-cost spanning subgraph from a given graph, satisfying certain connectivity requirements.

- Minimum Steiner Tree (Forest) [(Bern and Plassmann, IPL 1989)]. $(1, q)$-SCP is APX-hard.

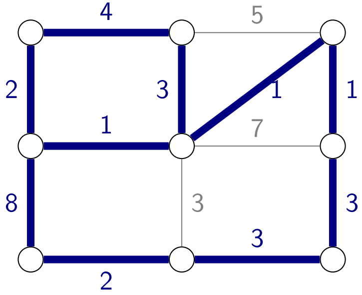


Related network design problems

Consider $q \geq |E|$, i.e., the attacker can remove all unprotected edges.

Survivable Network Design: Select a minimum-cost spanning subgraph from a given graph, satisfying certain connectivity requirements.

- Minimum Steiner Tree (Forest) [(Bern and Plassmann, IPL 1989)]. $(1, q)$-SCP is APX-hard.
- Minimum p-edge-connected Spanning Subgraph [Czumaj et al., SODA 1999].

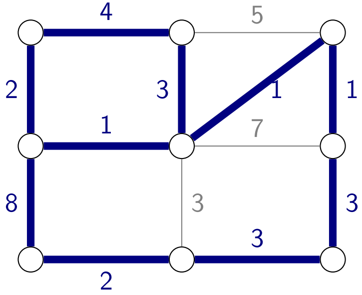


Related network design problems

Consider $q \geq |E|$, i.e., the attacker can remove all unprotected edges.

Survivable Network Design: Select a minimum-cost spanning subgraph from a given graph, satisfying certain connectivity requirements.

- Minimum Steiner Tree (Forest) [(Bern and Plassmann, IPL 1989)]. $(1, q)$-SCP is APX-hard.
- Minimum p-edge-connected Spanning Subgraph [Czumaj et al., SODA 1999]. (p, q)-GCP is APX-hard.

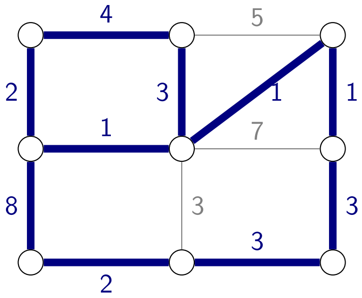


Related network design problems

Consider $q \geq |E|$, i.e., the attacker can remove all unprotected edges.

Survivable Network Design: Select a minimum-cost spanning subgraph from a given graph, satisfying certain connectivity requirements.

- Minimum Steiner Tree (Forest) [(Bern and Plassmann, IPL 1989)]. $(1, q)$-SCP is APX-hard.
- Minimum p-edge-connected Spanning Subgraph [Czumaj et al., SODA 1999]. (p, q)-GCP is APX-hard.
- The NP-hardness of $(1, q)$-GCP is unknown (equivalent to Minimum Spanning Tree when $q \geq |E|$).



Related network design problems

Flexible Network Design [Adjiashvili, Hommelsheim, Mühlenthaler, MP22]:

- Given an undirected graph G with edge .
- The edges are either safe or unsafe (non-uniform failure model).

Related network design problems

Flexible Network Design [Adjiashvili, Hommelsheim, Mühlenthaler, MP22]:

- Given an undirected graph G with edge .
- The edges are either safe or unsafe (non-uniform failure model).
- Select a minimum-cost spanning subgraph $G'(V, F)$.

Related network design problems

Flexible Network Design [Adjiashvili, Hommelsheim, Mühlenthaler, MP22]:

- Given an undirected graph G with edge .
- The edges are either safe or unsafe (non-uniform failure model).
- Select a minimum-cost spanning subgraph $G'(V, F)$.
- G' retains p-edge-connectivity after $\leq q$ failures of unsafe edges.

Related network design problems

Flexible Network Design [Adjiashvili, Hommelsheim, Mühlenthaler, MP22]:

- Given an undirected graph G with edge .
- The edges are either safe or unsafe (non-uniform failure model).
- Select a minimum-cost spanning subgraph $G'(V, F)$.
- G' retains p-edge-connectivity after $\leq q$ failures of unsafe edges.
- Approximation for constant p or q [Bansal et al., ICALP23] [Chekuri et al., ICALP23].

