Hierarchical Clustering: Objective Functions and Algorithms

Arne Nix

arne.nix@rwth-aachen.de

July 19, 2018, Aachen

Seminar: Theoretical Topics in Data Science 2
Lehrstuhl für Informatik 7, RWTH Aachen University

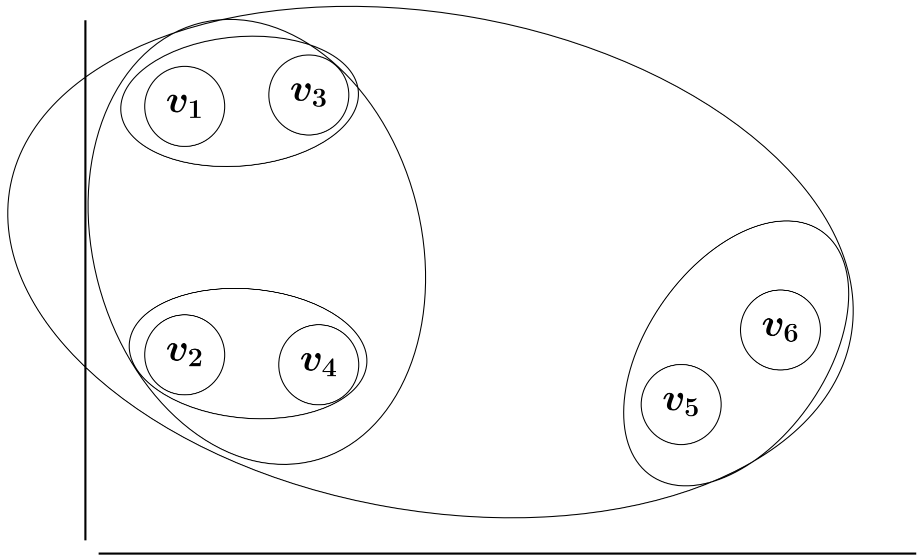
Outline

RWTH AACHEN UNIVERSITY

Motivation

Advantages of hierarchical clustering:

- **No need to specify number of clusters in advance.**
- **Cluster structure captured on all levels of granularity.**
- **Output: Binary cluster tree T with leaves $\mathcal{L}$ and internal nodes $\mathcal{N}$**
- Input: Similarity graph $G = (V,E,w)$
  - $w_{ij} := w(v_i,v_j)$ defines similarity between vertices $v_i,v_j \in V$



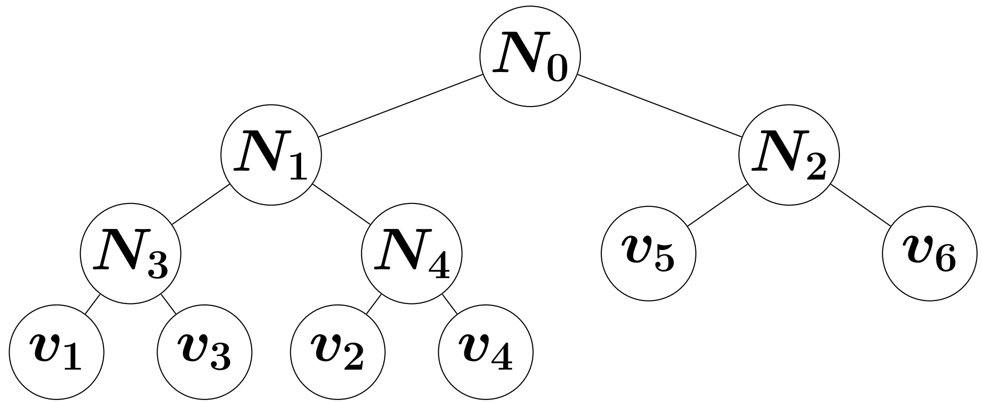


Figure: Example of a clustering and the corresponding cluster tree.

