Bridging the Gap Between Tree and Connectivity Augmentation

An Article by: Federica Cecchetto, Vera Traub and Rico Zenklusen

ACM SIGACT Symposium on Theory of Computing

Presented by Pooria Jalali Farahani and Mehrshad Taziki

Table of Contents

Problem Definition

- We consider the Connectivity Augmentation Problem, a classical problem in the area of Survivable Network Design.
- In this problem we are given a graph G and a set of links L. out goal is to increase the edge-connectivity of a graph by one through adding a set $F \subseteq L$ to G with minimum cardinality.
- This problem can be reduced to the Cactus augmentation probelm were the input graph is a cactus.

Theorem 1

There is a 1.393-approximation algorithm for the Connectivity Augmentation Problem.

- Before this article, the best approximation algorithm for CacAP, and therefore CAP, was a 1.91 approximation.

preliminaries

Definition (Cactus)

A cactus is a connected graph where each edge is in a unique cycle.

In order to extend the previous results on TAP into CacAP, we have to go through some definitions that makes CacAP more similar to TAP. (We assume our cactus has a arbitrarily root r.)

Definition (Terminal)

A terminal t in a cactus G is a vertex of degree 2 in G.

Definition (Ancestor)

A vertex u is said to be an ancestor of a vertex v, if and only if, every path from v to r passes through u.

preliminaries

Definition (Principal Subcactus)

Let $W \subseteq V - r$ be a connected component of $G - r$, we call $G[W \cup \{r\}]$ a principal subcactus of G.

We divide the links in L into three different groups.

1. cross-links are links with end-points in different principal sub cacti.
2. in-links are links with both end-points in a single principal sub cactus.
3. up-links are in-links from a vertex to one of its ancestors.

Definition (k-wide Instance)

A cacAP instance (G, A) is said to be k-wide if and only if every principal sub cacti of G has at most k terminals.

Modeling CacAP as a linear program

Our approach is LP-based, therefore we model the problem.

- Let $\mathcal{C}_G$ be all the min-cuts of G that doesn't contain r.
- For any link $\ell \in L$ we have a variable x_ℓ which indicates if ℓ is picked in our solution or not.
- For any min-cut $C \in \mathcal{C}_G$ at least one link covering C should be picked.
- The following LP models the problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{\ell \in L} x_\ell \\ \text{subject to} \quad & x(\delta_L(C)) \geq 1 \; \forall C \in \mathcal{C}_G \\ & x \geq 0 \end{aligned}$$

- We call this formulation the P_{Cut} formulation.

Table of Contents

2 Reduction to k-wide Instances

Residual Instance

At first, we discuss how to use a cheap covering of the heavy cuts $L_H \subset L$ to obtain an instance without any heavy cuts.

Definition (Residual Instance)

Let $\mathcal{I} = (G, L)$ be a $CacAP$ instance and let $L' \subset L$. Let $L' = \{\ell_1, \ldots, \ell_h\}$ be an ordering of the links in L'. The residual instance of $\mathcal{I}$ with respect to L and this ordering is the instance that arises by performing the following contraction operation sequentially for each link $\ell = \ell_1$ up to $\ell = \ell_h$: contract all vertices that are on every $u - v$ path in the cactus, where u and v are the endpoints of ℓ, into a single vertex.

Residual Instacne

2 Reduction to k-wide Instances

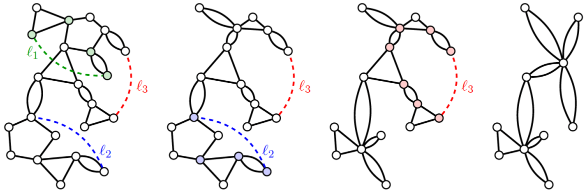


Figure: A cactus and three links ℓ_1, ℓ_2, ℓ_3 which are to be contracted are shown. In each graphic, the vertices that are going to be contracted are highlighted. Notice that the ordering of these links doesn't affect the final result.

Residual Instance

Lemma 2

Let $\mathcal{I} = (G, L)$ be a $CacAP$ instance and let $L' \subset L$. Then any order of the links in L' leads to the same residual instance of $\mathcal{I}$ with respect to L'.

Lemma 3

Let $\mathcal{I} = (G = (V, E), L)$ be a $CacAP$ instance, let $L' \subset L$, and let $\tilde{\mathcal{I}} = (\tilde{G} = (\tilde{V}, \tilde{E}), \tilde{L})$ be the residual instance of G with respect to L'. Consider the correspondence that assigns to each 2-cut $\tilde{W} \subset \tilde{V}$ of $\tilde{G}$ the set of vertices $W \subset V$ that got contracted into some vertex of $\tilde{W}$. Then this correspondence is a bijection between the 2-cuts in $\tilde{G}$ and the 2-cuts in G that are not covered by a link in L'.

Light and Heavy Cuts

2 Reduction to k-wide Instances

We fix an arbitrary vertex $r \in V$, called the root, and denote by

$$\mathcal{C}_G = \{C \subset V - \{r\} : |\delta_E(C)| = 2$$

Definition (Heavy and Light Cuts)

Let $x \in [0,1]^L$. A cut $C \in \mathcal{C}_G$ and the set $\delta_L(C)$ of links in the cut are called x-light if

$$x(\delta_L(C)) \leq \frac{16}{\epsilon}$$

Otherwise, C and $\delta_L(C)$ are called x-heavy.

Light and Heavy Cuts

Lemma 4

Let $L_H \subset L$ be a set of links such that $|\delta_{L_H}(C)| \geq 1$ for every x-heavy cut $C \in \mathcal{C}_G$. Then the residual instance with respect to L_H has no x-heavy cuts.

Now we are ready to use this lemma and the next theorem to get rid of the heavy cuts with a cheap link set and assume that all the cuts are light.