Related network design problems

Flexible Network Design [Adjiashvili, Hommelsheim, Mühlenthaler, MP22]:

- Given an undirected graph G with edge .
- The edges are either safe or unsafe (non-uniform failure model).
- Select a minimum-cost spanning subgraph $G'(V, F)$.
- G' retains p-edge-connectivity after $\leq q$ failures of unsafe edges.
- Approximation for constant p or q [Bansal et al., ICALP23] [Chekuri et al., ICALP23].

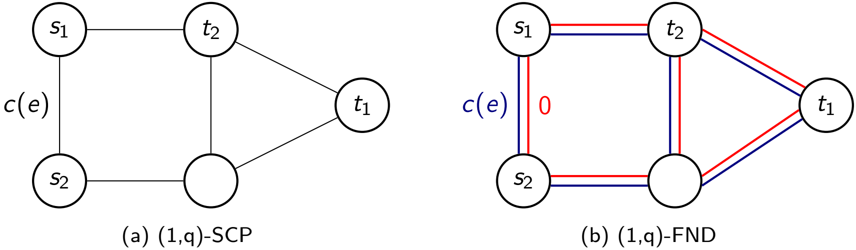


Figure: Reduction from $(1, q)$-SCP to $(1, q)$-Flexible Network Design.

Overview of our results

Overview of our results

Hardness:

- $(1, q)$-GCP is NP-hard even if $c(e) \equiv 1$.

Overview of our results

Hardness:

- $(1, q)$-GCP is NP-hard even if $c(e) \equiv 1$.
- Verifying a solution to (p, q)-stCP is NP-complete.

Overview of our results

Hardness:

- $(1, q)$-GCP is NP-hard even if $c(e) \equiv 1$.
- Verifying a solution to (p, q)-stCP is NP-complete.
- This implies that there is no α-approximation, which also holds for (p, q)-Flexible Network Design.

Overview of our results

Hardness:

- $(1, q)$-GCP is NP-hard even if $c(e) \equiv 1$.
- Verifying a solution to (p, q)-stCP is NP-complete.
- This implies that there is no α-approximation, which also holds for (p, q)-Flexible Network Design.

Exact algorithms for small values of p, q.

- $(p, 1)$-SCP:
- $(1, 2)$-SCP, $(2, 2)$-GCP.

Overview of our results

Hardness:

- $(1, q)$-GCP is NP-hard even if $c(e) \equiv 1$.
- Verifying a solution to (p, q)-stCP is NP-complete.
- This implies that there is no α-approximation, which also holds for (p, q)-Flexible Network Design.

Exact algorithms for small values of p, q.

- $(p, 1)$-SCP:
- $(1, 2)$-SCP, $(2, 2)$-GCP.

Approximation algorithms for general values of p, q.

Overview of our results

Hardness:

- $(1, q)$-GCP is NP-hard even if $c(e) \equiv 1$.
- Verifying a solution to (p, q)-stCP is NP-complete.
- This implies that there is no α-approximation, which also holds for (p, q)-Flexible Network Design.

Exact algorithms for small values of p, q.

- $(p, 1)$-SCP:
- $(1, 2)$-SCP, $(2, 2)$-GCP.

Approximation algorithms for general values of p, q.

- $\mathcal{O}(q \cdot \log p)$-approximation for (p, q)-SCP assuming p is constant.
- $\mathcal{O}(\log p \min\{\log n, p + q\})$-approximation for (p, q)-GCP.

Overview of our results

Hardness:

- $(1, q)$-GCP is NP-hard even if $c(e) \equiv 1$.
- Verifying a solution to (p, q)-stCP is NP-complete.
- This implies that there is no α-approximation, which also holds for (p, q)-Flexible Network Design.

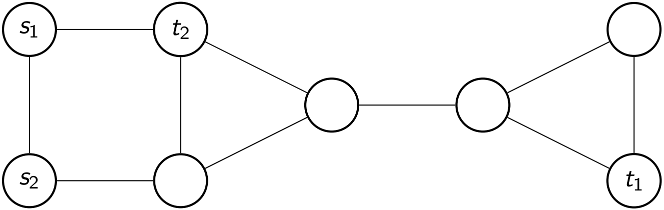
Exact algorithms for small values of p or q.

- $(p, 1)$-SCP.
- $(1, 2)$-SCP, $(2, 2)$-GCP.

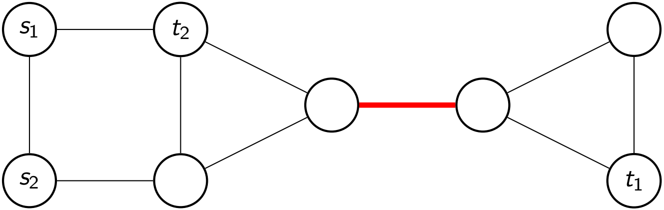
Approximation algorithms for general values of p, q.