Motivation

Advantages of hierarchical clustering:

- **No need to specify number of clusters in advance.**
- **Cluster structure captured on all levels of granularity.**
- **Output: Binary cluster tree T with leaves $\mathcal{L}$ and internal nodes $\mathcal{N}$**
- **Input: Similarity graph $G = (V, E, w)$**
  - $w_{ij} := w(v_i, v_j)$ **defines similarity between vertices** $v_i, v_j \in V$

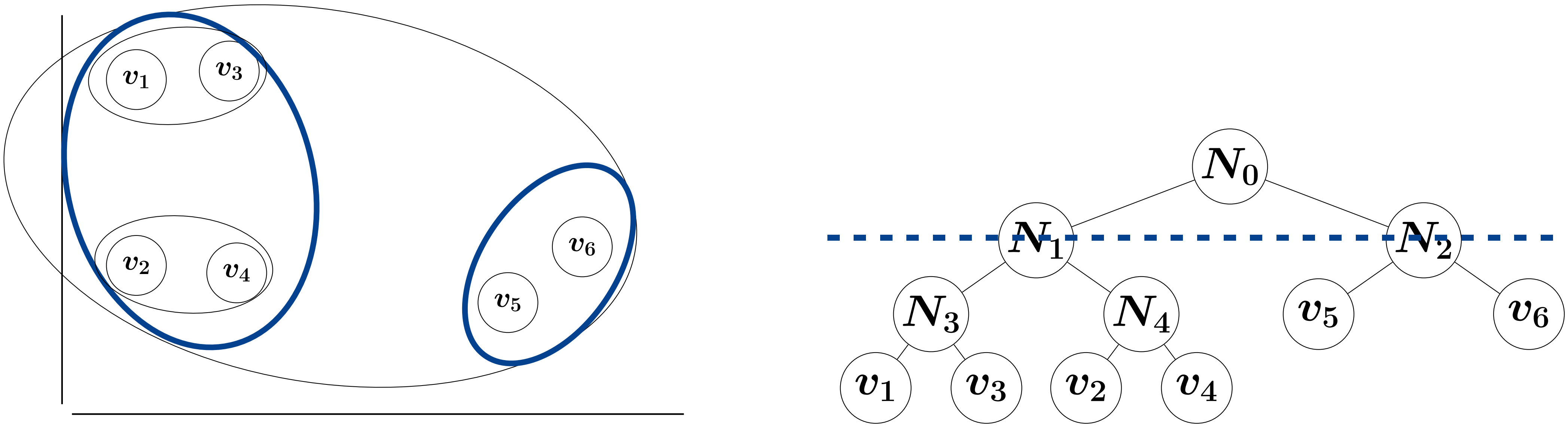


Figure: Example of a clustering and the corresponding cluster tree.

Motivation

Advantages of hierarchical clustering:

- No need to specify number of clusters in advance.
- Cluster structure captured on all levels of granularity.
- **Output:** Binary cluster tree T with leaves $\mathcal{L}$ and internal nodes $\mathcal{N}$
- **Input:** Similarity graph $G = (V,E,w)$
  - $w_{ij} := w(v_i,v_j)$ defines similarity between vertices $v_i,v_j \in V$

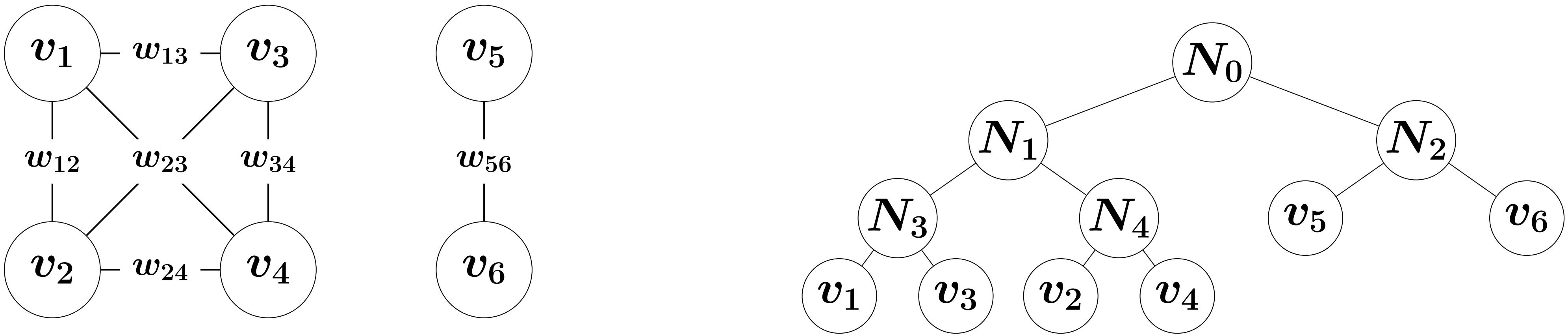


Figure: Example of a similarity graph and the corresponding cluster tree.