Light and Heavy cuts

Theorem 5

Let G, L be a CacAP instance and $\mathcal{H} \subset \mathcal{C}_G$, then the following LP has Integrality Gap 8 and we can compute such integral solution in polynomial time.

$$\begin{aligned} \text{minimize} \quad & \sum_{\ell \in L} x_\ell \\ \text{subject to} \quad & x(\delta_L(C)) \geq 1 \ \forall C \in \mathcal{H} \\ & x \geq 0 \end{aligned} \tag{1}$$

We prove this theorem in the next section. To obtain a cheap x-heavy cut covering, we apply the above theorem with $W = \{C \in \mathcal{C}_G : C \text{ is heavy}\}$. Then, by definition of heavy cuts, $\frac{\epsilon}{16} \cdot x$ is a feasible solution to LP (3). Thus we get an integral solution whose support $L_H \subset L$ is an x-heavy cut covering satisfying $|L_H| \leq \frac{\epsilon}{2} \cdot x(L)$, as desired.

Splitting Procedure

- To split on a 2-cut like C we contract all the vertices in C which results in the instance $\mathcal{I}_{V-C}$ and then similarly we make the instance $\mathcal{I}_C$.
- Our goal is to solve these two instances separately and then merge them to get a solution for the original instance $\mathcal{I}$.
- Assume that F_C and F_{V-C} are solutions for $\mathcal{I}_C$ and $\mathcal{I}_C$ respectively. The challenge here is that $F_C \cup F_{V-C}$ is not necessarily a solution for $\mathcal{I}$.

Splitting Procedure

2 Reduction to k-wide Instances

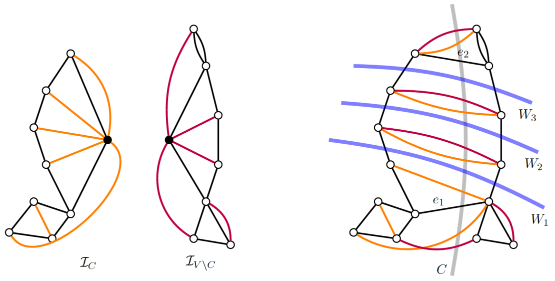


Figure: F_C is represented by orange links and F_{V-C} by red links. Blue cuts are not covered by $F_C \cup F_{V-C}$.

Structural Lemma

We now present a structural lemma about the 2-cuts that enables us to prove a result that bounds the extra links we need to add to our solutions for split instances to get a feasible solution for the original instance.

Lemma 6

Let $A, B \subset V$ be 2-cuts of G. If A and B cross, $i.e.$, $A \cap B$, $A - B$, $B - A$, and $V - (A \cup B)$ are nonempty, then the following holds:

1. $G - (\delta_E(A) \cup \delta_E(B))$ has exactly four connected components. The vertex sets of these connected components are $A \cap B$, $A - B$, $B - A$, and $V - (A \cup B)$. Each of them is a 2-cut in G.
2. $\delta_E(A)$ contains an edge in $E[B]$ and an edge in $E[V - B]$.

Structural Lemma

Proof

1. Because G is 2-edge-connected and $|\delta_E(A) \cup \delta_E(B)| \leq 4$, the graph $G(\delta_E(A) \cup \delta_E(B))$ has at most four connected components. Each of these connected components is a subset of $A \cap B$, $A - B$, $B - A$, or $V - (A \cup B)$. If A and B cross, these four sets are all nonempty, and hence $G(\delta_E(A) \cup \delta_E(B))$ has exactly four connected components which are $A \cap B$, $A - B$, $B - A$, and $V - (A \cup B)$. These sets being 2-cuts are concluded from the submodularity of the cuts and A, B, A^c, B^c being 2-cuts.
2. We use the fact that B and $V - B$ are connected hence there is an edge between B and $A \cap B$ which is in $\delta_E(A)$, and it is proven similarly for $V - B$.

$\square$

Bounding the Extra Links

We are now equipped to prove the next theorem that bounds the extra links.

Theorem 7

Given a feasible $CacAP$ instance $\mathcal{I} = (G = (V, E), L)$, a 2-cut $C \in \mathcal{C}_G$, and solutions $F_C, F_{V-C} \subset L$ to $\mathcal{I}_C$ and $\mathcal{I}_{V-C}$, respectively, one can efficiently compute a link set $F \subset L$ such that

1. $F_C \cup F_{V-C} \cup F$ is a $CacAP$ solution to $\mathcal{I}$.
2. $|F| \leq |\delta_L(C) \cap F_C| - 1$.

Bounding the Extra Links

2 Reduction to k-wide Instances

Proof

- Let $\delta_E(C) = \{e_1, e2\}$ and $W \in \mathcal{C}_G$ be a 2-cut which is not covered by any link in $F_C \cup F_{V-C}$. W crosses C hence by the second part of the structural lemma W contains exactly one of e_1 and e_2 and doesn't have the two endpoints of the other one in itself.
- Let $\mathcal{W} = \{W \subset V : |\delta_E(W)| = 2, e_1 \in E[W], \delta_L(W) \cap (F_C \cup F_{V-C}) = \emptyset\}$ Any 2-cut that is not covered by $F_C \cup F_{V-C}$ or its complement is in $\mathcal{W}$.
- $\mathcal{W}$ is a chain, hence we can write $\mathcal{W} = \{W_1, \cdots W_n\}$ with $W_1 \subsetneq W_2 \subsetneq \cdots \subsetneq W_n$, also let $W_0 = \emptyset and W_{n+1} = V$

Bounding the Extra Links

Proof(continued)

- We claim that:
  - $(W_i - W_{i-1}) \cap C$ is either empty or a 2-cut.
  - $(W_i - W_{i-1}) \cap (V - C)$ is either empty or a 2-cut.
- $(W_i - W_{i-1}) \cap C$ and $(W_i - W_{i-1}) \cap (V - C)$ are not empty.
- We now show that for every $i \in \{1, \cdots, n+1$ the set F_C contains a link in $L[W_i - W_{i-1}] \cap \delta_L(C)$.
- We conclude that $|F_C \cap \delta_L(C)| \geq n + 1 = |W| + 1$ and it completes the proof.

