On the Tree Search Problem with Non-uniform costs

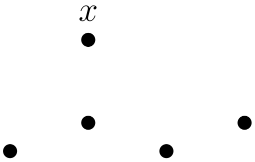
Ferdinando Cicalese, Balázs Keszegh, Bernard Lidický,
Dömötör Pálvölgyi, Tomáš Valla

University of Salerno, Rényi Institute, University of Illinois at Urbana-Champaign,
Eötvös University, Czech Technical University

27^{th} Cumberland Conference on Combinatorics,
Graph Theory & Computing
May 17, 2014

The search problem - general version

Input: n objects where one is special
Output: The special object

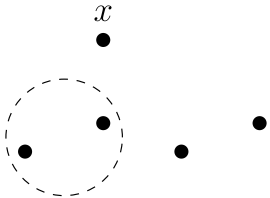


Questions: Is the special object in a subset S?
Answers: Yes/No.

The search problem - general version

Input: n objects where one is special

Output: The special object

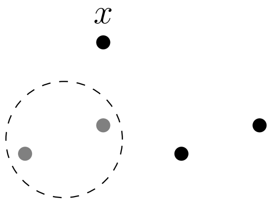


Questions: Is the special object in a subset S?

Answers: Yes/No.

The search problem - general version

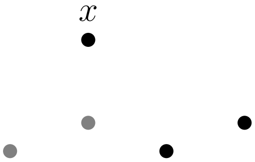
Input: n objects where one is special
Output: The special object



Questions: Is the special object in a subset S?
Answers: Yes/No.

The search problem - general version

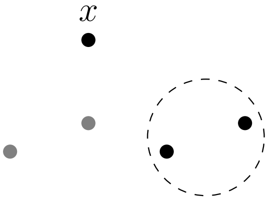
Input: n objects where one is special
Output: The special object



Questions: Is the special object in a subset S?
Answers: Yes/No.

The search problem - general version

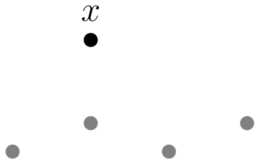
Input: n objects where one is special
Output: The special object



Questions: Is the special object in a subset S?
Answers: Yes/No.

The search problem - general version

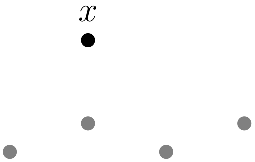
Input: n objects where one is special
Output: The special object



Questions: Is the special object in a subset S?
Answers: Yes/No.

The search problem - general version

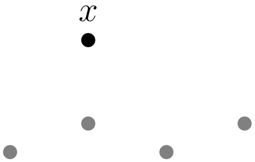
Input: n objects where one is special
Output: The special object



Questions: Is the special object in a subset S?
Answers: Yes/No.
Objective: Minimize number of questions.

The search problem - general version

Input: n objects where one is special
Output: The special object

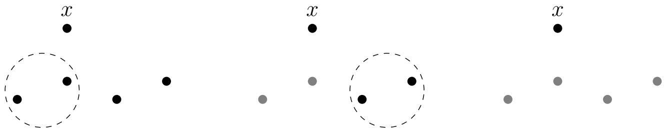


Questions: Is the special object in a subset S?
Answers: Yes/No.
Objective: Minimize number of questions.

Find a treasure quickly!

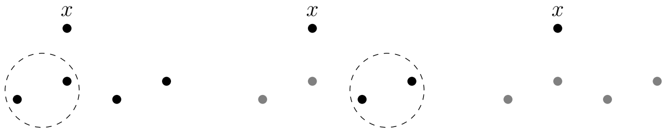
Online and offline versions

Online: Questions and answers alternate.

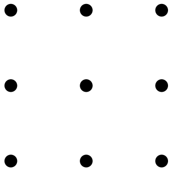


Online and offline versions

Online: Questions and answers alternate.

