Some Progress in DST on Planar Graphs

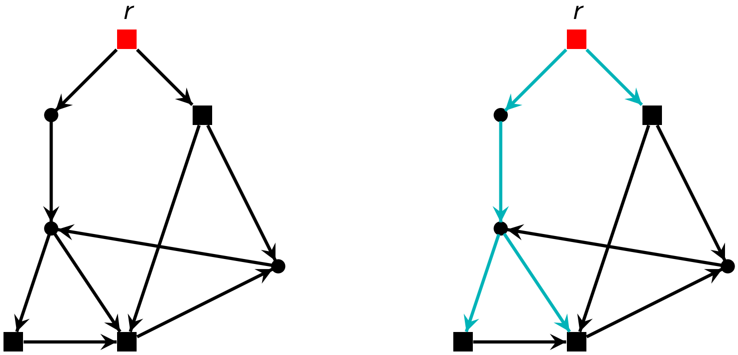
Zachary Friggstad   Ramin Mousavi

University of Alberta

Banff 2023

Directed Steiner Tree (DST) problem

Input: directed graph $G = (V, E)$, a root node $r \in V$, non-negative edge costs $c_e \geq 0$ for all $e \in E$, and a set of terminal nodes $X \subseteq V \setminus \{r\}$.
Output: minimum cost branching F rooted at r s.t. every terminal is reachable from r using F.



Let $n := |V|$ and $k := |X|$. Non-terminal nodes = Steiner nodes

Motivation

Planar DST: DST instance where the input graph is planar.

- DST is a generalization of (undirected) Steiner tree, group Steiner tree, and set cover.
- (Undirected) Steiner tree:
  - ≈ 1.39-approx in general graphs Byrka et al. 2010.
  - PTAS for planar instances Borradaile et al. 2009.
  - ≈ 1.22-approx for quasi-bipartite instances Goemans et al. 2012.
- DST is less understood:
  - No $O(\log^{2-\epsilon} n)$-approx for $\epsilon > 0$ Halperin and Krauthgamer 2003.
  - Best upper bound $O(k^{\epsilon})$ for any constant $\epsilon > 0$ Charikar et al. 1997.
  - $O(\frac{\log^2 k}{\log \log k})$-approx in quasi-polynomial time. This is tight! Grandoni et al. 2019.
  - $O(\log k)$-approx for quasi-bipartite DST and this is tight too! Hibi and Fujito 2012.

Motivation

Planar DST: DST instance where the input graph is planar.

- DST is a generalization of (undirected) Steiner tree, group Steiner tree, and set cover.
- (Undirected) Steiner tree:
  - ≈ 1.39-approx in general graphs Byrka et al. 2010.
  - PTAS for planar instances Borradaile et al. 2009.
  - ≈ 1.22-approx for quasi-bipartite instances Goemans et al. 2012.
- DST is less understood:
  - No $O(\log^{2-\epsilon} n)$-approx for $\epsilon > 0$ Halperin and Krauthgamer 2003.
  - Best upper bound $O(k^{\epsilon})$ for any constant $\epsilon > 0$ Charikar et al. 1997.
  - $O(\frac{\log^2 k}{\log \log k})$-approx in quasi-polynomial time. This is tight! Grandoni et al. 2019.
  - $O(\log k)$-approx for quasi-bipartite DST and this is tight too! Hibi and Fujito 2012.

In what settings (e.g., what family of graphs) DST is easier to approximate than in general graphs?

Our results

Theorem (Friggstad-M. 2023)

There is an $O(\log k)$-approximation for planar DST.

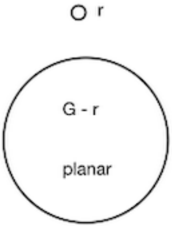
Quasi-bipartite DST: NO edge between any two Steiner nodes.

Theorem (Friggstad-M. 2023)

There is a 20-approximation for quasi-bipartite DST on planar graphs.