Splittable Instances

Definition (Small and Big Sets)

We call a set $C \subset V$ small if $|C \cap T| \leq \frac{k}{2}$. Otherwise, C is called big.

Definition (Splittable Instance)

Let $\mathcal{I} = (G = (V, E), L)$ be a $CacAP$ instance with root $r \in V$ and $x \in [0, 1]^L$. Then $\mathcal{I}$ is splittable at $C \in \mathcal{C}_G$ if C is x-light and big and $C \neq V - \{r\}$.

Definition (Unsplittable Instance)

An instance $\mathcal{I} = (G = (V, E), L)$ of $CacAP$ with root $r \in V$ is unsplittable if all 2-cuts in $\mathcal{C}_G$ are light and there is no $C \in \mathcal{C}_G$ such that $\mathcal{I}$ is splittable at C.

Algorithm for unsplittable case

Lemma 8

Every unsplittable $CacAP$ instance is k-wide.

Lemma 9

Let $v \in V - \{r\}$. Then the set of descendants of v is a 2-cut of G.

Lemma 10

For every vertex $v \in V$, there is a descendant t of v in G such that t is a terminal.

Lemma 11

Let $\mathcal{I} = (G, L)$ be a k-wide instance of the cactus augmentation problem and let $L' \subset L$. Then the residual instance of $\mathcal{I}$ with respect to L' is k-wide.

.

Algorithm for unsplittable case

2 Reduction to k-wide Instances

Now we want to use the algorithm for the k-wide case to obtain an algorithm for the unsplittable which allows cheap merging in the general case. For this reason, we prove the theorem below.

Theorem 12

Let $\epsilon' = \frac{\epsilon}{4}$. Suppose there is an α-approximation algorithm $\mathcal{A}$ for k-wide $CacAP$ instances. Then there is a polynomial-time algorithm that, given an unsplittable $CacAP$ instance $\mathcal{I} = (G = (V, E), L)$, a vector $x \in [0, 1]^L$, and a vertex s of G with $|\delta_E(s)| = 2$ and $x(\delta_L(s)) \leq \frac{\epsilon}{16}$, either returns

- A $CacAP$ solution $F \subset L$ with $|F| + |\delta_F(s)| \leq (1 + \epsilon') \cdot \alpha \cdot (x(L) + x(\delta_L(s)))$, or
- A vector $w \in R^L$ such that $w^T x < w^T x'$ for all $x' \in P_{CacAP}(\mathcal{I})$.

Algorithm for unsplittable case

2 Reduction to k-wide Instances

Proof

- By our previous lemma, this instance is k-wide. If $\delta_L(s) < 1$, we return $\chi^{\delta_L(s)}$ as our separating vector.
- Otherwise, for every $S \subseteq delta_L(s)$ with cardinality less than $\frac{1+\epsilon'}{\epsilon'} \cdot \frac{16}{\epsilon}$, we apply $\mathcal{A}$ to the residual instance of G with respect to S.
- If for a set S, the algorithm returns a solution F' such that $|F'|+2|S|\leq (1+\epsilon') \cdot \alpha \cdot (x(L) + x(\delta_L(s)))$. We return $F = F' \cup S$.
- Otherwise, define $\mu = 1 + \frac{\epsilon'(x(L)+x(\delta_L(s)))}{x(\delta_L(s))}$.
- Now, return the vector corresponding to the following constraint:

$$x'(L) + \mu \cdot x'(\delta_L(s)) > x(L) + \mu \cdot x(\delta_L(s)) \ \forall x' \in P_{CacAP}(\mathcal{I})$$

Algorithm for unsplittable case

2 Reduction to k-wide Instances

Proof (Continue)

Suppose this doesn't hold. Therefore, there exists a solution F^* for $\mathcal{I}$ that:

$$|F^*| + \mu \cdot \delta_{F^*}(s) \leq x(L) + \mu \cdot x(\delta_L(s)) = (1+\epsilon') \cdot (x(L) + x(\delta_L(s)))$$

Set $S = \delta_{F^*}(s)$. This follows:

$$|S| \leq \frac{1+\epsilon'}{\epsilon'} x(\delta_L(s)) \leq \frac{1+\epsilon'}{\epsilon'} \cdot \frac{16}{\epsilon}$$

So, S was considered during our procedure.

Algorithm for unsplittable case

Proof (Continue)

Let F' be the output of our algorithm on the residual instance of G with respect to S. Now, $|F'| \leq \alpha|F^* - S|$ because $F^* - S$ is a feasible solution for this residual instance.

Now we have:

$$|F'| + 2|S| \leq \alpha|F^* - S| + 2|S| \leq \alpha(|F^*| + |S|) \leq \alpha(|F^*| + \mu|\delta_{F^*}(s)|)$$

$$\alpha(|F^*| + \mu|\delta_{F^*}(s)|) \leq (1 + \epsilon') \cdot \alpha \cdot (x(L) + x(\delta_L(s)))$$

But this contradicts the fact that we didn't return this choice of S.

□

Round Or Cut Method

Theorem 13

Suppose there is an α-approximation algorithm $\mathcal{A}$ for k-wide $CacAP$ instances. Then for any $CacAP$ instance $\mathcal{I} = (G = (V, E), L)$ and, a vector $x \in [0, 1]^L$ such that no cut is x-heavy, there is a polynomial time algorithms that returns one of the followings:

- A $CacAP$ solution F such that $|F| \leq \alpha \cdot (1 + \frac{\epsilon}{2}) \cdot x(L)$
- A vector $w \in R^L$ such that $w^T x < w^T x'$ for all $x' \in P_{CacAP}(\mathcal{I})$.