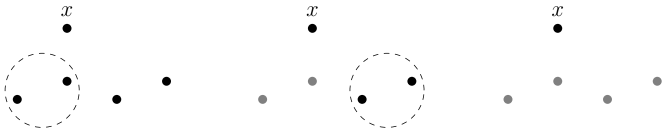


Offline: All questions first, then all answers.

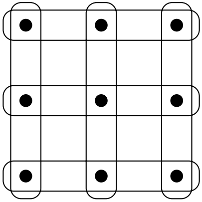


Online and offline versions

Online: Questions and answers alternate.

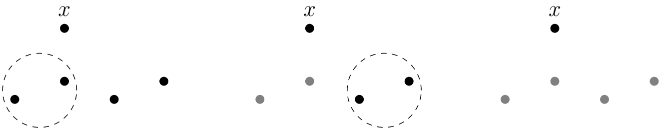


Offline: All questions first, then all answers.

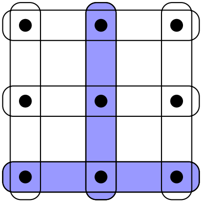


Online and offline versions

Online: Questions and answers alternate.

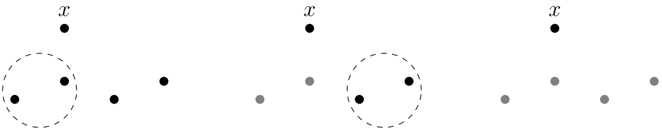


Offline: All questions first, then all answers.

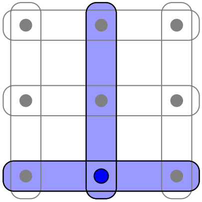


Online and offline versions

Online: Questions and answers alternate.

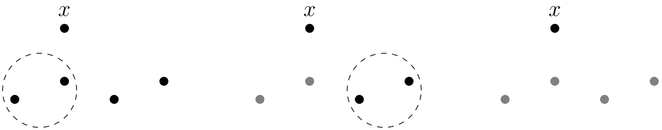


Offline: All questions first, then all answers.

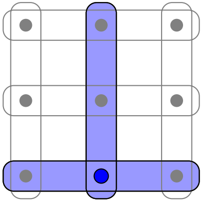


Online and offline versions

Online: Questions and answers alternate.



Offline: All questions first, then all answers.



We consider only online version.

Searching

Find 4 in an ordered list using binary search.

$1 \quad 4 \quad 6 \quad 7$

Searching

Find 4 in an ordered list using binary search.

$$\boxed{1 \quad 4} \quad 6 \quad 7$$

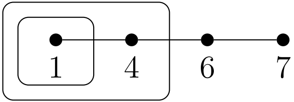
Searching

Find 4 in an ordered list using binary search.

$\boxed{1}$ $\quad 4 \quad 6 \quad 7$

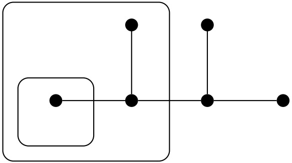
Searching

Find 4 in an ordered list using binary search.



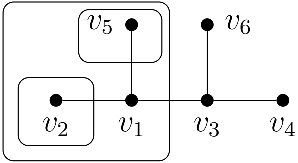
Searching

Find 4 in an ordered list using binary search.



Searching in trees

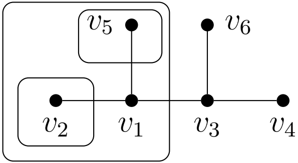
Input: Tree T, one hidden vertex



Output:

Searching in trees

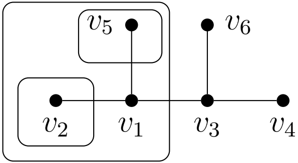
Input: Tree T, one hidden vertex



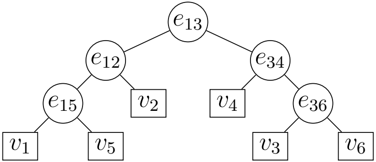
Output: Decision tree D of minimum depth