Literature

S. Dasgupta [Dasgupta 16]

A cost function for similarity-based hierarchical clustering

- **Hierarchical clustering as combinatorial optimization problem.**
- **Introduces simple cost function.**
- **Proposes Recursive ϕ-Sparsest-Cut clustering algorithm.**

V. Cohen-Addad, V. Kanade et al. [Cohen-Addad & Kanade^{+} 18]:

Hierarchical Clustering: Objective Functions and Algorithms

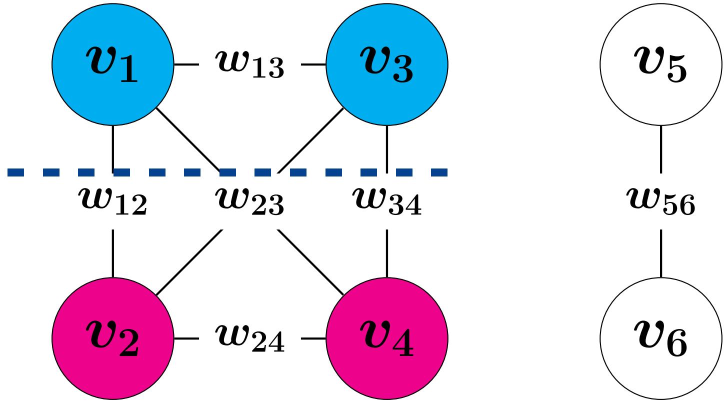
- **Continue to formalize hierarchical clustering.**
- **Develop framework for cost functions in hierarchical clustering.**
- **Formally define admissible (“good”) cost functions.**
- **Propose and prove new clustering algorithms.**

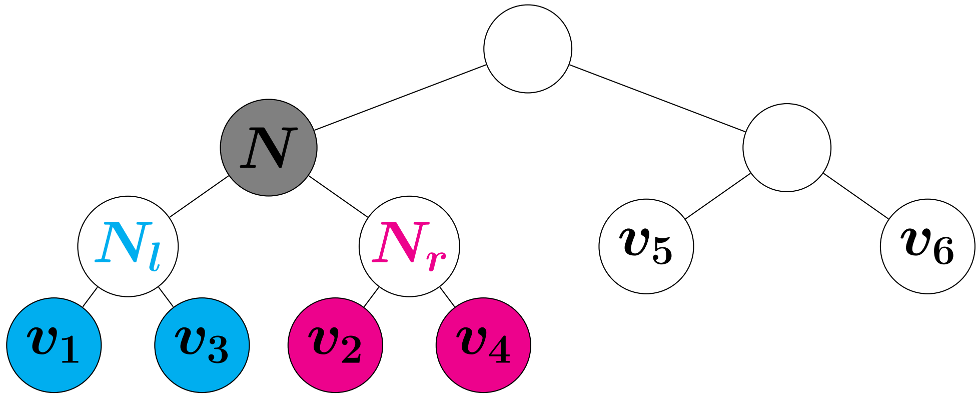
Outline

Cost Function for Cluster Trees

- Most hierarchical clustering algorithms defined procedurally
  - No underlying objective function
  - Output not defined precisely
- Advantages of cost functions:
  - Precise definition possible
  - Study complexity
  - Compare efficiency
  - Incorporating constraints or prior information gets simpler

Cost Function for Cluster Trees





Definition (Cost Function)

For cluster tree T:

$$\Gamma(T) = \sum_{N \in \mathcal{N}} \gamma(N)$$

For $N \in \mathcal{N}$:

$$\gamma(N) = \sum_{u \in L(N_l), v \in L(N_r)} w(u,v) \cdot g(|L(N_l)|,|L(N_r)|)$$

where

- N_l **and** N_r**: left and right children of** N
- $L(N)$**: leaves of the subtree induced by** N

Cost Function: Example

$$\gamma(N) = \sum_{u \in L(N_l), v \in L(N_r)} w(u,v) \cdot g(|L(N_l)|,|L(N_r)|)$$

- Idea: Perform expensive cuts as far down as possible.
- Cost function proposed by [Dasgupta 16]:
  - $g(a,b) := a + b \to$ total number of leaves below N

Figure: Example of clustering using cost function $\tilde{g}$ and unit weights.