- $\mathcal{O}(q \cdot \log p)$-approximation for (p, q)-SCP assuming p is constant.
- $\mathcal{O}(\log p \min\{\log n, p + q\})$-approximation for (p, q)-GCP.

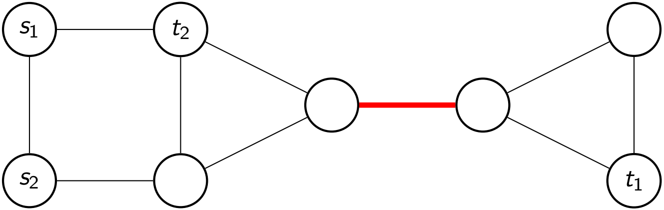
A toy example: $(1, 1)$-SCP



A toy example: $(1, 1)$-SCP

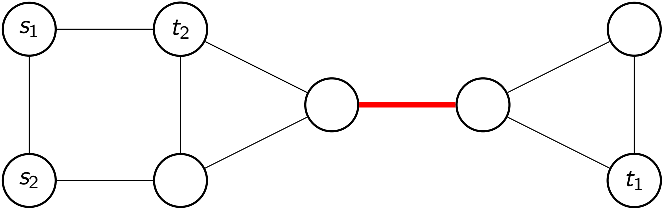


A toy example: $(1, 1)$-SCP



The attacker will only remove bridges.

A toy example: $(1,1)$-SCP



The attacker will only remove bridges.

The optimal solution is to protect all bridges that separates some (s_i, t_i)!

Generalization to $(p, 1)$-SCP

Generalization to $(p, 1)$-SCP

Algorithm for $(p, 1)$-SCP

- Consider critical cuts: p-edge-cuts which separates some (s_i, t_i).
- Protect all edges in critical cuts.

Generalization to $(p, 1)$-SCP

Algorithm for $(p, 1)$-SCP

- Consider critical cuts: p-edge-cuts which separates some (s_i, t_i).
- Protect all edges in critical cuts.

Question: Implementation in polynomial time?

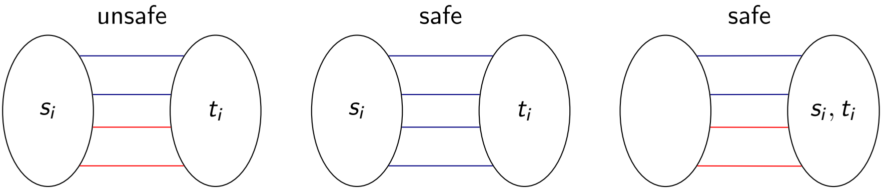
Generalization to $(p, 1)$-SCP

Algorithm for $(p, 1)$-SCP

- Consider critical cuts: p-edge-cuts which separates some (s_i, t_i).
- Protect all edges in critical cuts.

Question: Implementation in polynomial time?

- Unsafe cuts: critical but have $< p$ protected edges.
- Safe cuts: not critical or already have p protected edges.



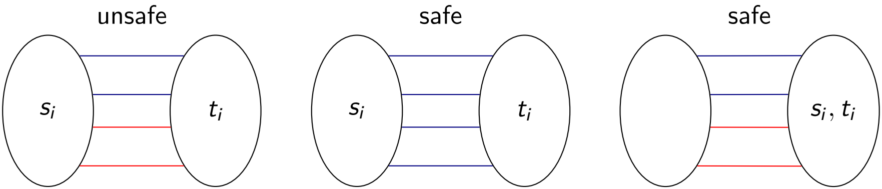
Generalization to $(p, 1)$-SCP

Algorithm for $(p, 1)$-SCP

- Consider critical cuts: p-edge-cuts which separates some (s_i, t_i).
- Protect all edges in critical cuts.

Question: Implementation in polynomial time?

- Unsafe cuts: critical but have $< p$ protected edges.
- Safe cuts: not critical or already have p protected edges.



Idea: Iteratively find an unsafe cut and protect the edges in the cut.

The capacity trick: distinguish safe and unsafe cuts

The capacity trick: distinguish safe and unsafe cuts

Idea:

- Properly define $u : E \to \mathbb{R}_{\geq 0}$ s.t. $u(\text{safe cut}) > u(\text{unsafe cut})$.