Searching in trees

Input: Tree T

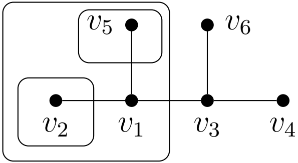


Output: Decision tree D of minimum depth

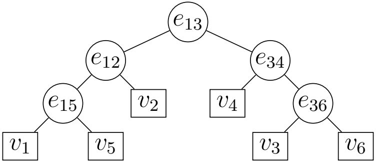


Searching in trees

Input: Tree T



Output: Decision tree D of minimum depth

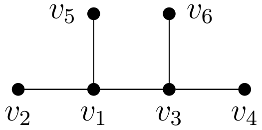


Theorem (Lam, Yue '98)

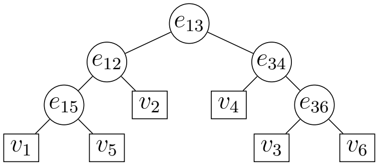
An optimal decision tree D can be computed in a polynomial time.

Searching in trees with costs on edges

Input: Tree T

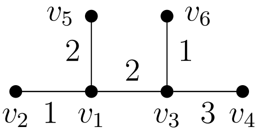


Output: Optimal decision tree D

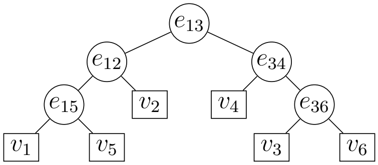


Searching in trees with costs on edges

Input: Tree T, cost $c : E(T) \to \mathbb{Z}$

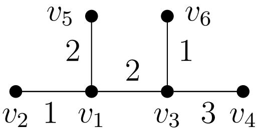


Output: Optimal decision tree D

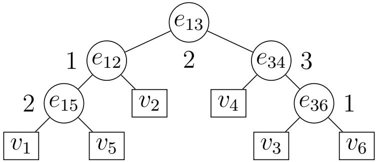


Searching in trees with costs on edges

Input: Tree T, cost $c : E(T) \to \mathbb{Z}$

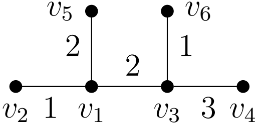


Output: Optimal decision tree D

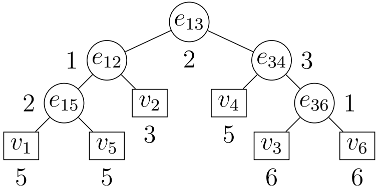


Searching in trees with costs on edges

Input: Tree T, cost $c : E(T) \to \mathbb{Z}$

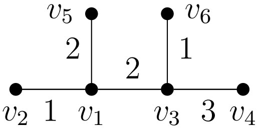


Output: Optimal decision tree D (cheapest worst case)

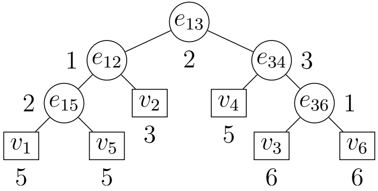


Searching in trees with costs on edges

Input: Tree T, cost $c : E(T) \to \mathbb{Z}$



Output: Optimal decision tree D (cheapest worst case)



$$cost(D) := \max_{v \in V(T)} \{cost^D(v)\} \quad OPT(T, c) := \min_{D} \{cost(D)\}$$

Known results

Input: Tree T on n vertices, cost c
Output: Decision tree D with $cost(D) = OPT(T, c)$

Theorem (Dereniowsky, '06)

NP-complete if diameter of T is 10
$O(\log(n))$*-approximation algorithm*

$$cost(D) = O(\log(n)) \cdot OPT(T, c)$$

Known results

Input: Tree T on n vertices, cost c
Output: Decision tree D with $cost(D) = OPT(T, c)$

Theorem (Dereniowsky, '06)