The only remaining thing to conclude our reduction is an algorithm that covers the x-heavy with a relatively small cost.

Round Or Cut Method

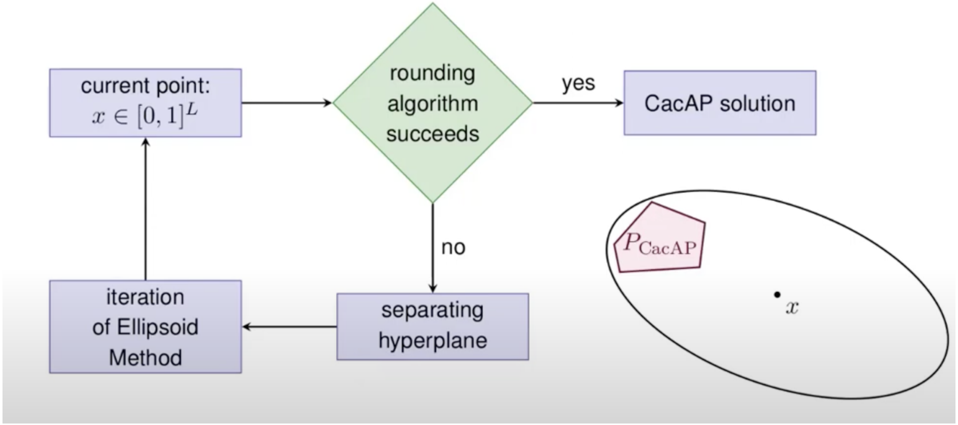


Figure: An Illustration for the Round-or-Cut procedure.

Table of Contents

3 Bounding by Splitting

Solving an Instance Without Cross-Links

Theorem [BFG+14]

Weighted CacAP can be solved in time $3^{|T|}\text{poly}(n)$ where T is the set of terminals.

- Using this Theorem, we can split each cross-link $\{u, v\}$ into two uplinks $\{u, r\}$ and $\{r, v\}$.
- Now we can solve each principal subcacti efficiently using Theorem **[BFG+14]** to find the optimal answer F.
- Since each cross-link is repeated two times, $c(F) \leq c^T x + \sum_{\ell \in L_{\text{cross}}} x_\ell c_\ell$
- This concludes one of our backbone procedures.

Table of Contents

4 Covering Heavy Cuts

In this section we will prove the following theorem.

Theorem 5

Let G, L be a CacAP instance and $\mathcal{H} \subset \mathcal{C}_G$, then the following LP has Integrality Gap 8 and we can compute such integral solution in polynomial time.

$$\begin{array}{ll} \text{minimize} & \sum_{\ell \in L} x_\ell \\ \text{subject to} & x(\delta_L(C)) \geq 1 \ \forall C \in \mathcal{H} \\ & x \geq 0 \end{array}$$

Firstly, we solve the problem for the case where G is a cycle. We will use a lemma proved in **[DKL76]** to reduce the general case to this case.

Solving the Problem When G Is a Cycle

Let the vertices of G be $v_0, v_1, \ldots, v_n$ and imagine G is rooted from $r = v_0$.

Now a min-cut $C \in \mathcal{C}_G$ is in the form of an interval $v_a, \ldots v_b$ with $a \leq b$.
We can divide the links $\ell = (u, v)$ covering C into two groups:

1. A link with one endpoint in $(v_a, \ldots, v_b)$ and one end point in $(v_{b+1}, \ldots, v_n)$.
2. A link with one endpoint in $(v_0, \ldots, v_{a-1})$ and one end point in $(v_a, \ldots, v_b)$.

In order to cover every min-cut in $\mathcal{H}$, for every $\{v_a, \ldots v_b\}$, at list one of the mentioned links has to be picked.

Modeling the Problem as a Rectangle Hitting Problem

Based on the previous intuition we can model the problem as the following rectangle hitting problem.

Definition (Rechtangle Hitting Problem)

- For each min-cut, $C = \{v_a, \dots v_b\}$, we have two rectangles, $R_C^{\rightarrow}$ and $R_C^{\uparrow}$. were $R_C^{\uparrow} = [a, b] \times [b+1, n]$ and $R_C^{\rightarrow} = [0, a-1] \times [a, b]$.
- For each link, $\ell = (u, v) \in L$ with $x_\ell > 0$, we have a point with Coordinates (u, v).
- Our goal is to pick the least amount of points such that for every $C \in \mathcal{H}$, at least one of the rectangles $R_C^{\rightarrow}$ or $R_C^{\uparrow}$ is covered by a point.

Modeling the Problem as a Rectangle Hitting Problem

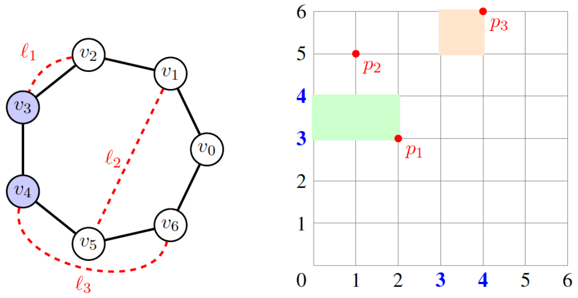


Figure: The rectangle hitting problem for the cycle C_7 and min-cut $\{v_3, v_4\}$. Note that each point p_i corresponds to a link ℓ_i.