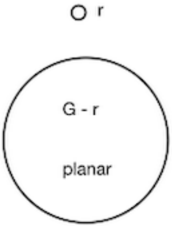
- Also we bound the integrality gap of the natural cut-based LP.
- It is extendable to graphs excluding a fixed minor.

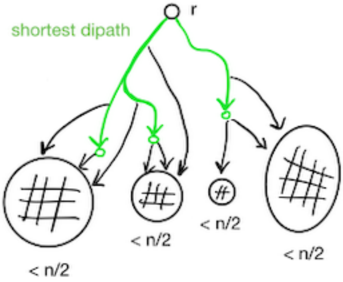
Thorup's balance separator – 2001



- There exists a separator consists of 3 shortest path from r.

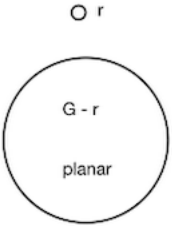
Thorup's balance separator – 2001

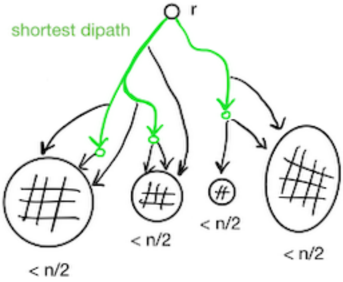




- There exists a separator consists of 3 shortest path from r.

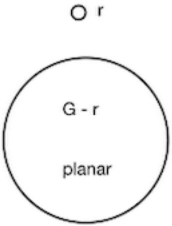
Thorup's balance separator – 2001

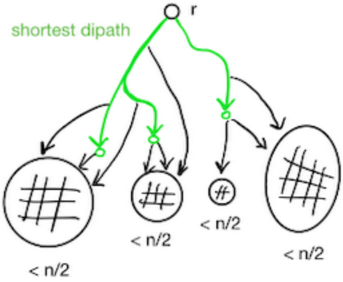




- There exists a separator consists of 3 shortest path from r.
- The cost of this balanced separator is at most $3 \times$ the furthest distance from r.

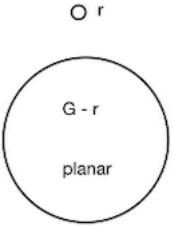
Thorup's balance separator – 2001

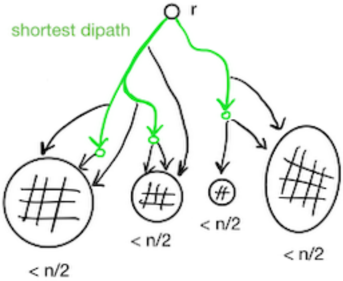




- There exists a separator consists of 3 shortest path from r.
- The cost of this balanced separator is at most 3 $\times$ the furthest distance from r.
- Works with weighted vertices. E.g., we could make sure every weakly connected component has at most $\frac{k}{2}$ terminals.

Thorup's balance separator – 2001





- There exists a separator consists of 3 shortest path from r.
- The cost of this balanced separator is at most $3 \times$ the furthest distance from r.
- Works with weighted vertices. E.g., we could make sure every weakly connected component has at most $\frac{k}{2}$ terminals.
- Similar type separator is used in undirected k-MST and Steiner tree in planar graphs Cohen-Addad 2022.

A slow algorithm

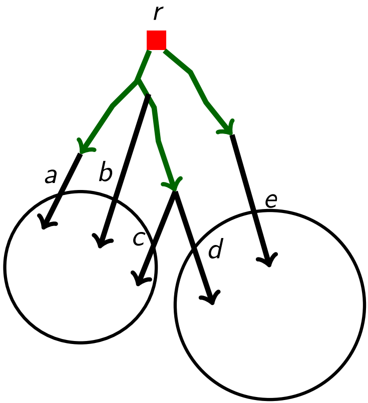
1. Guess opt value ($\mathrm{poly}(n)$ many choices). Remove all Steiner nodes with distance more than opt from r.