NP-complete if diameter of T is 10
$O(\log(n))$*-approximation algorithm*

$$cost(D) = O(\log(n)) \cdot OPT(T, c)$$

Theorem (Cicalese, Jacobs, Laber, Valentin '12)

NP-complete for diameter of T is 6
NP-complete for max degree of T is 3

Known results

Input: Tree T on n vertices, cost c
Output: Decision tree D with $cost(D) = OPT(T, c)$

Theorem (Dereniowsky, '06)

NP-complete if diameter of T is 10
$O(\log(n))$*-approximation algorithm*

$$cost(D) = O(\log(n)) \cdot OPT(T, c)$$

Theorem (Cicalese, Jacobs, Laber, Valentin '12)

NP-complete for diameter of T is 6
NP-complete for max degree of T is 3
$O(n^2)$*-time algorithm if T is a path*
$O(n2^n)$*-time algorithm*

Known results

Input: Tree T on n vertices, cost c
Output: Decision tree D with $cost(D) = OPT(T, c)$

Theorem (Dereniowsky, '06)

NP-complete if diameter of T is 10
$O(\log(n))$*-approximation algorithm*

$$cost(D) = O(\log(n)) \cdot OPT(T, c)$$

Theorem (Cicalese, Jacobs, Laber, Valentin '12)

NP-complete for diameter of T is 6
NP-complete for max degree of T is 3
$O(n^2)$*-time algorithm if T is a path*
$O(n2^n)$*-time algorithm*
$O\left(\frac{\log n}{\log\log\log n}\right)$*-approximation algorithm*

Our results

Input: Tree T on n vertices, cost c
Output: Decision tree D with $cost(D) = OPT(T, c)$

Theorem (Cicalese, Keszegh, L., Pálvölgyi, Valla)

NP-complete for diameter 6 subdivided stars (from Knapsack)
$O\left(\frac{\log n}{\log\log n}\right)$*-approximation algorithm*

Our results

Input: Tree T on n vertices, cost c
Output: Decision tree D with $cost(D) = OPT(T, c)$

Theorem (Cicalese, Keszegh, L., Pálvölgyi, Valla)

NP-complete for diameter 6 subdivided stars (from Knapsack)
$O\left(\frac{\log n}{\log \log n}\right)$*-approximation algorithm*

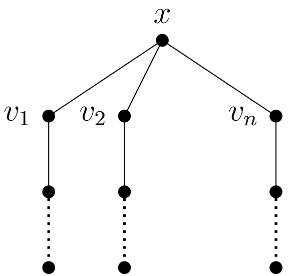
    

Problem
Is there $O\left(\frac{\log n}{\log n}\right)$*-approximation algorithm?*

Proof warm-up

Lemma

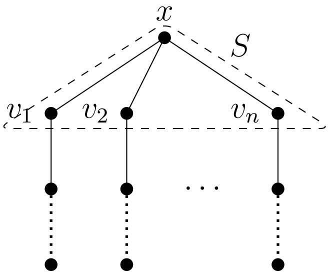
There is a 2-approximation algorithm for subdivided stars.



Proof warm-up

Lemma

There is a 2-approximation algorithm for subdivided stars.

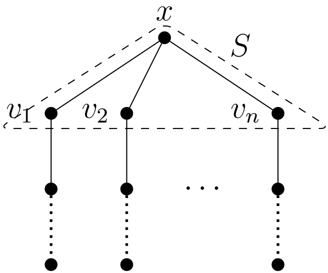


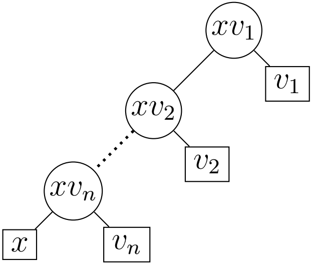
Find a decision tree for S

Proof warm-up

Lemma

There is a 2-approximation algorithm for subdivided stars.



