Searching in Graphs

Michał Szyfelbein

Faculty of Electronics, Telecommunications and Informatics
Gdańsk University of Technology, Poland

October 6, 2025

Binary Search

Binary Search - a classical strategy used to efficiently locate a hidden target element t in a linearly ordered set S using $O(\log n)$ comparison operations.

Binary Search

Binary Search - a classical strategy used to efficiently locate a hidden target element t in a linearly ordered set S using $O(\log n)$ comparison operations.

median

$\downarrow$

1	2	4	5	7	10	15	16	18	21	22	25	31	32

lesser than median — greater than median

Figure: Example of a sorted array containing 14 elements.

Searching in Graphs

Easy to generalize to trees: A **query** to a vertex v returns information whether v is the target, and if not, which connected component of $G - v$ contains t. The question is:

What is the best strategy of searching in a graph?

Searching in Graphs

Easy to generalize to trees: A **query** to a vertex v returns information whether v is the target, and if not, which connected component of $G - v$ contains t. The question is:

What is the best strategy of searching in a graph?

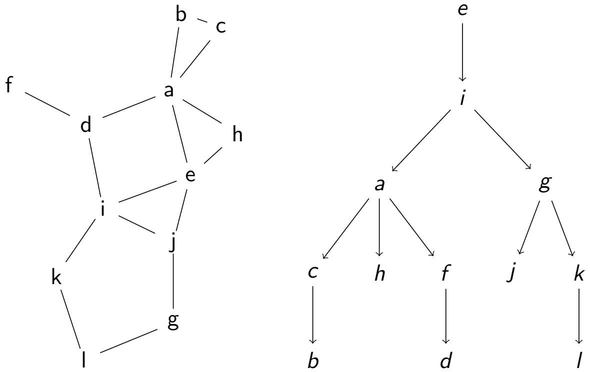


Figure: Sample input graph a decision tree for it.

Our setup

We consider the following variant of this problem:

Graph Search Problem (GSP)

Input: Tree G, a query cost function $c : V(G) \to \mathbb{N}$ and a weight function $w : V(G) \to \mathbb{N}$.

Our setup

We consider the following variant of this problem:

Graph Search Problem (GSP)

Input: Tree G, a query cost function $c : V(G) \to \mathbb{N}$ and a weight function $w : V(G) \to \mathbb{N}$.

Output: A decision tree D minimizing the weighted average search cost:

$$c_G(D) = \sum_{x \in V(G)} w(x) \cdot \sum_{q \in Q_G(D,x)} c(q)\,.$$

where $Q_G(D, x)$ denotes the sequence of queries performed along the unique path in D from the root $r(D)$ to x.

Why do we care?

Useful in:

1. Scheduling of parallel database join operations,
2. Automated bug detection in computer code,
3. Parallel Cholesky factorization of matrices,
4. Hierarchical clustering of data,
5. Parallel assembly of multi-part products from their components.

Why do we care?

Our setup:

1. Average case - it is natural to assume that the search strategies we design are intended to be used repeatedly.
2. Weight function - some vertices may serve as targets more frequently than the others.
3. Query costs - performing a query may require significant resources, such as time or money.
4. Have not yet been investigated.

Many names

- Binary Search [OP06; Der+17; DMS19; EKS16; DW22; DW24; DŁU25; DGW24; DŁU21; DGP23],
- Tree Search Problem [Jac+10; Cic+14; Cic+16],
- Binary Identification Problem [Cic+12],
- Ranking Colorings [Knu73; Der06; Der08; DK06; DN06; LY98],
- Ordered Colorings [KMS95],
- Elimination Trees [Pot88],
- Hub Labeling [Ang18],
- Tree-Depth [NO06; BDO23],
- Partition Trees [Høg24],
- Hierarchical Clustering [Das16; Coh+19; CC17],
- Search Trees on Trees [BK22; Ber+22],
- LIFO-Search [GHT12].

How to tackle the problem

Bad news: The Graph Search Problem is **NP-hard** even when restricted to bounded degree trees and bounded diameter diamter trees.

How to tackle the problem

Bad news: The Graph Search Problem is **NP-hard** even when restricted to bounded degree trees and bounded diameter diamter trees.

We want to find an algorithm providing a good **approximation** for the problem:

- $(4 + \epsilon)$-approximation for trees.
- $O\left(\sqrt{\log n}\right)$-approximation for general graphs.

Weighted α-Separator Problem

Weighted α-Separator Problem

Input: Graph G, a cost function $c : V \to \mathbb{N}$, a weight function $w : V \to \mathbb{N}$ and a real number α.

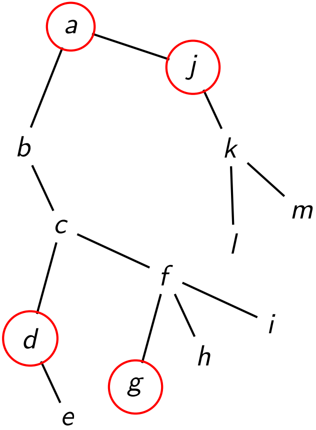
Weighted α-Separator Problem

Weighted α-Separator Problem

Input: Graph G, a cost function $c : V \to \mathbb{N}$, a weight function $w : V \to \mathbb{N}$ and a real number α.