The capacity trick: distinguish safe and unsafe cuts

Idea:

- Properly define $u : E \to \mathbb{R}_{\geq 0}$ s.t. $u(\text{safe cut}) > u(\text{unsafe cut})$.
- Compute minimum-capacity (s_i, t_i) cuts to find unsafe cuts.

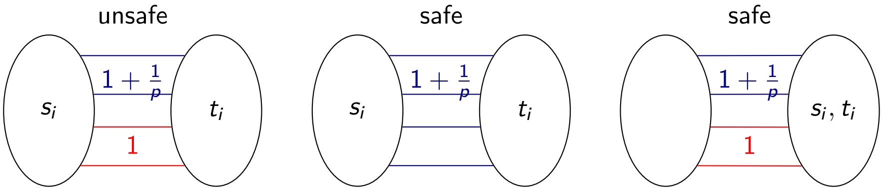
The capacity trick: distinguish safe and unsafe cuts

Idea:

- Properly define $u : E \to \mathbb{R}_{\geq 0}$ s.t. $u(\text{safe cut}) > u(\text{unsafe cut})$.
- Compute minimum-capacity (s_i, t_i) cuts to find unsafe cuts.

The capacity function:

- $u(e) = 1$ if e is unprotected.
- $u(e) = 1 + \frac{1}{p}$ if e is protected.



Overview of our results

Hardness:

- $(1, q)$-GCP is NP-hard even if $c(e) \equiv 1$.
- Verifying a solution to (p, q)-stCP is NP-complete.
- This implies that there is no α-approximation, which also holds for (p, q)-Flexible Network Design.

Exact algorithms for small values of p or q.

- $(p, 1)$-SCP.
- $(1, 2)$-SCP, $(2, 2)$-GCP.

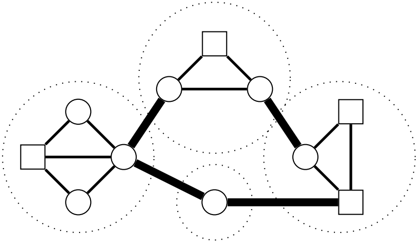
Approximation algorithms for general values of p, q.

- $\mathcal{O}(q \cdot \log p)$-approximation for (p, q)-SCP assuming p is constant.
- $\mathcal{O}(\log p \min\{\log n, p + q\})$-approximation for (p, q)-GCP.

A Divide and Conquer algorithm for $(1,2)$-SCP

A decomposition lemma

There is a polynomial-time algorithm which decompose a 2EC graph G into disjoint 2EC subgraphs $G_1, \ldots, G_k$ s.t. $G / \bigcup_{i=1}^{k} G_i$ forms a cycle.

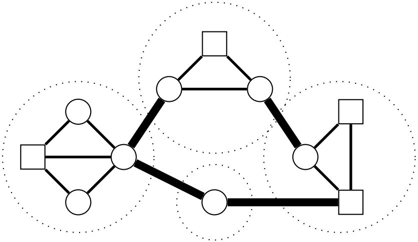


(a) Rectangles are terminals.

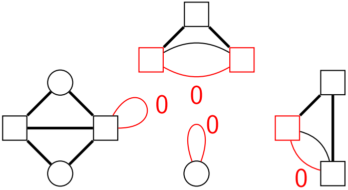
A Divide and Conquer algorithm for $(1,2)$-SCP

A decomposition lemma

There is a polynomial-time algorithm which decompose a 2EC graph G into disjoint 2EC subgraphs $G_1, \ldots, G_k$ s.t. $G / \bigcup_{i=1}^{k} G_i$ forms a cycle.



(a) Rectangles are terminals.



(b) Independent sub-instances

Overview of our results

Hardness:

- $(1, q)$-GCP is NP-hard even if $c(e) \equiv 1$.
- Verifying a solution to (p, q)-stCP is NP-complete.
- This implies that there is no α-approximation, which also holds for (p, q)-Flexible Network Design.

Exact algorithms for small values of p or q.

- $(p, 1)$-SCP.
- $(1, 2)$-SCP, $(2, 2)$-GCP.

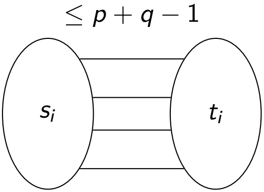
Approximation algorithms for general values of p, q.

- $\mathcal{O}(q \cdot \log p)$-approximation for (p, q)-SCP assuming p is constant.
- $\mathcal{O}(\log p \min\{\log n, p + q\})$-approximation for (p, q)-GCP.