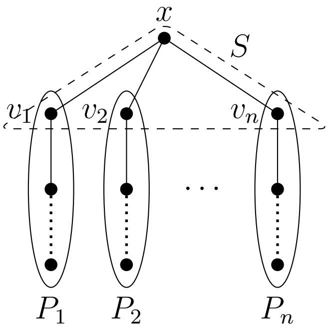


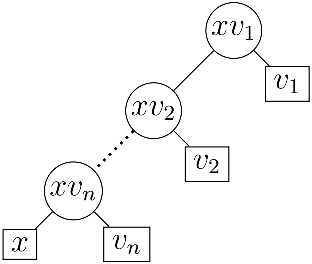
Find a decision tree for S
Cost : $OPT(T, c)$

Proof warm-up

Lemma

There is a 2-approximation algorithm for subdivided stars.





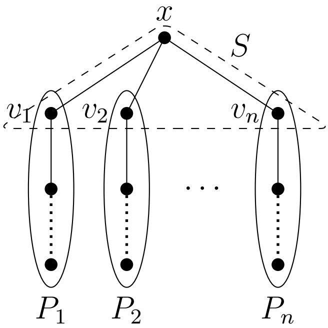
Find a decision tree for S

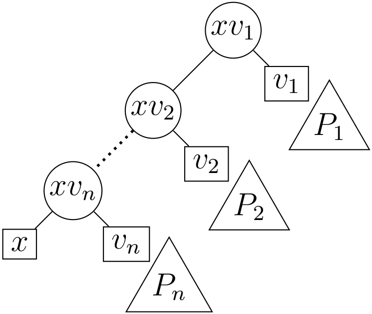
Cost : $OPT(T, c)$

Proof warm-up

Lemma

There is a 2-approximation algorithm for subdivided stars.





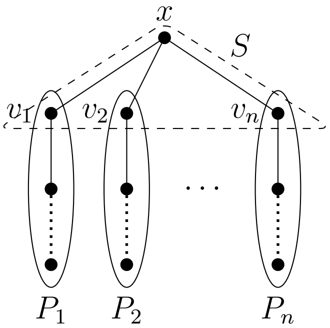
Find a decision tree for S and for P_i

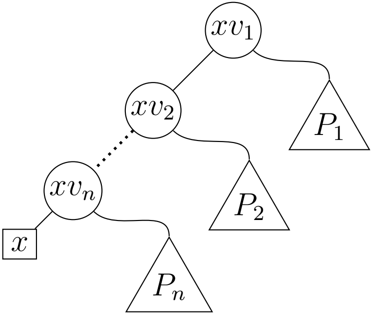
$Cost : OPT(T, c)$

Proof warm-up

Lemma

There is a 2-approximation algorithm for subdivided stars.

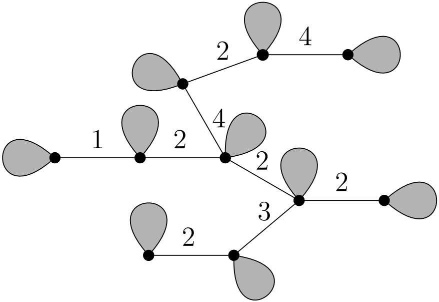




Find a decision tree for S and for P_i and combine them.
Cost : $OPT(T, c) + OPT(T, c)$

Proof sketch (algorithm for trees)

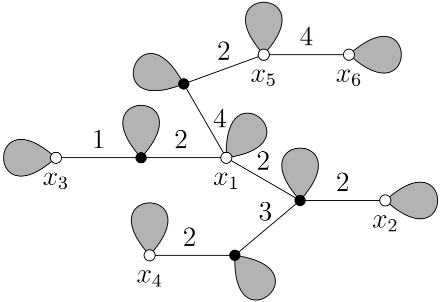
Idea: Find a small separator S, small resulting components, recurse



Fix $t = \log(n)$;

Proof sketch (algorithm for trees)