Output: A set $S \subseteq V(G)$ called **separator** such that for every $H \in G - S$: $w(H) \leq w(G)/\alpha$ and $c(S)$ is minimized.

Example of a separator



	$w(v)$	$c(v)$
a	2	3
b	1	4
c	3	6
d	2	2
e	4	1
f	0	3
g	1	1
h	4	3
i	2	3
j	5	2
k	1	2
l	2	3
m	3	4

Figure: Sample input tree T and a weighted 3-separator (the circled vertices) of cost 8.

How to find the separator

Bad news again: The Weighted α-separator Problem is **NP-hard** as well, but there exists a biciriteria **FPTAS** for trees:

How to find the separator

Bad news again: The Weighted α-separator Problem is **NP-hard** as well, but there exists a biciriteria **FPTAS** for trees:

Theorem

Let S be an optimal weighted α-separator for (T, c, w, α). For any $\delta > 0$ there exists an algorithm `SeparatorFPTAS`*, which returns a separator S', such that:*

1. $c(S') \leq c(S)$.
2. $w(H) \leq \frac{(1+\delta)\cdot w(T)}{\alpha}$ *for every* $H \in T - S'$.
3. *The algorithm runs in* $O\left(n^3/\delta^2\right)$ *time.*

Min-Ratio Vertex Cut Problem

Min-Ratio Vertex Cut Problem

Input: Graph $G = (V(G), E(G))$, the cost function $c : V \to \mathbb{N}$ and the weight function $w : V \to \mathbb{N}$.

Min-Ratio Vertex Cut Problem

Min-Ratio Vertex Cut Problem

Input: Graph $G = (V(G), E(G))$, the cost function $c : V \to \mathbb{N}$ and the weight function $w : V \to \mathbb{N}$.

Output: A partition (A, S, B) of $V(G)$ called *vertex-cut*, such that there are no $u \in A$ and $v \in B$ for which $uv \in E(G)$, minimizing the ratio:

$$\alpha_{c,w}(A, S, B) = \frac{c(S)}{w(A \cup S) \cdot w(B \cup S)}.$$

How to find the cut

Min-Ratio Vertex Cut Problem is also **NP-hard**. However, we invoke the following result of [FHL05]:

Theorem

Given a graph $G = (V(G), E(G))$, *the cost function* $c : V \to \mathbb{N}$ *and the weight function* $w : V \to \mathbb{N}$, *there exists a polynomial-time algorithm, which computes a partition* (A, S, B), *such that:*

$$\alpha_{c,w}(A, S, B) = O\left(\sqrt{\log n}\right) \cdot \alpha_{c,w}(G).$$

Notation

- $\mathcal{R}_D(G) = \{V(G_{D,v}) \,|\, v \in V(G)\}$ - the family of all candidate subsets of D in G.
- D^* - optimal decision tree.
- $\mathcal{L}_k^*$ - the subfamily of $\mathcal{R}_{D^*}(G)$ consisting of all maximal elements H of $\mathcal{R}_{D^*}(G)$ with $w(H) \leq k$. We call such a set the k-th *level* of $\mathtt{OPT}(G)$.
- $S_k^* = V(G) - \mathcal{L}_k^*$ - vertices belonging to the separator at the level $\mathcal{L}_k^*$. S_k^* forms a Weighted $w(G)\,/k$-separator of G.
- For any $H_1, H_2 \in \mathcal{R}_D(G)$, we have $H_1 \cup H_2 \neq \emptyset$ if and only if $H_1 \subseteq H_2$ or $H_2 \subseteq H_1$, so $\mathcal{R}_D(G)$ is laminar. Thus, for any $k_1 \neq k_2$, we have $\mathcal{L}_{k_1}^* \cap \mathcal{L}_{k_2}^* = \emptyset$.

Example of the connection

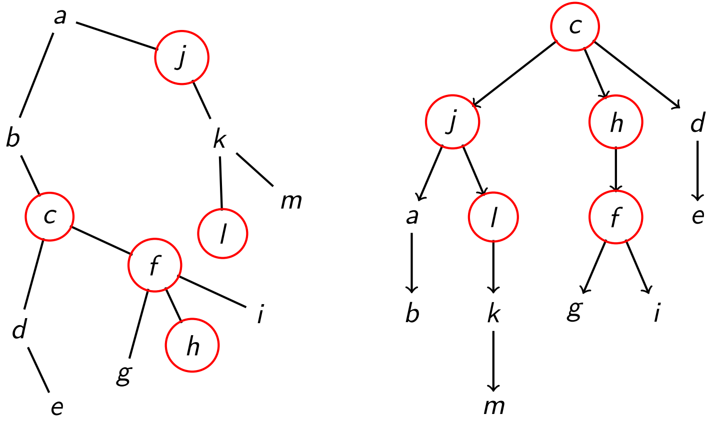


Figure: A 13/2-separator induced by the subtree of the decision tree consisting of circled vertices.

Basic lemmas

Lemma

Let $G_{D,v}$ be the candidate subgraph of G in which v is queried when using D. Then, $c_G(D) = \sum_{v \in V(G)} w(G_{D,v}) \cdot c(v)$.