Modeling the Problem as a Rectangle Hitting Problem

Since every min cut is covered by x, and LP feasible answer, therefore for every $C \in \mathcal{H}$ we have:

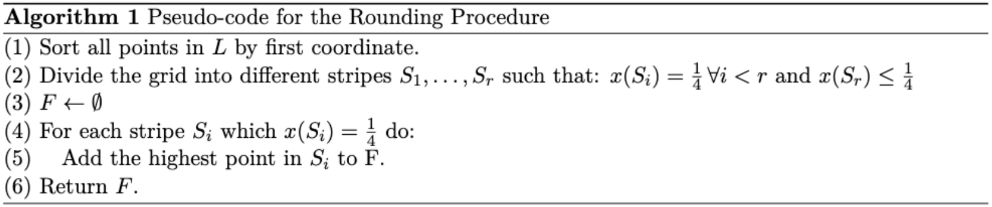
$$\sum_{\ell \in L, p_\ell \in R_C^{\uparrow}} x_\ell + \sum_{\ell \in L, p_\ell \in R_C^{\rightarrow}} x_\ell \geq 1 \rightarrow \sum_{\ell \in L, p_\ell \in R_C^{\uparrow}} x_\ell \geq \frac{1}{2} \text{ or, } \sum_{\ell \in L, p_\ell \in R_C^{\rightarrow}} x_\ell \geq \frac{1}{2}$$

So, we can partition the cuts in $\mathcal{H}$ into two parts, $\mathcal{H}^{\rightarrow}$ and $\mathcal{H}^{\uparrow}$.

Rounding the LP Solution

4 Covering Heavy Cuts

- Now, we propose a rounding procedure that covers every rectangle in $\mathcal{H}^{\rightarrow}$ (or $\mathcal{H}^{\uparrow}$) with cost bounded by $4x(L)$.
- Merging these two solutions gives us a set of points with cost at most $8x(L)$ that covers every cut in $\mathcal{H}$.



Algorithm 1 Pseudo-code for the Rounding Procedure

```
(1) Sort all points in $L$ by first coordinate.
(2) Divide the grid into different stripes $S_1, \ldots, S_r$ such that: $x(S_i) = \frac{1}{4}\ \forall i < r$ and $x(S_r) \leq \frac{1}{4}$
(3) $F \leftarrow \emptyset$
(4) For each stripe $S_i$ which $x(S_i) = \frac{1}{4}$ do:
(5)     Add the highest point in $S_i$ to F.
(6) Return $F$.
```

Analysis of the Rounding Procedure

- Now, we prove the approximation factor of this procedure.

Proof

Imagine a rectangle $R_C^{\uparrow} \in \mathcal{H}^{\uparrow}$.
Let S_a, S_b be the first and last stripes that $R_C^{\uparrow}$ intersect with.
Now, we divide the proof into two cases:

1. $R_C^{\uparrow}$ contains all the points in S_a: In this case, obviously $R_C^{\uparrow}$ also contains the highest point in S_a, therefore, it's covered.
2. $R_C^{\uparrow}$ doesn't contain all the points in S_a: Then, $x(R_C^{\uparrow} \cap S_a) < \frac{1}{4}$. since $x(R_C^{\uparrow}) \geq \frac{1}{2}$, there must be and stripe S_i, $a < i < b$, such that $R_C^{\uparrow}$ at least contains a point in S_i. Since the highest point in S_i can't be above $R_C^{\uparrow}$, it should be inside of it.

$\square$

Table of Contents

5 Rounding Using Chvatal-Gomory Cuts

Goal

5 Rounding Using Chvatal-Gomory Cuts

In this section we prove the following lemma.

Lemma 14

For a k-wide CacAP Instance G, L with a non negative cost function, there exists a polynomial time solvable polytope $P_{\text{Cross}} \supset P_{\text{Cut}}^{\text{Integral}}$ such that any $x \in P_{\text{Cross}}$ can be rounded to a feasible solution F where:

- $c(F) \leq c^T x + \sum_{\ell \in L_{\text{in}} - L_{\text{up}}} x_\ell c_\ell$

- We do this by adding a sufficient amount of $\{0, \frac{1}{2}\}$-Chvatal-Gomory cuts to P_{Cut}.

Chvatel-Gomory Cuts

Definition (Chvatel-Gomory Cut)

A Chvatel-Gomory Cut for a polytope P of the form $Ax \geq b$ is any constraint of type:

$$\lambda^T Ax \geq \lceil \lambda^T b \rceil$$

Where $\lambda \in \{0, \frac{1}{2}\}^n$ and $\lambda^T A$ is integral.

Theorem[FGKS18]

For an instance of TAP, the polytope $P_{\text{Cut}}^{\text{CG}}$, which is obtained by adding all of the Chvatel-Gomory Cuts to the P_{Cut} polytope, is polynomially solvable. Furthermore, $x \in P_{\text{Cut}}^{\text{CG}}$ can be rounded to a solution F which $c(F) \leq c^T x + \sum_{\ell \in L_{\text{in}} - L_{\text{up}}} x_\ell c_\ell$

A Simple 2-Approximation using Bidirecting

5 Rounding Using Chvatal-Gomory Cuts

- A simple two approximation for TAP works by splitting each link into two up-links.
- As mentioned, this idea doesn't work in the CacAP's case.
- Instead of splitting a link $\{u, v\}$, we replace it with two directed links (u, v) and (v, u).
- We say a cut C is covered only if a links that enters C is picked.