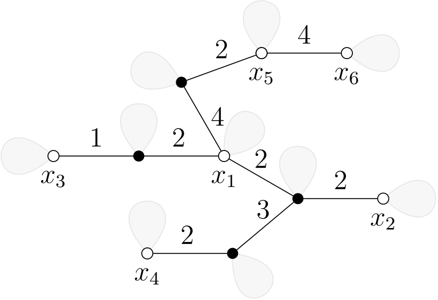
Idea: Find a small separator S, small resulting components, recurse



Fix $t = \log(n)$; pick t centroids $S = \{x_1 \ldots\}$; (components n/t)

Proof sketch (algorithm for trees)

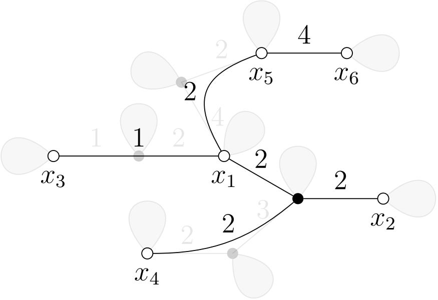
Idea: Find a small separator S, small resulting components, recurse



Fix $t = \log(n)$; pick t centroids $S = \{x_1 \ldots\}$; (components n/t)
build auxiliary graph Y of size $2t$

Proof sketch (algorithm for trees)

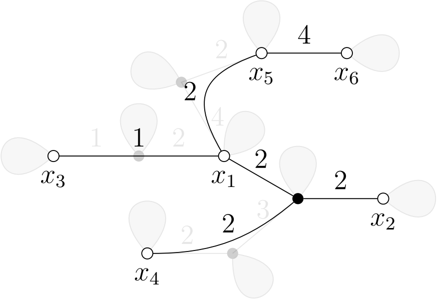
Idea: Find a small separator S, small resulting components, recurse

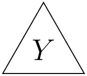


Fix $t = \log(n)$; pick t centroids $S = \{x_1 \ldots\}$; (components n/t)
build auxiliary graph Y of size $2t$

Proof sketch (algorithm for trees)

Idea: Find a small separator S, small resulting components, recurse



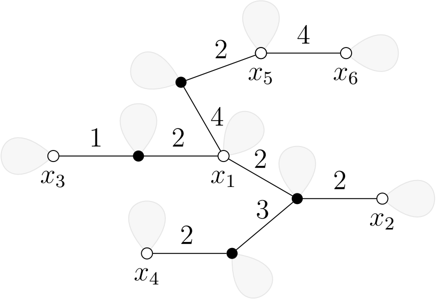


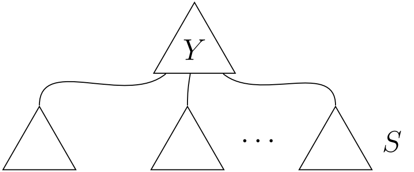
Fix $t = \log(n)$; pick t centroids $S = \{x_1 \ldots\}$; (components n/t)
build auxiliary graph Y of size $2t$, solve Y in $O(t2^{2t})$

Cost: $OPT(T, c)$

Proof sketch (algorithm for trees)

Idea: Find a small separator S, small resulting components, recurse



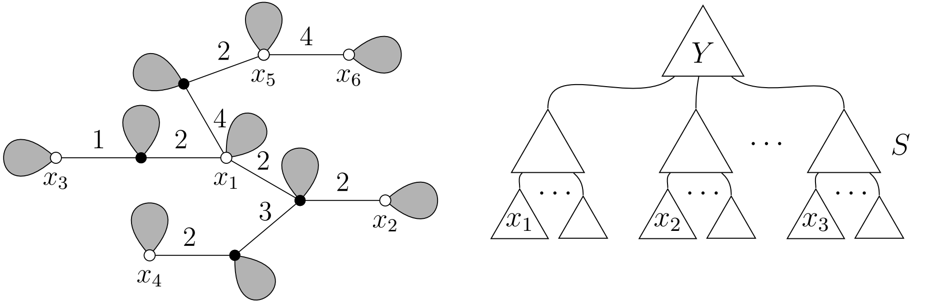


Fix $t = \log(n)$; pick t centroids $S = \{x_1 \ldots\}$; (components n/t)
build auxiliary graph Y of size $2t$, solve Y in $O(t2^{2t})$
solve subdivided stars in S