Basic lemmas

Lemma

Let $G_{D,v}$ be the candidate subgraph of G in which v is queried when using D. Then, $c_G(D) = \sum_{v \in V(G)} w(G_{D,v}) \cdot c(v)$.

Lemma

$OPT(G) = \sum_{k=0}^{w(G)-1} c(S_k^*)$.

Basic lemmas

Lemma

Let $G_{D,v}$ be the candidate subgraph of G in which v is queried when using D. Then, $c_G(D) = \sum_{v \in V(G)} w(G_{D,v}) \cdot c(v)$.

Lemma

$\mathit{OPT}(G) = \sum_{k=0}^{w(G)-1} c(S_k^*)$.

Proof.

Consider any vertex v. For every $0 \leq k < w(G_{D^*,v})$, $v \notin \bigcup_{H \in \mathcal{L}_k^*} H$, so $v \in S_k^*$ and the contribution of v to the cost is $w(G_{D^*,v}) \cdot c(v)$:

$$\sum_{k=0}^{w(G)-1} c(S_k^*) = \sum_{v \in V(G)} \sum_{k=0}^{w(G_{D^*,v})-1} c(v) = \mathit{OPT}(G).$$

□

Basic lemmas

Lemma

$$2 \cdot \mathit{OPT}(G) = 2 \cdot \sum_{k=0}^{w(T)-1} c(S_k^*) \geq \sum_{k=0}^{w(T)} c\left(S_{\lfloor k/2 \rfloor}^*\right).$$

Basic lemmas

Lemma

$$2 \cdot OPT(G) = 2 \cdot \sum_{k=0}^{w(T)-1} c(S_k^*) \geq \sum_{k=0}^{w(T)} c\left(S^*_{\lfloor k/2 \rfloor}\right).$$

Lemma

Let $\mathcal{G}$ be any subgraph of G and $0 \leq \beta \leq 1$. Then:

$$\beta \cdot w(\mathcal{G}) \cdot c\left(S^*_{\lfloor w(\mathcal{G})/2 \rfloor} \cap \mathcal{G}\right) \leq \sum_{k=(1-\beta)w(\mathcal{G})+1}^{w(\mathcal{G})} c\left(S^*_{\lfloor k/2 \rfloor} \cap \mathcal{G}\right).$$

Searching in Trees

We will iteratively use the FPTAS for the Weighted α-separator problem to create an $(4 + \epsilon)$-approximation algorithm for the Tree Search Problem:

Searching in Trees

We will iteratively use the FPTAS for the Weighted α-separator problem to create an $(4+\epsilon)$-approximation algorithm for the Tree Search Problem:

Theorem

For any $\epsilon > 0$ there exists an $(4+\epsilon)$-approximation algorithm for the Tree Search Problem running in $O\left(n^4/\epsilon^2\right)$ time.

The algorithm

proc $\texttt{DecisionTree}(T, c, w, \epsilon)$:

1. $S_T \leftarrow \texttt{SeparatorFPTAS}\left(T, c, w, \alpha = 2, \delta = \frac{\epsilon}{4+\epsilon}\right)$.
2. $D_T \leftarrow$ arbitrary partial decision tree for T, built from vertices of S_T.
3. For each $H \in T - S_T$:
   3.1 $D_H \leftarrow \texttt{DecisionTree}(H, c, w, \epsilon)$.
   3.2 Hang D_H in D_T below the last query to $v \in N_T(H)$.
4. Return D_T.

Structure of the solution

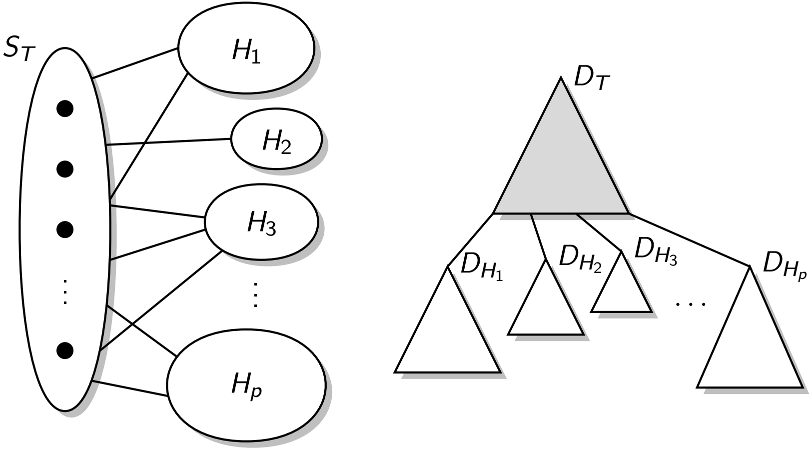


Figure: The separator S_T produced by the algorithm and the structure of the decision tree built using S_T.

Bounding the cost of a single recurrence call

- $\mathcal{T}$ - subtree of T for which the procedure was called.

Bounding the cost of a single recurrence call

- $\mathcal{T}$ - subtree of T for which the procedure was called.
- Let $S^*_{\mathcal{T}} = S^*_{\lfloor w(\mathcal{T})/2 \rfloor} \cap \mathcal{T}$.