Lemma 15

The following LP models this problem. Moreover, it's Integral for any weighted CacAP instance.

$$\begin{aligned} \text{minimize} \quad & \sum_{\ell \in \overrightarrow{L}} x_\ell c_\ell \\ \text{subject to} \quad & x(\delta^-_{\overrightarrow{L}}(C)) \geq 1 \ \forall C \in \mathcal{C}_G \\ & x \geq 0 \end{aligned} \tag{2}$$

Obtaining the Laminar Family $\mathcal{L}$

We obtain the laminar family from the dual of LP (3).

$$\begin{aligned} \text{maximize} \quad & \sum_{C \in \mathcal{C}_G} y_C \\ \text{subject to} \quad & \sum_{\substack{C \in \mathcal{C}_G : \\ \ell \in \delta^-_{\overrightarrow{L}}(C)}} y_C \leq c_\ell \ \forall \ell \in \overrightarrow{L} \\ & y \geq 0 \end{aligned} \tag{3}$$

Lemma 16

We can compute in polynomial time an optimum solution y^* of the dual LP (4) such that y^* is minimal and has laminar support.

Obtaining the Laminar Family $\mathcal{L}$

5 Rounding Using Chvatal-Gomory Cuts

Lemma 17

For $R \subseteq L_{\text{Cross}}$, we can efficiently compute a set $F \subseteq L$ such that:

- $R \cup F$ is a feasible CacAP answer.
- $c(F) \leq \sum_{\substack{C \in \mathcal{C}_G: \\ \delta_L(C) \cap R = \emptyset}} y^*_C$, where y^* is the minimal dual solution with laminar support.

Proof

- Let y be the restriction of y^* to the min-cuts not covered by R. y is the optimal answer of the LP (4) for G', the residual instance of G with respect to R.
- By the strong duality theorem this is the optimum of LP (3) for G'. Because LP (3) is integral, we can get the desired set of links F by solving LP (3) for G'.

$\square$

Saving on cross-links via Chvatal-Gomory cuts

- Let $\mathcal{L}$ be the minimal laminar family in precious section. We define $P^{\mathcal{L}}_{\text{Cut}}$ as the following:

$$P^{\mathcal{L}}_{\text{Cut}} := \{x \geq 0 | x(\delta_L(C)) \geq 1 \, \forall C \in \mathcal{L}\}$$

- Now, get $P^{\mathcal{L}}_{\text{CG}}$ by adding all of the Chvatal-Gomory cuts of $P^{\mathcal{L}}_{\text{Cut}}$ to it.
- We define $P_{\text{Cross}} := P_{\text{Cut}} \cap P^{\mathcal{L}}_{\text{CG}}$.
- Now, we show that P_{Cross} fulfills the properties we wanted.

Proof

We can efficiently separate over P_{Cut} since it has polynomial number of constraints. Also, $\mathcal{L}$ was a laminar family, we can represent it using a tree T. Now, $P^{\mathcal{L}}_{\text{CG}}$ is all the Chvatal-Gomory cuts of a TAP on T. using the work of **[FGKS18]**, we can also separate on $P^{\mathcal{L}}_{\text{CG}}$. This completes a separation oracle for P_{Cross}.

Saving on cross-links via Chvatal-Gomory cuts

5 Rounding Using Chvatal-Gomory Cuts

Proof[Continue]

Let $x \in P_{\text{Cross}}$, Since x covers all the cuts in $\mathcal{C}_G$, therefore, it covers all cuts in $\mathcal{L}$. So it fulfills all the constraints of $P^{\text{CG}}_{\text{Cut}}$ for the TAP instance over the tree that models $\mathcal{L}$.
Now, if we use the results of [FGKS18], this TAP solution can be rounded to a solution $F \subseteq L_{\text{Cross}} \cup \overrightarrow{L_{\text{in}}}$ such that $c(F) \leq c^T x + \sum_{\ell \in L_{\text{in}} - L_{\text{up}}} x_\ell c_\ell$.
However, this solution only covers the cuts in $\mathcal{L}$ and is not necessarily feasible for our CacAP instance.
We apply the previous Lemma when $R = F_{\text{Cross}}$ to obtain a set $G \subseteq L$ stisfying the following property:

1. $F_{\text{Cross}} \cup H$ is a CacAP solution.
2. $c(H) \leq \sum\limits_{C \in \mathcal{C}_G : \delta_L(C) \cap F_{\text{Cross}} = \emptyset} y^*_C$ where y^* is the optimal minimal answer of LP (4).

Saving on cross-links via Chvatal-Gomory cuts

Proof[Continue]

Since y^* is zero for min cuts out of $\mathcal{L}$, we can only consider the min-cuts in $\mathcal{L}$.
Every cut in $\mathcal{L}$ is covered by F, a cut not covered by F_{Cross}, is covered by F_{in} therefore:

$$c(H) \leq \sum_{\substack{C \in \mathcal{C}_G: \\ \delta_L(C) \cap F_{\text{Cross}} = \emptyset}} y^*_C \leq \sum_{\ell \in F_{\text{in}}} \sum_{\substack{C \in \mathcal{C}_G: \\ \ell \in \delta^-_L(C)}} y^*_C \leq \sum_{\ell \in F_{\text{in}}} c_\ell = c(F_{\text{in}})$$

This bounds the cost of solution $F_{\text{Cross}} \cup H$ by the following:

$$c(F_{\text{Cross}} \cup H) \leq c(F_{\text{Cross}}) + c(H) \leq c(F_{\text{Cross}}) + c(F_{\text{in}}) = c(F) \leq c^T x + \sum_{\ell \in L_{\text{in}} - L_{\text{up}}} x_\ell c_\ell$$

□

Table of Contents

6 Going Below 1.5 Using Stack Analysis

In this section, we try to strengthen our procedures to get a better that 1.5-approximation factor for solving k-wide instances.

- Name the principal sub cacti of G by $G_1, \dots, G_q$ and let L_i be the links that have at least one endpoint in G_i.
- When we solve each instance of (G_i, L_i) independently to get a solution F_i that covers all min-cuts in G_i, the solution $\bigcup_{i=1}^{q} F_i$ can have many redundant links.
- Our main goal is to delete these links to improve our backbone procedures.