Cost: $OPT(T, c) + 2OPT(T, c)$

Proof sketch (algorithm for trees)

Idea: Find a small separator S, small resulting components, recurse

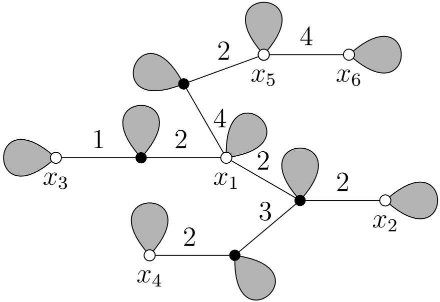


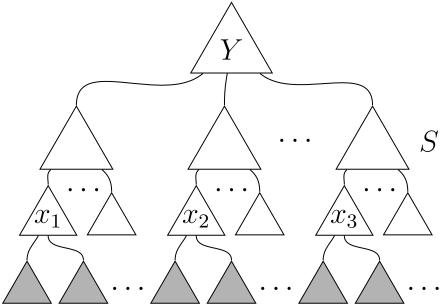
Fix $t = \log(n)$; pick t centroids $S = \{x_1 \ldots\}$; (components n/t)
build auxiliary graph Y of size $2t$, solve Y in $O(t2^{2t})$
solve subdivided stars in S, solve neighbors of S

Cost: $OPT(T, c) + 2OPT(T, c) + OPT(T, c)$

Proof sketch (algorithm for trees)

Idea: Find a small separator S, small resulting components, recurse



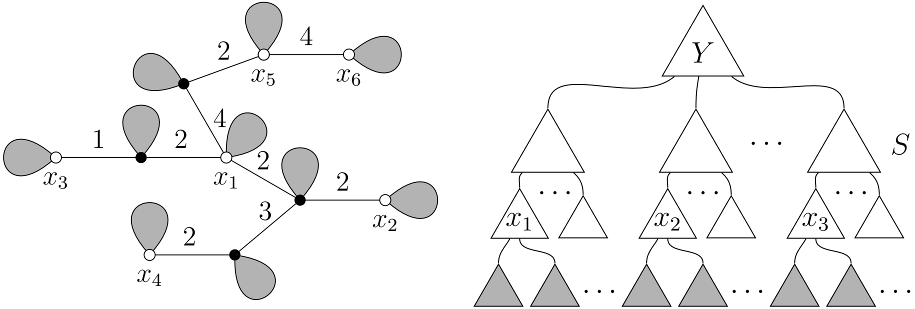


Fix $t = \log(n)$; pick t centroids $S = \{x_1 \ldots\}$; (components n/t)
build auxiliary graph Y of size $2t$, solve Y in $O(t2^{2t})$
solve subdivided stars in S, solve neighbors of S, recursion $\times k$;

Cost: $(OPT(T,c) + 2OPT(T,c) + OPT(T,c)) \times k$

Proof sketch (algorithm for trees)

Idea: Find a small separator S, small resulting components, recurse



Fix $t = \log(n)$; pick t centroids $S = \{x_1 \ldots\}$; (components n/t)
build auxiliary graph Y of size $2t$, solve Y in $O(t2^{2t})$
solve subdivided stars in S, solve neighbors of S, recursion $\times k$;
$t^k = n$ hence $k = \log(n)/\log(t)$
Cost: $(OPT(T,c) + 2OPT(T,c) + OPT(T,c)) \times k$

Thank you for your attention!