A slow algorithm

1. Guess opt value ($\mathrm{poly}(n)$ many choices). Remove all Steiner nodes with distance more than opt from r.
2. Compute a balanced shortest path separator rooted at r.

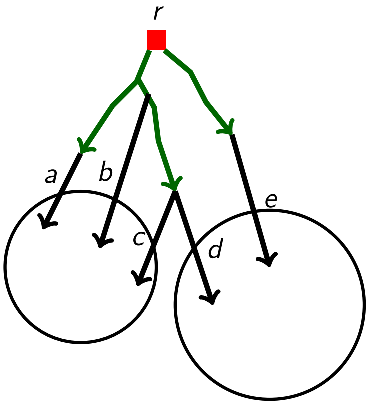
A slow algorithm

1. Guess opt value ($\mathrm{poly}(n)$ many choices). Remove all Steiner nodes with distance more than opt from r.
2. Compute a balanced shortest path separator rooted at r.



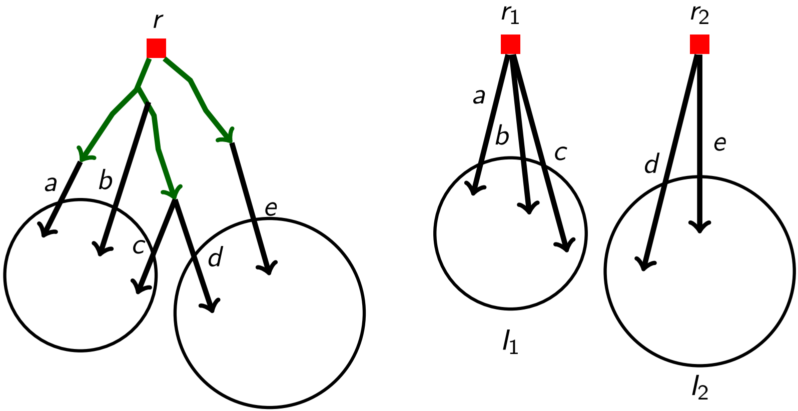
A slow algorithm

1. Guess opt value ($\mathrm{poly}(n)$ many choices). Remove all Steiner nodes with distance more than opt from r.
2. Compute a balanced shortest path separator rooted at r.
3. Compute the smaller subinstances.



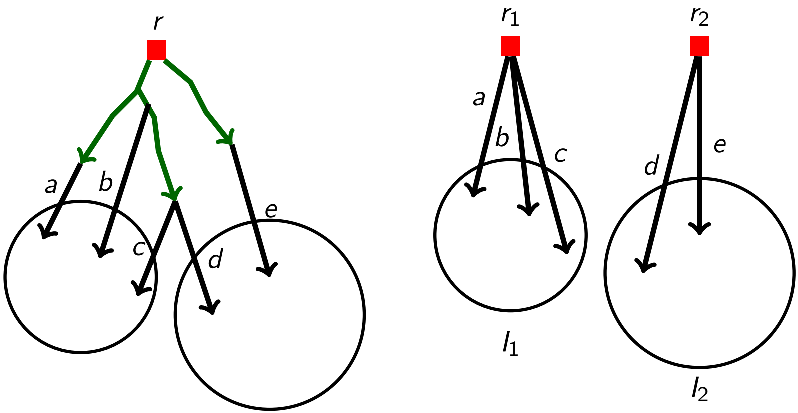
A slow algorithm

1. Guess opt value (poly(n) many choices). Remove all Steiner nodes with distance more than opt from r.
2. Compute a balanced shortest path separator rooted at r.
3. Compute the smaller subinstances.