Augmentation-based approximation algorithms

Critical cuts: $\mathcal{S} := \{S \subset V \mid |\delta(S)| \leq p + q - 1, S \text{ separates some terminal pair}\}$.

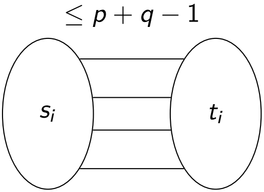
$F \subseteq E$ is feasible if and only if it contains $\geq p$ edges in each critical cut.



Augmentation-based approximation algorithms

Critical cuts: $\mathcal{S} := \{S \subset V \mid |\delta(S)| \leq p + q - 1, S \text{ separates some terminal pair}\}$.

$F \subseteq E$ is feasible if and only if it contains $\geq p$ edges in each critical cut.



$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e \\
\text{s.t.} \quad & \sum_{e \in \delta(S)} x_e \geq p \quad \forall S \in \mathcal{S} \\
& 0 \leq x_e \leq 1 \quad \forall e \in E
\end{aligned}
$$

$$
\begin{aligned}
\max \quad & \sum_{S \in \mathcal{S}} p \cdot y_S - \sum_{e \in E} z_e \\
\text{s.t.} \quad & \sum_{S : S \in \mathcal{S}, e \in \delta(S)} y_S - z_e \leq c_e \quad \forall e \in E \\
& y_S, z_e \geq 0 \quad \forall S \in \mathcal{S}, \forall e \in E
\end{aligned}
$$

Augmentation

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta(S)} x_e \geq p \quad \forall S \in \mathcal{S} \\ & 0 \leq x_e \leq 1 \quad \forall e \in E \end{aligned}$$

$$\begin{aligned} \max \quad & \sum_{S \in \mathcal{S}} p \cdot y_S - \sum_{e \in E} z_e \\ \text{s.t.} \quad & \sum_{S: S \in \mathcal{S}, e \in \delta(S)} y_S - z_e \leq c_e \quad \forall e \in E \\ & y_S, z_e \geq 0 \quad \forall S \in \mathcal{S}, \forall e \in E \end{aligned}$$

- Our primal-dual algorithm has p phases [Williamson et al. 1995].

Augmentation

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta(S)} x_e \geq p \quad \forall S \in \mathcal{S} \\ & 0 \leq x_e \leq 1 \quad \forall e \in E \end{aligned}$$

$$\begin{aligned} \max \quad & \sum_{S \in \mathcal{S}} p \cdot y_S - \sum_{e \in E} z_e \\ \text{s.t.} \quad & \sum_{S : S \in \mathcal{S}, e \in \delta(S)} y_S - z_e \leq c_e \quad \forall e \in E \\ & y_S, z_e \geq 0 \quad \forall S \in \mathcal{S}, \forall e \in E \end{aligned}$$

- Our primal-dual algorithm has p phases [Williamson et al. 1995].
- The augmentation problem: given that each critical cut contains $\geq i - 1$ protected edges, we protect more edges to ensure $\geq i$ protected edges.

Augmentation

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta(S)} x_e \geq p \quad \forall S \in \mathcal{S} \\ & 0 \leq x_e \leq 1 \quad \forall e \in E \end{aligned}$$

$$\begin{aligned} \max \quad & \sum_{S \in \mathcal{S}} p \cdot y_S - \sum_{e \in E} z_e \\ \text{s.t.} \quad & \sum_{S : S \in \mathcal{S}, e \in \delta(S)} y_S - z_e \leq c_e \quad \forall e \in E \\ & y_S, z_e \geq 0 \quad \forall S \in \mathcal{S}, \forall e \in E \end{aligned}$$

- Our primal-dual algorithm has p phases [Williamson et al. 1995].
- The augmentation problem: given that each critical cut contains $\geq i - 1$ protected edges, we protect more edges to ensure $\geq i$ protected edges.

$$\begin{aligned} \min \quad & \sum_{e \in E \setminus X_{i-1}} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta(S) \setminus X_{i-1}} x_e \geq 1 \quad \forall S \in \mathcal{S}_i \\ & x_e \geq 0 \quad \forall e \in E \setminus X_{i-1} \end{aligned}$$

$$\begin{aligned} \max \quad & \sum_{S \in \mathcal{S}_i} y_S \\ \text{s.t.} \quad & \sum_{S : S \in \mathcal{S}_i, e \in \delta(S)} y_S \leq c_e \quad \forall e \in E \setminus X_{i-1} \\ & y_S \geq 0 \quad \forall S \in \mathcal{S}_i \end{aligned}$$