L_{cross}-minimal solutions

Definition (Minimal Link)

We say that a link ℓ_1 is minimal with respect to ℓ_2 if for any shadow $\hat{\ell_1}$ of ℓ_1 , the 2-cuts covered by $\{\hat{\ell_1}, \ell_2\}$ are a strict subset of those covered by $\{\ell_1, \ell_2\}$.

Definition (L_{cross}-minimal Instance)

We say a set $F \subseteq L$ is L_{cross}-minimal if for any two distinct links $\ell_1 \in L_{\text{cross}} \cap F$ and $\ell_2 \in F$, ℓ_1 is minimal with respect to ℓ_2.

- It's easy to see that in an L_{cross}-minimal solution for G_i, no two cross-links have ancestry relationship.

Bundle Constraints

Now that we know the definition of L_{cross}-Minimal solutions, for any $i \in [q]$, we define the polytope P_i^{min} as the following:

$$P_i^{\text{min}} = \text{Convex-Hull}(\{\chi^F | F \subseteq L_i \text{ is an } L_{\text{cross}}\text{-minimal solution for } G_i.\})$$

Theorem 18

Let G be a k-wide instance of CacAP and k be a constant. Then, we can efficiently optimize a linear objective function on the polytope P_i^{min}. Equivalently, we can also separate on this polytope.

Bundle Constraints

Now we are ready to formulate the relaxation that we will use. We define

$$P^{\text{min}}_{\text{bundle}} = \{x \in [0,1]^L \, : \, x|_{L_i} \in P^{\text{min}}_i \, \forall i \in [q]\}$$

- In short words, we are forcing the restriction of our solution to every principal sub cacti to be a convex combination of L_{cross}-minimal Solutions.
- We can efficiently optimize a linear objective on $P^{\text{min}}_{\text{bundle}}$ because for all $i \in [q]$, the polytope P^{min}_i is separable.

The Rounding Algorithm

Lemma 19

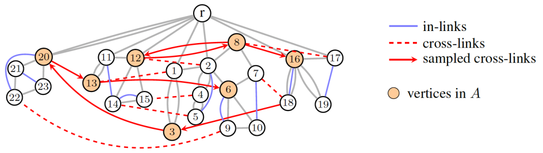
Given sets F_i^{cross} of cross-links for all $i \in [q]$, we can compute in polynomial time a minimum cardinality set F of cross-links such that for all $i \in [q]$ the set F covers every min-cut $C \in \mathcal{C}_{G_i}$ that is covered by F_i^{cross}.

Proof

- Define A as the set of all endpoints of links in $\bigcup_{i=1}^{q} F_i^{\text{cross}}$.
- We are looking for a minimum-cardinality set F of cross-links that cover all 2-cuts $C \in \mathcal{C}_G$ which $A \cap C \neq \emptyset$.
- A set F of cross-links has this property if and only if for every $a \in A$ there is a descendant of a that is an endpoint of a link $\ell \in F$.
- This problem can be modeled and solved using the edge-cover problem.

Reduction to Edge Cover Problem

6 Going Below 1.5 Using Stack Analysis



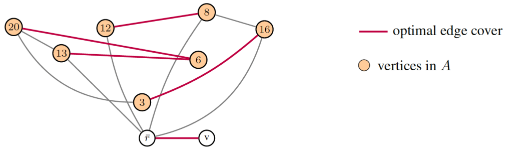
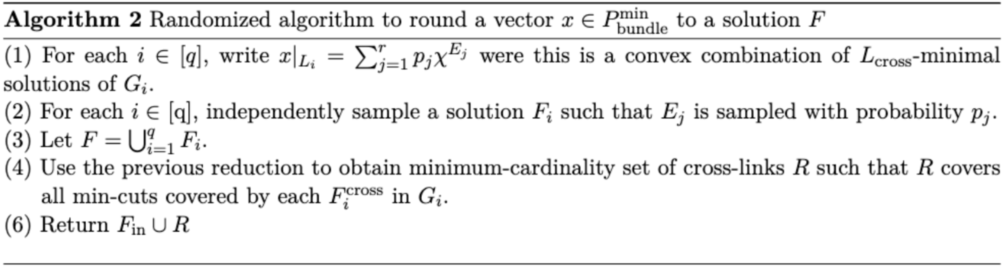


Figure: The top picture shows a possible solution obtained by taking the union of one solution for each principal subcactus. The heads of the cross-links highlight from which principal subcactus a cross-link has been sampled. Cross-links not appearing in the solution are dashed.

Rounding a vector in $P_{\text{bundle}}^{\text{min}}$



Algorithm 2 Randomized algorithm to round a vector $x \in P_{\text{bundle}}^{\text{min}}$ to a solution F

```
(1) For each $i \in [q]$, write $x|_{L_i} = \sum_{j=1}^{r} p_j \chi^{E_j}$ were this is a convex combination of $L_{\text{cross}}$-minimal solutions of $G_i$.
(2) For each $i \in$ [q], independently sample a solution $F_i$ such that $E_j$ is sampled with probability $p_j$.
(3) Let $F = \bigcup_{i=1}^{q} F_i$.
(4) Use the previous reduction to obtain minimum-cardinality set of cross-links $R$ such that $R$ covers
    all min-cuts covered by each $F_i^{\text{cross}}$ in $G_i$.
(6) Return $F_{\text{in}} \cup R$
```

- Since the output of algorithms 2 is a L_{cross}-minimal solution restricted to every principal sub cactus, this output is a feasible solution.
- In this rounding procedure the solution of every principal subcacti is sampled independently, therefore, many cross-links sampled in this solution are redundant.