A slow algorithm

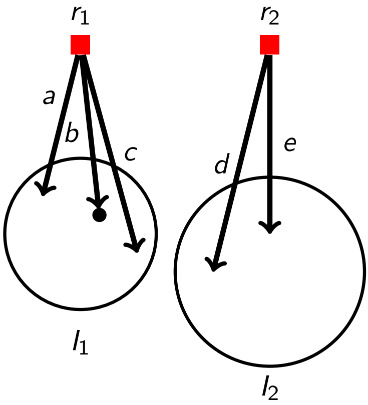
1. Guess opt value ($\mathrm{poly}(n)$ many choices). Remove all Steiner nodes with distance more than opt from r.
2. Compute a balanced shortest path separator rooted at r.
3. Compute the smaller subinstances.



Cost(balanced separator) $\leq 3 \cdot \mathrm{opt}$.

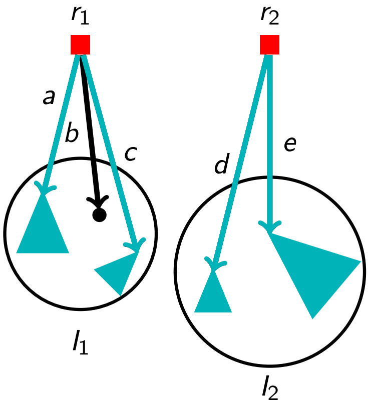
A slow algorithm

3. Solve the subinstances separately.



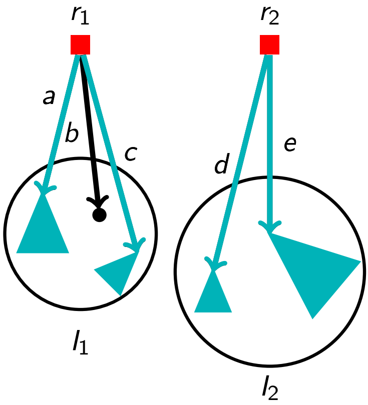
A slow algorithm

3. Solve the subinstances separately.



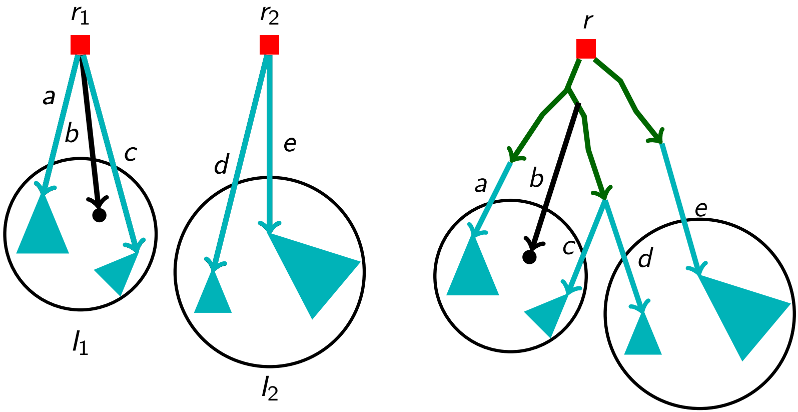
A slow algorithm

3. Solve the subinstances separately.
4. Merge the solutions of the subinstances using the balanced separator.



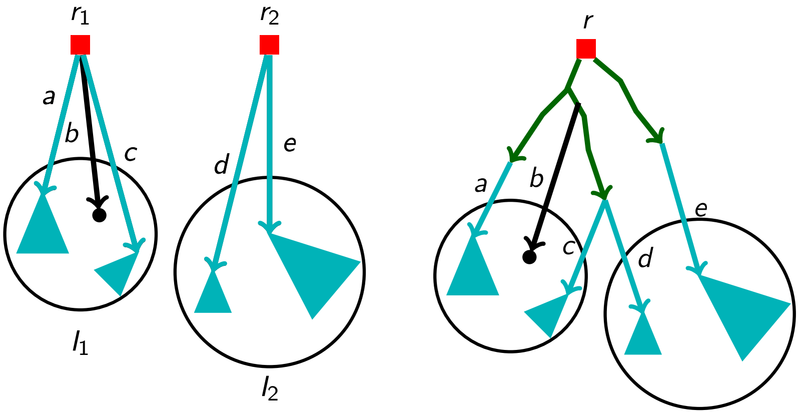
A slow algorithm

3. Solve the subinstances separately.
4. Merge the solutions of the subinstances using the balanced separator.



A slow algorithm

3. Solve the subinstances separately.
4. Merge the solutions of the subinstances using the balanced separator.



$\mathrm{opt}_1 + \mathrm{opt}_2 \leq \mathrm{opt}$.