Augmentation-based approximation

(informal) Dual mapping [Williamson et al. 1995]

Given a dual feasible solution $\{y_S^{(i)}\}$ of the ith phase, we can construct a dual feasible solution $\{y_S, z_e\}$ to the main LP s.t.

$$\sum_{S \in \mathcal{S}_i} y_S^{(i)} \leq \frac{1}{p-i+1}\Big(\sum_{S \in \mathcal{S}} p \cdot y_S - \sum_{e \in E} z_e\Big) \leq \frac{1}{p-i+1}\textsc{Opt}\ .$$

Augmentation-based approximation

(informal) Dual mapping [Williamson et al. 1995]

Given a dual feasible solution $\{y_S^{(i)}\}$ of the ith phase, we can construct a dual feasible solution $\{y_S, z_e\}$ to the main LP s.t.

$$\sum_{S\in\mathcal{S}_i} y_S^{(i)} \le \frac{1}{p-i+1}\Big(\sum_{S\in\mathcal{S}} p\cdot y_S - \sum_{e\in E} z_e\Big) \le \frac{1}{p-i+1}\textsc{Opt}\ .$$

(informal) Total cost of p phases

Given a K-approximation algorithm for the augmentation problem, the total cost of is

$$\sum_{i=1}^{p} cost(phase_i) \le K\sum_{S\in\mathcal{S}_i} y_S^{(i)} \le K\sum_{i=1}^{p}\frac{1}{p-i+1}\textsc{Opt} = \mathcal{O}(K\log p\cdot\textsc{Opt}).$$

Overview of our results

Hardness:

- $(1, q)$-GCP is NP-hard even if $c(e) \equiv 1$.
- Verifying a solution to (p, q)-stCP is NP-complete.
- This implies that there is no α-approximation, which also holds for (p, q)-Flexible Network Design.

Exact algorithms for small values of p or q.

- $(p, 1)$-SCP.
- $(1, 2)$-SCP, $(2, 2)$-GCP.

Approximation algorithms for general values of p, q.

- $\mathcal{O}(q \cdot \log p)$-approximation for (p, q)-SCP assuming p is constant.
- $\mathcal{O}(\log p \cdot \min\{\log n, p + q\})$-approximation for (p, q)-GCP.

Open questions

Open questions

- $(1, q)$-GCP when $q \geq 3$ is constant.

Open questions

- $(1, q)$-GCP when $q \geq 3$ is constant.

- $q = \mathit{mincut}$, i.e. find minimum-cost edge set that intersects with all minimum cuts.

Open questions

- $(1, q)$-GCP when $q \geq 3$ is constant.
- $q = mincut$, i.e. find minimum-cost edge set that intersects with all minimum cuts.
- FPT results?

Augmentation-based approximation

We solve the augmentation problems approximately.

- $(p + q - 1)$-approximation for (p, q)-SCP, assuming p is constant.
- $\mathcal{O}(\min\{\log n, p + q\})$-approximation for (p, q)-GCP.

Augmentation-based approximation

We solve the augmentation problems approximately.

- $(p + q - 1)$-approximation for (p, q)-SCP, assuming p is constant.
- $\mathcal{O}(\min\{\log n, p + q\})$-approximation for (p, q)-GCP.

Idea:

- Starting from $y = 0$, iteratively find a violating cut S and increase y_S.

Augmentation-based approximation

We solve the augmentation problems approximately.

- $(p + q - 1)$-approximation for (p, q)-SCP, assuming p is constant.
- $\mathcal{O}(\min\{\log n, p + q\})$-approximation for (p, q)-GCP.

Idea:

- Starting from $y = 0$, iteratively find a violating cut S and increase y_S.
- Use the "capacity trick" to distinguish violating and non-violating cuts.

Augmentation-based approximation

We solve the augmentation problems approximately.

- $(p+q-1)$-approximation for (p,q)-SCP, assuming p is constant.
- $\mathcal{O}(\min\{\log n, p+q\})$-approximation for (p,q)-GCP.

Idea:

- Starting from $y=0$, iteratively find a violating cut S and increase y_S.
- Use the "capacity trick" to distinguish violating and non-violating cuts.
- For (p,q)-GCP, set u s.t. $u(\text{violating cuts}) \leq 2u(\text{mincut})$.
- The number of 2-approximate mincuts is polynomial and they can be enumerated in polynomial time [Karger 1993].