Cost Function: Example

$$\gamma(N) = \sum_{u \in L(N_l), v \in L(N_r)} w(u,v) \cdot (|L(N_l)| + |L(N_r)|)$$

- **Idea: Perform expensive cuts as far down as possible.**
- **Cost function proposed by [Dasgupta 16]:**
  - $g(a,b) := a + b \rightarrow$ **total number of leaves below** N

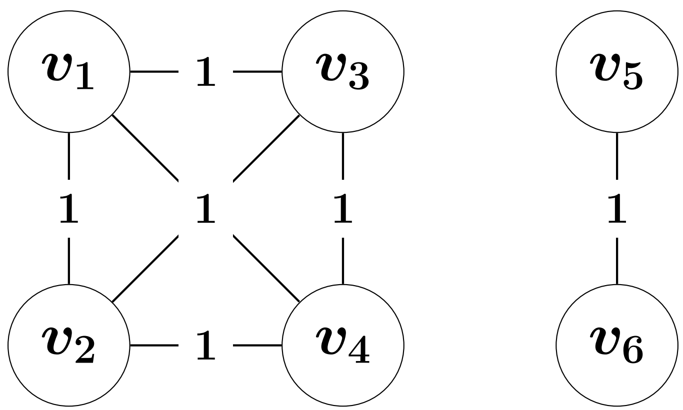


Figure: **Example of clustering using cost function $\tilde{g}$ and unit weights.**

Cost Function: Example

$$\gamma(N) = \sum_{u \in L(N_\mathsf{l}), v \in L(N_\mathsf{r})} w(u,v) \cdot (|L(N_\mathsf{l})| + |L(N_\mathsf{r})|)$$

- **Idea: Perform expensive cuts as far down as possible.**
- **Cost function proposed by [Dasgupta 16]:**
  - $g(a,b) := a + b \rightarrow$ **total number of leaves below** N

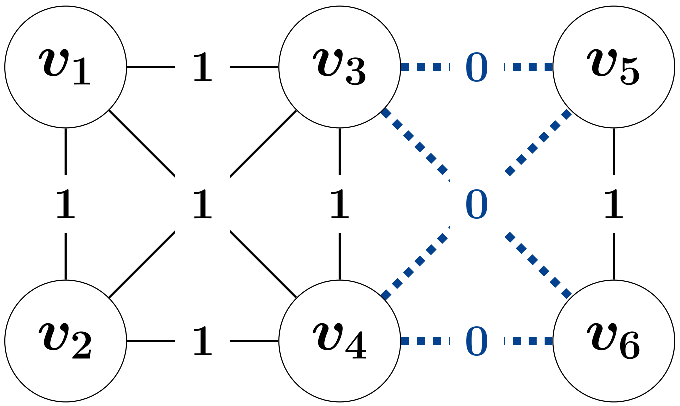
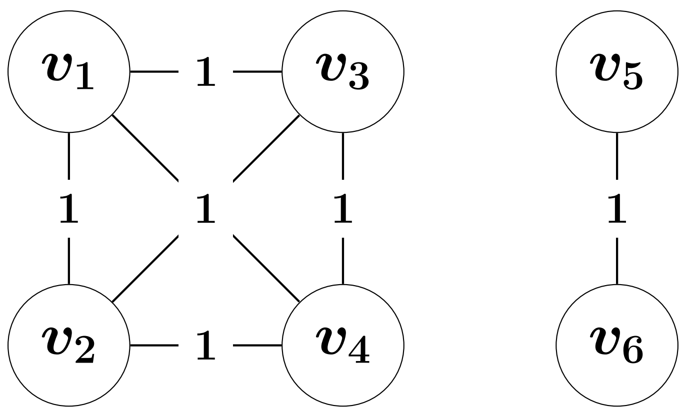


Figure: Example of clustering using cost function $\tilde{g}$ and unit weights.