Bounding the cost of a single recurrence call

- $\mathcal{T}$ - subtree of T for which the procedure was called.
- Let $S^*_{\mathcal{T}} = S^*_{\lfloor w(\mathcal{T})/2 \rfloor} \cap \mathcal{T}$.
- By the optimality of $S_{\mathcal{T}}$, $c(S_{\mathcal{T}}) \leq c(S^*_{\mathcal{T}})$.

Bounding the cost of a single recurrence call

- $\mathcal{T}$ - subtree of T for which the procedure was called.
- Let $S^*_{\mathcal{T}} = S^*_{\lfloor w(\mathcal{T})/2 \rfloor} \cap \mathcal{T}$.
- By the optimality of $S_{\mathcal{T}}$, $c(S_{\mathcal{T}}) \leq c(S^*_{\mathcal{T}})$.
- Let $\beta = \frac{1-\delta}{2}$. We have:

Bounding the cost of a single recurrence call

- $\mathcal{T}$ - subtree of T for which the procedure was called.
- Let $S^*_{\mathcal{T}} = S^*_{\lfloor w(\mathcal{T})/2 \rfloor} \cap \mathcal{T}$.
- By the optimality of $S_{\mathcal{T}}$, $c(S_{\mathcal{T}}) \leq c(S^*_{\mathcal{T}})$.
- Let $\beta = \frac{1-\delta}{2}$. We have:

$$w(\mathcal{T}) \cdot c(S_{\mathcal{T}})$$

$$\leq w(\mathcal{T}) \cdot c(S^*_{\mathcal{T}}) \leq \frac{2}{1-\delta} \cdot \sum_{k=\frac{1+\delta}{2} \cdot w(\mathcal{T})+1}^{w(\mathcal{T})} c\left(S^*_{\lfloor k/2 \rfloor} \cap \mathcal{T}\right).$$

Bounding the cost of the solution

- D - the decision tree returned by DecisionTree(T, c, w, ϵ).

Bounding the cost of the solution

- D - the decision tree returned by DecisionTree(T, c, w, ϵ).
- By definition: $\frac{4}{1-\delta} = 4 + \epsilon$. Therefore:

Bounding the cost of the solution

- D - the decision tree returned by DecisionTree(T, c, w, ϵ).
- By definition: $\frac{4}{1-\delta} = 4 + \epsilon$. Therefore:

$$
\begin{aligned}
c_T(D) &\leq \sum_{\mathcal{T}} w(\mathcal{T}) \cdot c(S_{\mathcal{T}}) \\
&\leq \frac{2}{1-\delta} \cdot \sum_{\mathcal{T}} \sum_{k=\frac{1+\delta}{2}\cdot w(\mathcal{T})+1}^{w(\mathcal{T})} c\left(S^*_{\lfloor k/2 \rfloor} \cap \mathcal{T}\right) \\
&\leq \frac{2}{1-\delta} \cdot \sum_{k=0}^{w(T)} c\left(S^*_{\lfloor k/2 \rfloor}\right) \leq \frac{4}{1-\delta} \cdot \mathtt{OPT}(T) \\
&= (4+\epsilon) \cdot \mathtt{OPT}(T)
\end{aligned}
$$

Bounding the cost of the solution

- D - the decision tree returned by DecisionTree(T, c, w, ϵ).
- By definition: $\frac{4}{1-\delta} = 4 + \epsilon$. Therefore:

$$
\begin{aligned}
c_T(D) &\leq \sum_{\mathcal{T}} w(\mathcal{T}) \cdot c(S_{\mathcal{T}}) \\
&\leq \frac{2}{1-\delta} \cdot \sum_{\mathcal{T}} \sum_{k=\frac{1+\delta}{2}\cdot w(\mathcal{T})+1}^{w(\mathcal{T})} c\left(S^*_{\lfloor k/2 \rfloor} \cap \mathcal{T}\right) \\
&\leq \frac{2}{1-\delta} \cdot \sum_{k=0}^{w(T)} c\left(S^*_{\lfloor k/2 \rfloor}\right) \leq \frac{4}{1-\delta} \cdot \mathtt{OPT}(T) \\
&= (4+\epsilon) \cdot \mathtt{OPT}(T)
\end{aligned}
$$

- $1/\delta = (4+\epsilon)/\epsilon = 4/\epsilon + 1$ so the running time is $O\left(n^4/\epsilon^2\right)$.

Searching in Graphs

We will iteratively use the f_n-approximation algorithm for the Min-Ratio Vertex Cut Problem to create an $O(f_n)$-approximation algorithm for GSP:

Searching in Graphs

We will iteratively use the f_n-approximation algorithm for the Min-Ratio Vertex Cut Problem to create an $O(f_n)$-approximation algorithm for GSP:

Theorem

Let f_n be the approximation ratio of any polynomial time algorithm for the Min-Ratio Vertex Cut Problem. Then, there exists an $O(f_n)$-approximation algorithm for the GSP, running in polynomial time.