How to make it run faster?!

Idea:

How to make it run faster?!

Idea: instead of guessing the opt, let the algorithm find (approximately) the opt.

How to make it run faster?!

Idea: instead of guessing the opt, let the algorithm find (approximately) the opt.

- Start with an upper bound $\widetilde{\mathrm{opt}}$ s.t. $\mathrm{opt} \leq \widetilde{\mathrm{opt}} \leq \mathrm{poly}(n)$.

How to make it run faster?!

Idea: instead of guessing the opt, let the algorithm find (approximately) the opt.

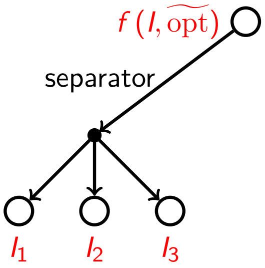
- Start with an upper bound $\widetilde{\mathrm{opt}}$ s.t. $\mathrm{opt} \le \widetilde{\mathrm{opt}} \le \mathrm{poly}(n)$.

$f(I, \widetilde{\mathrm{opt}})$ 

How to make it run faster?!

Idea: instead of guessing the opt, let the algorithm find (approximately) the opt.

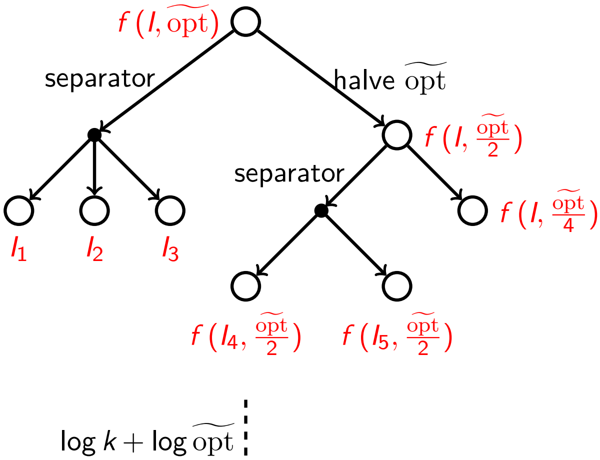
- Start with an upper bound $\widetilde{\mathrm{opt}}$ s.t. $\mathrm{opt} \leq \widetilde{\mathrm{opt}} \leq \mathrm{poly}(n)$.



How to make it run faster?!

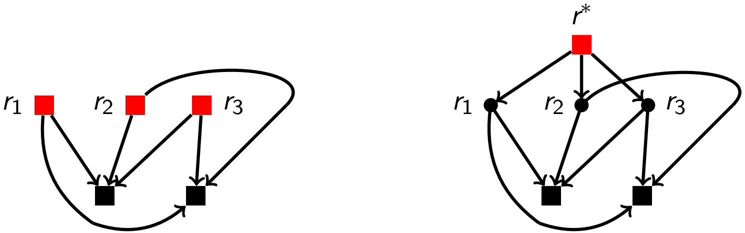
Idea: instead of guessing the opt, let the algorithm find (approximately) the opt.

- Start with an upper bound $\widetilde{\mathrm{opt}}$ s.t. $\mathrm{opt} \leq \widetilde{\mathrm{opt}} \leq \mathrm{poly}(n)$.



Extensions

- Trivially works for node-weighted planar DST. The usual reduction does not preserve planarity.
- R roots instead of one.



- We can get $O(R + \log k)$-approximation for multiple roots instances by extending Thorup's separator to multi-rooted instances.