Cost Function: Example

$$\gamma(N) = \sum_{u \in L(N_\mathsf{l}), v \in L(N_\mathsf{r})} w(u,v) \cdot (|L(N_\mathsf{l})| + |L(N_\mathsf{r})|)$$

- **Idea: Perform expensive cuts as far down as possible.**
- **Cost function proposed by [Dasgupta 16]:**
  - $g(a,b) := a + b \rightarrow$ **total number of leaves below** N



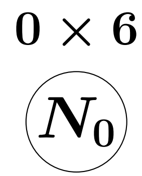
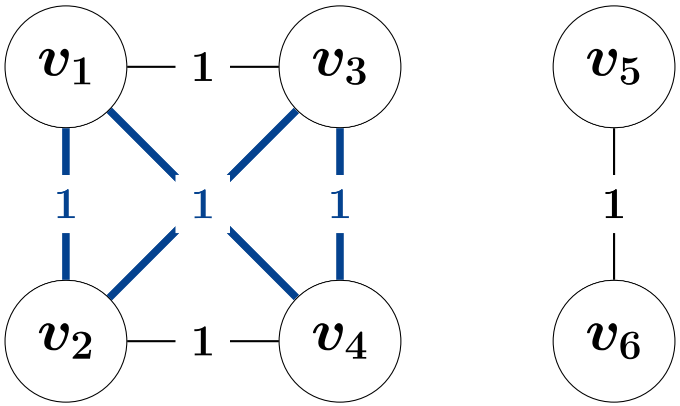


Figure: Example of clustering using cost function $\tilde{g}$ and unit weights.

Cost Function: Example

$$\gamma(N) = \sum_{u \in L(N_{\mathsf{l}}), v \in L(N_{\mathsf{r}})} w(u,v) \cdot (|L(N_{\mathsf{l}})| + |L(N_{\mathsf{r}})|)$$

- **Idea: Perform expensive cuts as far down as possible.**
- **Cost function proposed by [Dasgupta 16]:**
  - $g(a,b) := a + b \rightarrow$ **total number of leaves below** N



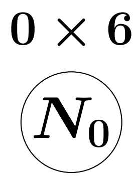


Figure: Example of clustering using cost function $\tilde{g}$ and unit weights.

Cost Function: Example

$$\gamma(N) = \sum_{u \in L(N_{\mathsf{l}}), v \in L(N_{\mathsf{r}})} w(u,v) \cdot (|L(N_{\mathsf{l}})| + |L(N_{\mathsf{r}})|)$$

- **Idea: Perform expensive cuts as far down as possible.**
- **Cost function proposed by [Dasgupta 16]:**
  - $g(a,b) := a + b \rightarrow$ **total number of leaves below** N