The algorithm

proc `DecisionTree`(T, c, w, ϵ):

1. $A_G, S_G, B_G \leftarrow$ `AlgorithmMinCut`(G, c, w).
2. $D_G \leftarrow$ arbitrary partial decision tree for G, built from vertices of S_G.
3. For each $H \in G - S_G$:
   3.1 $D_H \leftarrow$ `DecisionTree`(H, c, w).
   3.2 Hang D_H in D_G below the last query to $v \in N_G(H)$.
4. Return D_G.

Key technical lemma

- $\mathcal{G}$ - any subgraph of G, for which the procedure was called

Key technical lemma

- $\mathcal{G}$ - any subgraph of G, for which the procedure was called
- $S^*_{\mathcal{G}} = S^*_{\lfloor w(\mathcal{G})/2 \rfloor} \cap \mathcal{G}$.

Key technical lemma

- $\mathcal{G}$ - any subgraph of G, for which the procedure was called
- $S^*_{\mathcal{G}} = S^*_{\lfloor w(\mathcal{G})/2 \rfloor} \cap \mathcal{G}$.

Lemma

Let $\mathcal{H} = \mathcal{G} - S^*_{\mathcal{G}}$ and let $\lambda = 6 + 2\sqrt{5}$ be the unique, positive solution of the equation $\frac{1}{4} - \frac{1}{2\sqrt{\lambda}} = \frac{1}{\lambda}$. Then, we can partition $\mathcal{H}$ into two sets, $\mathcal{A}$ and $\mathcal{B}$ such that for $A = \bigcup_{H \in \mathcal{A}} V(H)$ and $B = \bigcup_{H \in \mathcal{B}} V(H)$, we have:

$$w\left(A \cup S^*_{\mathcal{G}}\right) \cdot w\left(B \cup S^*_{\mathcal{G}}\right) \geq w(\mathcal{G})^2 / \lambda.$$

Key technical lemma

Proof.

There are two cases:

1. $w\left(S^*_{\mathcal{G}}\right) \geq w(\mathcal{G})/\sqrt{\lambda}$. Take arbitrary partition $\mathcal{A}, \mathcal{B}$ of $\mathcal{H}$. We have: $w\left(A \cup S^*_{\mathcal{G}}\right) \cdot w\left(B \cup S^*_{\mathcal{G}}\right) \geq w\left(S^*_{\mathcal{G}}\right)^2 \geq w(\mathcal{G})^2/\lambda$.

Key technical lemma

Proof.

There are two cases:

1. $w\left(S^*_{\mathcal{G}}\right) \geq w(\mathcal{G})/\sqrt{\lambda}$. Take arbitrary partition $\mathcal{A}, \mathcal{B}$ of $\mathcal{H}$. We have: $w\left(A \cup S^*_{\mathcal{G}}\right) \cdot w\left(B \cup S^*_{\mathcal{G}}\right) \geq w\left(S^*_{\mathcal{G}}\right)^2 \geq w(\mathcal{G})^2/\lambda$.
2. $w\left(S^*_{\mathcal{G}}\right) \leq w(\mathcal{G})/\sqrt{\lambda}$. For any $\mathcal{A}, \mathcal{B}$, $\frac{w(A \cup B)}{w(\mathcal{G})} \geq 1 - \frac{1}{\sqrt{\lambda}}$. Pick $\mathcal{A}, \mathcal{B}$, so that $w(A) \geq w(B) \geq \left(\frac{1}{2} - \frac{1}{\sqrt{\lambda}}\right) \cdot w(\mathcal{G})$ (always possible as $\frac{1}{2} - \frac{1}{\sqrt{\lambda}} > 0$ and for each H, $w(H) \leq w(\mathcal{G})/2$):

Key technical lemma

Proof.

There are two cases:

1. $w\left(S_{\mathcal{G}}^{*}\right) \geq w(\mathcal{G}) / \sqrt{\lambda}$. Take arbitrary partition $\mathcal{A}, \mathcal{B}$ of $\mathcal{H}$. We have: $w\left(A \cup S_{\mathcal{G}}^{*}\right) \cdot w\left(B \cup S_{\mathcal{G}}^{*}\right) \geq w\left(S_{\mathcal{G}}^{*}\right)^{2} \geq w(\mathcal{G})^{2} / \lambda$.
2. $w\left(S_{\mathcal{G}}^{*}\right) \leq w(\mathcal{G}) / \sqrt{\lambda}$. For any $\mathcal{A}, \mathcal{B}$, $\frac{w(A \cup B)}{w(\mathcal{G})} \geq 1 - \frac{1}{\sqrt{\lambda}}$. Pick $\mathcal{A}, \mathcal{B}$, so that $w(A) \geq w(B) \geq \left(\frac{1}{2} - \frac{1}{\sqrt{\lambda}}\right) \cdot w(\mathcal{G})$ (always possible as $\frac{1}{2} - \frac{1}{\sqrt{\lambda}} > 0$ and for each H, $w(H) \leq w(\mathcal{G}) / 2$):

$$\begin{aligned} w\left(A \cup S_{\mathcal{G}}^{*}\right) \cdot w\left(B \cup S_{\mathcal{G}}^{*}\right) &\geq w(A) \cdot w(B) \\ &\geq \left(\left(1 - 1/\sqrt{\lambda}\right) \cdot w(\mathcal{G}) - w(B)\right) \cdot w(B) \\ &\geq w(\mathcal{G})^{2}/2 \cdot \left(1/2 - 1/\sqrt{\lambda}\right) = w(\mathcal{G})^{2}/\lambda. \end{aligned}$$