Quasi-bipartite DST on planar graphs

Recall no edge between any two Steiner nodes and the input graph is planar.

Result: 20-approximation via a “modified” primal-dual scheme.

LP Relaxation

The LP relaxation:

$$\begin{array}{rrcll} \textbf{minimize}: & \sum_e c_e \cdot x_e & & & \\ \textbf{s.t.}: & x(\delta^{in}(S)) & \geq & 1 & \forall S \subseteq V - \{r\}, S \cap X \neq \emptyset \\ & x & \geq & 0 & \end{array}$$

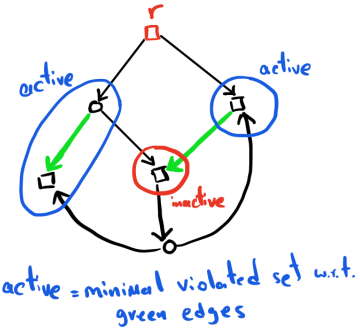
And the dual:

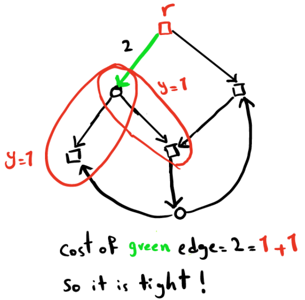
$$\begin{array}{rrcll} \textbf{maximize}: & \sum_S y_S & & & \\ \textbf{subject to}: & \sum_{S: e \in \delta^{in}(S)} y_S & \leq & c_e & \forall\ e \\ & y & \geq & 0 & \end{array}$$

What is known about this LP?

- 2 in undirected graphs.
- $\Omega(\sqrt{k})$ [Zosin and Khuller, 2002], also $\Omega(n^{0.0418})$ [Li and Laekhanukit, 2022].
- $O(\log k)$ in quasi-bipartite graphs [F., Konemann, and Shadravan, 2016].

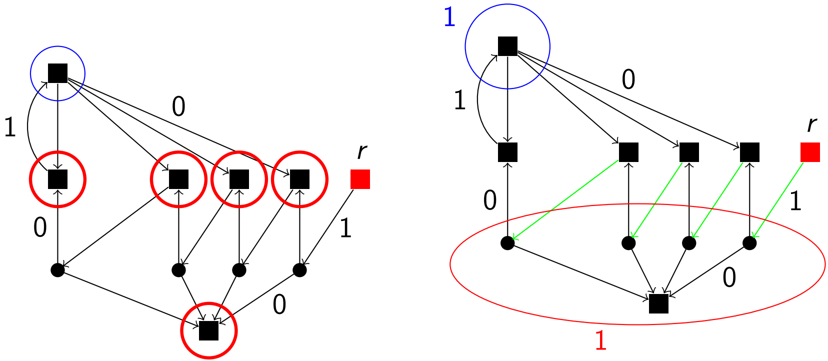
Primal-dual basics





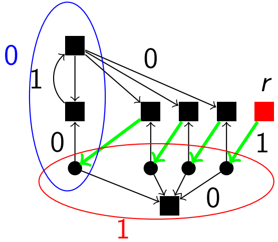
- increase active sets (moats) until an edge goes tight. Add the edge in to your solution.
- do a post-processing (reverse delete)
- the total cost of edges bought should be "comparable" to the total dual value increased.

What goes wrong on DST?!



- the bottom moat raised its dual value from zero to 1 but is responsible for purchasing many (4 here) green edges.
- the total dual raised in the algorithm is 2 but optimal solution has cost $4 + 1$ (note we can replace 4 by an arbitrary large number).