Stack Analysis

- To simplify our analysis, we assign each vertex u to one of its descendant terminal t_u.
- For each terminal t, create a stack S_t. We assign every cross links $\ell = \{u, v\}$ to two stacks, S_{t_u} and S_{t_v}.
- Imagine $t_{u_1} = t_{u_2} = t$ were u_i is an endpoint of ℓ_i. The link ℓ_1 is on top of ℓ_2 in the stack S_t, if and only if, u_1 is an ancestor of u_2. We show this by $\ell_2 \preceq_t \ell_1$.
- Since each F_i was L_{cross}-minimal, it contains at most one vertex from each stack S_t.

Definition (Dominated Link)

Let $i, j \in$ [q] with $i \neq j$, and let $\ell_i \in F_i$ and $\ell_j \in F_j$. We say that ℓ_i dominates ℓ_j if there is a terminal t such that $\ell_i, \ell_j \in S_t$ and $\ell_i \preceq_t \ell_j$.
A dominated link l_i can be removed from a solution F.

Assigning Links to Stacks

6 Going Below 1.5 Using Stack Analysis

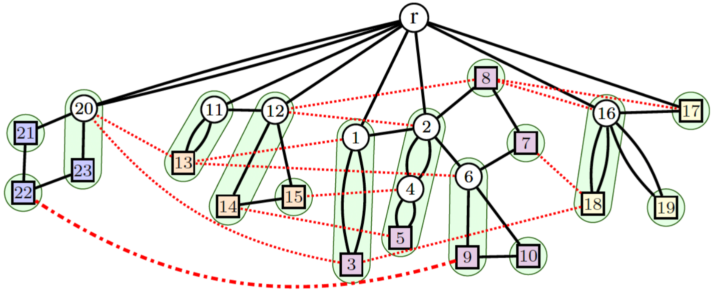


Figure: An instance in which we partition the set of cross-links into stacks

Bounding the Removable Links

Definition (Removable Set)

We call a set $R \subseteq F$ removable if and only if $F - R$ is still a feasible answer for our CacAP instance.

Lemma 20

Let D be the set of all dominated links in F_{cross}. And let R be the maximum cardinality removable set of F_{cross}. Then we have:

$$|R| \geq \frac{|D|}{3}$$

Now, we only need to bound the size of D to calculate the expected approximation factor of our algorithm.

Bounding the Removable Links

Lemma 21

$$\mathbb{E}[|D|] \geq \sum_{t \in T} f(x(S_t))$$

Where $f(x) := x + e^{-x} - 1.$

Proof

$$\mathbb{E}[|D|] \geq \sum_{t \in T} \mathbb{P}[\text{A link sampled at } S_T \text{ is dominated}]$$

$$\mathbb{P}[\text{A link sampled at } S_T \text{ is dominated}] = \sum_{\ell \in S_T} x_\ell \Big(1 - \prod_{\hat{\ell} \in B(\ell)} (1 - x_{\hat{\ell}})\Big) \geq \sum_{\ell \in S_T} x_\ell e^{-x(B(\ell))}$$

Bounding the Removable Links

Proof[Continue]

$$\sum_{\ell \in S_T} x_\ell e^{-x(B(\ell))} \geq \int_0^{x(S_t)} 1 - e^{-z} dz = f(x(S_t))$$

$$\mathbb{E}[|D|] \geq \sum_{t \in T} \mathbb{P}[\text{A link sampled at } S_T \text{ is dominated}] \geq \sum_{t \in T} f(x(S_t))$$

$\square$

This means our algorithm at least removes $\frac{\sum_{t \in T} f(x(S_t))}{3}$ edges in expectation.

Analysing the Algorithm

6 Going Below 1.5 Using Stack Analysis

- Define $\alpha := \frac{x(L_{\text{cross}})}{x(L)}$.
- Every cross-link in contained in two different stacks, then, $\sum_{t\in T} x(S_t) = 2\alpha x(L)$.
- The rounding procedure using Chvatel-Gomory cuts gives a $2-\alpha$ approximation factor.
- Our new rounding approach at most gives us

$$\left(1+\alpha-\frac{\sum_{t\in T} f(x(S_t))}{3x(L)}\right)x(L)$$

many links in expectation.

Analysing the Algorithm

Since the algorithm returns the best of these solutions, its factor is at most

$$\max\{\min\{2-\alpha, 1+\alpha-\frac{2\alpha}{3\sum_{t\in T} y_t}\sum_{t\in T} f(x(y_t))\} : 0 \le y_t \le 1, \sum_{t\in T} y_t \ge \alpha|T|, 0 \le \alpha \le 1\}$$

It can be shown that this optimization problem has value is strictly less than 1.459.

Table of Contents

7 References

References

7 References

[CTZ22] F. Cecchetto, V. Traub, and R. Zenklusen. Bridging the gap between tree and connectivity augmentation: Unified and stronger approaches. In Proceedings of 53rd Annual ACM Symposium on Theory of Computing (STOC), pages 370–383, 2021.

[BFG+14] M. Basavaraju, F. V. Fomin, P. Golovach, P. Misra, M. S. Ramanujan, and S. Saurabh. Parameterized algorithms to preserve connectivity. In Proceedings of 41st International Colloquium on Automata, Languages, and Programming (ICALP), 2014.

[Adj18] D. Adjiashvili. Beating approximation factor two for weighted tree augmentation with bounded costs. ACM Transactions on Algorithms, 15(2):19:1–19:26, 2018.

[FGKS18] S. Fiorini, M. Gross, J. Konemann, and L. Sanita. Approximating weighted tree augmentation via Chvatal-Gomory Cuts. In Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2018.

[DKL76] E. A. Dinitz, A. V. Karzanov, and M. V. Lomonosov. On the structure of the system of minimum edge cuts of a graph. Studies in Discrete Optimization, 1976.

Bridging the Gap Between Tree and Connectivity Augmentation

Thank you for listening!
Any questions?