Bounding the cost of a single recurrence call

- $(A, S^*_{\mathcal{G}}, B)$ in the above lemma is a vertex cut of $\mathcal{G}$. We have:

Bounding the cost of a single recurrence call

- $\left(A, S^*_{\mathcal{G}}, B\right)$ in the above lemma is a vertex cut of $\mathcal{G}$. We have:

$$\begin{aligned}\alpha_{c,w}(\mathcal{G}) &\leq \alpha_{c,w}\left(A, S^*_{\mathcal{G}}, B\right) \\ &= \frac{c\left(S^*_{\mathcal{G}}\right)}{w\left(A \cup S^*_{\mathcal{G}}\right) \cdot w\left(B \cup S^*_{\mathcal{G}}\right)} \leq \frac{\lambda \cdot c\left(S^*_{\mathcal{G}}\right)}{w(\mathcal{G})^2}.\end{aligned}$$

Bounding the cost of a single recurrence call

- $\left(A, S^*_{\mathcal{G}}, B\right)$ in the above lemma is a vertex cut of $\mathcal{G}$. We have:

$$\begin{aligned}\alpha_{c,w}(\mathcal{G}) &\leq \alpha_{c,w}\left(A, S^*_{\mathcal{G}}, B\right) \\ &= \frac{c\left(S^*_{\mathcal{G}}\right)}{w\left(A \cup S^*_{\mathcal{G}}\right) \cdot w\left(B \cup S^*_{\mathcal{G}}\right)} \leq \frac{\lambda \cdot c\left(S^*_{\mathcal{G}}\right)}{w(\mathcal{G})^2}.\end{aligned}$$

- Let $(A_{\mathcal{G}}, S_{\mathcal{G}}, B_{\mathcal{G}}) = \texttt{AlgorithmMinCut}(\mathcal{G}, c, w)$, assume without loss of generality that $w(A_{\mathcal{G}}) \geq w(B_{\mathcal{G}})$. We have:

Bounding the cost of a single recurrence call

- $\left(A, S^*_{\mathcal{G}}, B\right)$ in the above lemma is a vertex cut of $\mathcal{G}$. We have:

$$\alpha_{c,w}(\mathcal{G}) \leq \alpha_{c,w}\left(A, S^*_{\mathcal{G}}, B\right)$$
$$= \frac{c\left(S^*_{\mathcal{G}}\right)}{w\left(A \cup S^*_{\mathcal{G}}\right) \cdot w\left(B \cup S^*_{\mathcal{G}}\right)} \leq \frac{\lambda \cdot c\left(S^*_{\mathcal{G}}\right)}{w(\mathcal{G})^2}.$$

- Let $(A_{\mathcal{G}}, S_{\mathcal{G}}, B_{\mathcal{G}}) = \texttt{AlgorithmMinCut}(\mathcal{G}, c, w)$, assume without loss of generality that $w(A_{\mathcal{G}}) \geq w(B_{\mathcal{G}})$. We have:

$$\alpha_{c,w}(A_{\mathcal{G}}, S_{\mathcal{G}}, B_{\mathcal{G}}) = \frac{c(S_{\mathcal{G}})}{w(A_{\mathcal{G}} \cup S_{\mathcal{G}}) \cdot w(B_{\mathcal{G}} \cup S_{\mathcal{G}})} \leq f_n \cdot \frac{\lambda \cdot c\left(S^*_{\mathcal{G}}\right)}{w(\mathcal{G})^2}.$$

Bounding the cost of a single recurrence call

- Let $\beta = w(B_{\mathcal{G}} \cup S_{\mathcal{G}}) \,/\, w(\mathcal{G})$.

Bounding the cost of a single recurrence call

- Let $\beta = w(B_{\mathcal{G}} \cup S_{\mathcal{G}}) / w(\mathcal{G})$.
- $(1 - \beta) \cdot w(\mathcal{G}) = w(A_{\mathcal{G}})$. We have:

$$
\begin{aligned}
w(\mathcal{G}) \cdot c(S_{\mathcal{G}}) &\le \lambda \cdot f_n \cdot \frac{w(A_{\mathcal{G}} \cup S_{\mathcal{G}}) \cdot w(B_{\mathcal{G}} \cup S_{\mathcal{G}})}{w(\mathcal{G})} \cdot c\left(S^*_{\mathcal{G}}\right) \\
&\le \lambda \cdot f_n \cdot w(B_{\mathcal{G}} \cup S_{\mathcal{G}}) \cdot c\left(S^*_{\mathcal{G}}\right) \\
&\le \lambda \cdot f_n \cdot \sum_{k=w(A_{\mathcal{G}})+1}^{w(\mathcal{G})} c\left(S^*_{\lfloor k/2 \rfloor} \cap \mathcal{G}\right).
\end{aligned}
$$

Bounding the cost of the solution

- D - the decision tree returned by DecisionTree(T, c, w).