Fixing the Problem



Edges $e = (u, v)$ serve one of three roles to each moat it enters

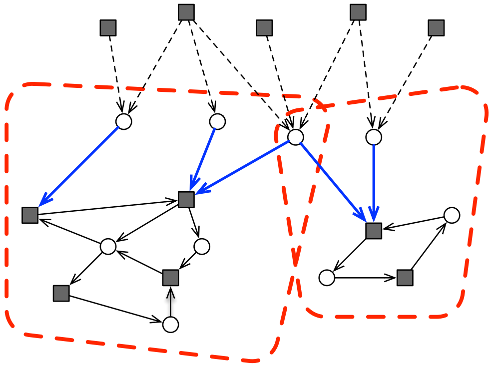
- **Antenna**: u is Steiner, v is a terminal, else
- **Killer**: if e is purchased the moat will die, else
- **Expansion**: purchasing e will expand the moat.

Each edge e has three "buckets" for money.

- Active moats pay into appropriate buckets for edges.
- When one of the buckets of e "fills", buy e (break ties by buying only one).
- Standard reverse delete.

Analysis - Structure of Active Moats

Consider a given set $F \subseteq E$ purchased so far.



Active moats are (disjoint) strongly-connected components of F containing a terminal plus **purchased antennas**, i.e. edges entering the SCC from Steiner nodes.

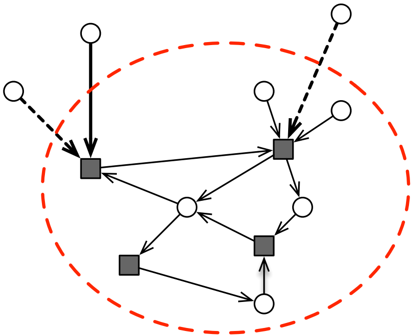
Overlap between moats is limited to incoming Steiner nodes.

Analysis

We show the active moats are paying, on average, toward $O(1)$ buckets of final edges to provide the approx. guarantee.

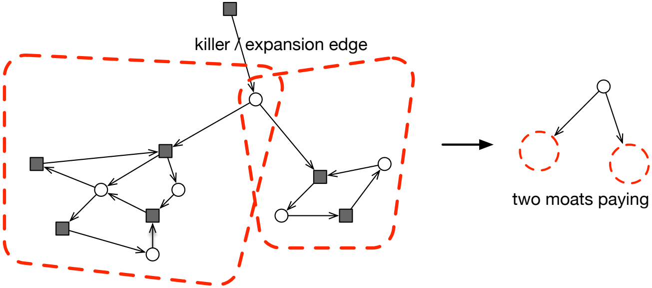
We handle this in three cases: antenna, killer, expansion edges.

Very easy to bound antenna edges: no active moat has more than one incoming antenna edge (reverse delete).



Analysis - Killer & Expansion Edges

Claim: If # killer + expansion edges is $O(1)$ times # active moats, we are done.



To see this:

- Contract the SCC part of all active moats (i.e. not antenna edges). Graph remains planar.
- Average degree counting arguments.

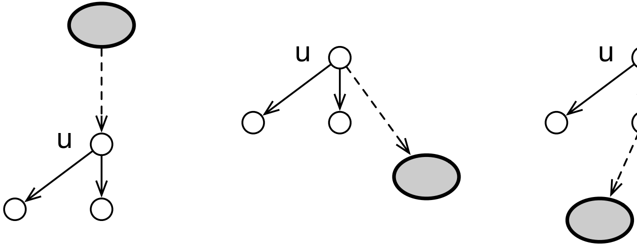
We also have # killer $\leq$ # active moats (each moat sees at most one).

Analysis - Expansion Edges

(**High-Level Idea**): We establish a tree of active/inactive moats and expansion edges (u, v) with the following properties.

1) Each "leaf" in the tree is an active moat.
and
2) For each expansion edge $e = (u, v)$, either

- The ancestor of u is an active moat that can reach u without using other expansion edges, **or**
- An active moat lies under u separated by ≤ 1 expansion edge.



A token argument then finishes the counting.

Next steps

- Is there a PTAS? Even $O(1)$ for planar DST (non-QBT) is an important open problem.
- The integrality gap could be $O(1)$ in planar graphs.