Bounding the cost of the solution

- D - the decision tree returned by $\mathrm{DecisionTree}(T, c, w)$.
- We have:

$$
\begin{aligned}
c_G(D) &\leq \sum_{\mathcal{G}} w(\mathcal{G}) \cdot c(S_{\mathcal{G}}) \\
&\leq \lambda \cdot f_n \cdot \sum_{\mathcal{G}} \sum_{k=w(A_{\mathcal{G}})+1}^{w(\mathcal{G})} c\left(S^*_{\lfloor k/2 \rfloor} \cap \mathcal{G}\right) \\
&\leq \lambda \cdot f_n \cdot \sum_{k=0}^{w(G)} c\left(S^*_{\lfloor k/2 \rfloor}\right) \\
&\leq 2 \cdot \lambda \cdot f_n \cdot \mathtt{OPT}(G) = \left(12 + 4\sqrt{5}\right) \cdot f_n \cdot \mathtt{OPT}(G)\,.
\end{aligned}
$$

Thank you for your attention!

Questions?

Bibliography I

[Ang18] Haris Angelidakis. "Shortest path queries, graph partitioning and covering problems in worst and beyond worst case settings". In: *ArXiv* abs/1807.09389 (2018). URL: https://api.semanticscholar.org/CorpusID:51718679.

[BK22] Benjamin Berendsohn and László Kozma. "Splay trees on trees". In: Jan. 2022, pp. 1875–1900. ISBN: 978-1-61197-707-3. DOI: 10.1137/1.9781611977073.75.

[Ber+22] Benjamin Berendsohn et al. *Fast approximation of search trees on trees with centroid trees*. Sept. 2022. DOI: 10.48550/arXiv.2209.08024.

[BDO23] Piotr Borowiecki, Dariusz Dereniowski, and Dorota Osula. "The complexity of bicriteria tree-depth". In: *Theoretical Computer Science* 947 (2023), p. 113682. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2022.12.032. URL: https://www.sciencedirect.com/science/article/pii/S0304397522007666.

[CC17] Moses Charikar and Vaggos Chatziafratis. "Approximate hierarchical clustering via sparsest cut and spreading metrics". In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '17. Barcelona, Spain: Society for Industrial and Applied Mathematics, 2017, pp. 841–854.

[Cic+12] Ferdinando Cicalese et al. "The binary identification problem for weighted trees". In: *Theoretical Computer Science* 459 (2012), pp. 100–112. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2012.06.023.

[Cic+14] Ferdinando Cicalese et al. "Improved Approximation Algorithms for the Average-Case Tree Searching Problem". In: *Algorithmica* 68 (Apr. 2014). DOI: 10.1007/s00453-012-9715-6.

Bibliography II

[Cic+16] Ferdinando Cicalese et al. "On the tree search problem with non-uniform costs". In: *Theoretical Computer Science* 647 (2016), pp. 22–32. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2016.07.019.

[Coh+19] Vincent Cohen-addad et al. "Hierarchical Clustering: Objective Functions and Algorithms". In: *J. ACM* 66.4 (June 2019). ISSN: 0004-5411. DOI: 10.1145/3321386. URL: https://doi.org/10.1145/3321386.

[Das16] Sanjoy Dasgupta. "A cost function for similarity-based hierarchical clustering". In: *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '16. Cambridge, MA, USA: Association for Computing Machinery, 2016, pp. 118–127. ISBN: 9781450341325. DOI: 10.1145/2897518.2897527. URL: https://doi.org/10.1145/2897518.2897527.

[DMS19] Argyrios Deligkas, George B. Mertzios, and Paul G. Spirakis. "Binary Search in Graphs Revisited". In: *Algorithmica* 81.5 (May 2019), pp. 1757–1780. ISSN: 1432-0541. DOI: 10.1007/s00453-018-0501-y.

[Der06] Dariusz Dereniowski. "Edge ranking of weighted trees". In: *Discrete Applied Mathematics* 154.8 (2006), pp. 1198–1209. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2005.11.005.

[Der08] Dariusz Dereniowski. "Edge ranking and searching in partial orders". In: *Discrete Applied Mathematics* 156.13 (2008). Fifth International Conference on Graphs and Optimization, pp. 2493–2500. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2008.03.007.

[DGP23] Dariusz Dereniowski, Przemysław Gordinowicz, and Paweł Prałat. "Edge and Pair Queries—Random Graphs and Complexity". In: *The Electronic Journal of Combinatorics* 30.2 (2023). DOI: 10.37236/11159. URL: https://www.combinatorics.org/ojs/index.php/eljc/article/view/v30i2p34.

Bibliography III

[DGW24] Dariusz Dereniowski, Przemysław Gordinowicz, and Karolina Wr'obel. "On multidimensional generalization of binary search". In: *ArXiv* abs/2404.13193 (2024). URL: https://api.semanticscholar.org/CorpusID:269293685.

[DK06] Dariusz Dereniowski and Marek Kubale. "Efficient Parallel Query Processing by Graph Ranking". In: *Fundam. Inform.* 69 (Feb. 2006), pp. 273–285. DOI: 10.3233/FUN-2006-69302.

[DŁU21] Dariusz Dereniowski, Aleksander Łukasiewicz, and Przemysław Uznański. "An Efficient Noisy Binary Search in Graphs via Median Approximation". In: *Combinatorial Algorithms*. Ed. by Paola Flocchini and Lucia Moura. Cham: Springer International Publishing, 2021, pp. 265–281. ISBN: 978-3-030-79987-8.

[DŁU25] Dariusz Dereniowski, Aleksander Łukasiewicz, and Przemysław Uznański. "Noisy (Binary) Searching: Simple, Fast and Correct". In: *42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025)*. Ed. by Olaf Beyersdorff et al. Vol. 327. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025, 29:1–29:18. ISBN: 978-3-95977-365-2. DOI: 10.4230/LIPIcs.STACS.2025.29. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.STACS.2025.29.

[DN06] Dariusz Dereniowski and Adam Nadolski. "Vertex rankings of chordal graphs and weighted trees". In: *Information Processing Letters* 98.3 (2006), pp. 96–100. ISSN: 0020-0190. DOI: https://doi.org/10.1016/j.ipl.2005.12.006.

Bibliography IV

[DW22] Dariusz Dereniowski and Izajasz Wrosz. "Constant-Factor Approximation Algorithm for Binary Search in Trees with Monotonic Query Times". In: *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*. Ed. by Stefan Szeider, Robert Ganian, and Alexandra Silva. Vol. 241. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 42:1–42:15. ISBN: 978-3-95977-256-3. DOI: 10.4230/LIPIcs.MFCS.2022.42.

[DW24] Dariusz Dereniowski and Izajasz Wrosz. *Searching in trees with monotonic query times*. 2024. arXiv: 2401.13747 [cs.DS]. URL: https://arxiv.org/abs/2401.13747.

[Der+17] Dariusz Dereniowski et al. "Approximation Strategies for Generalized Binary Search in Weighted Trees". In: *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*. Ed. by Ioannis Chatzigiannakis et al. Vol. 80. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017, 84:1–84:14. ISBN: 978-3-95977-041-5. DOI: 10.4230/LIPIcs.ICALP.2017.84.

[EKS16] Ehsan Emamjomeh-Zadeh, David Kempe, and Vikrant Singhal. "Deterministic and probabilistic binary search in graphs". In: June 2016, pp. 519–532. DOI: 10.1145/2897518.2897656.

[FHL05] Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. "Improved approximation algorithms for minimum-weight vertex separators". In: *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*. STOC '05. Baltimore, MD, USA: Association for Computing Machinery, 2005, pp. 563–572. ISBN: 1581139608. DOI: 10.1145/1060590.1060674. URL: https://doi.org/10.1145/1060590.1060674.

Bibliography V

[GHT12] Archontia C. Giannopoulou, Paul Hunter, and Dimitrios M. Thilikos. "LIFO-search: A min–max theorem and a searching game for cycle-rank and tree-depth". In: *Discrete Applied Mathematics* 160.15 (2012), pp. 2089–2097. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2012.03.015. URL: https://www.sciencedirect.com/science/article/pii/S0166218X12001199.

[Høg24] Svein Høgemo. "Tight Approximation Bounds on a Simple Algorithm for Minimum Average Search Time in Trees". In: *ArXiv* abs/2402.05560 (2024). URL: https://api.semanticscholar.org/CorpusID:267547530.

[Jac+10] Tobias Jacobs et al. "On the Complexity of Searching in Trees: Average-Case Minimization". In: *Automata, Languages and Programming*. Ed. by Samson Abramsky et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 527–539. ISBN: 978-3-642-14165-2.

[KMS95] Meir Katchalski, William McCuaig, and Suzanne Seager. "Ordered colourings". In: *Discrete Mathematics* 142.1 (1995), pp. 141–154. ISSN: 0012-365X. DOI: https://doi.org/10.1016/0012-365X(93)E0216-Q. URL: https://www.sciencedirect.com/science/article/pii/0012365X93E0216Q.

[Knu73] Donald Knuth. *The Art Of Computer Programming, vol. 3: Sorting And Searching*. Addison-Wesley, 1973, pp. 391–392.

[LY98] Tak Wah Lam and Fung Ling Yue. "Edge ranking of graphs is hard". In: *Discrete Applied Mathematics* 85.1 (1998), pp. 71–86. ISSN: 0166-218X. DOI: https://doi.org/10.1016/S0166-218X(98)00029-8.

Bibliography VI

[NO06] Jaroslav Nešetřil and Patrice Ossona de Mendez. "Tree-depth, subgraph coloring and homomorphism bounds". In: *European Journal of Combinatorics* 27.6 (2006), pp. 1022–1041. ISSN: 0195-6698. DOI: https://doi.org/10.1016/j.ejc.2005.01.010. URL: https://www.sciencedirect.com/science/article/pii/S0195669805000570.

[OP06] Krzysztof Onak and Pawel Parys. "Generalization of Binary Search: Searching in Trees and Forest-Like Partial Orders". In: *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. 2006, pp. 379–388. DOI: 10.1109/FOCS.2006.32.

[Pot88] Alex Pothen. *The Complexity of Optimal Elimination Trees*. Technical Report CS-88-16. Penn State University CS-88-16. Pennsylvania State University, Department of Computer Science, Apr. 1988.