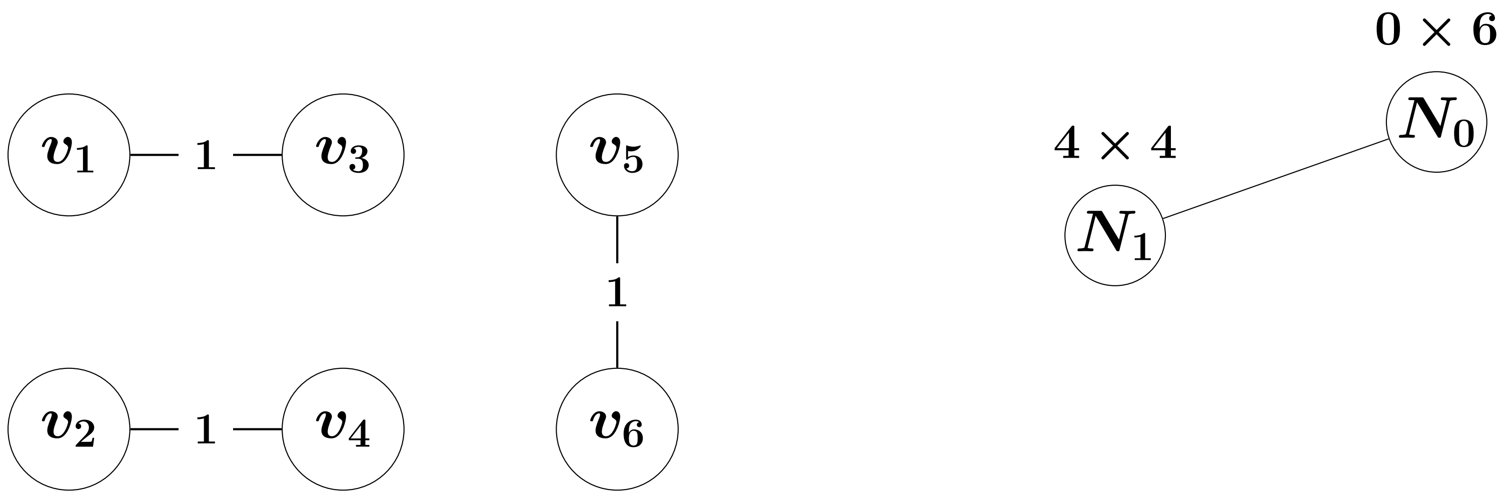


Figure: **Example of clustering using cost function $\tilde{g}$ and unit weights.**

Cost Function: Example

$$\gamma(N) = \sum_{u \in L(N_l), v \in L(N_r)} w(u,v) \cdot (|L(N_l)| + |L(N_r)|)$$

- **Idea: Perform expensive cuts as far down as possible.**
- **Cost function proposed by [Dasgupta 16]:**
  - $g(a,b) := a + b \rightarrow$ **total number of leaves below** N

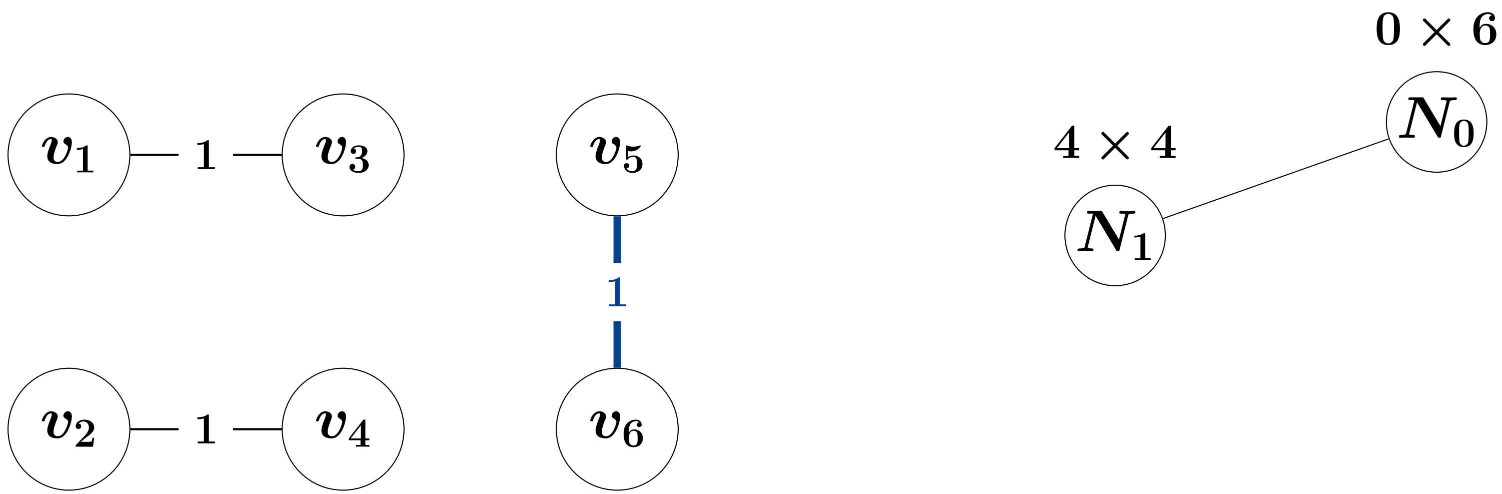


Figure: **Example of clustering using cost function $\tilde{g}$ and unit weights.**

RWTH AACHEN UNIVERSITY

Cost Function: Example

$$\gamma(N) = \sum_{u \in L(N_\mathsf{l}), v \in L(N_\mathsf{r})} w(u,v) \cdot (|L(N_\mathsf{l})| + |L(N_\mathsf{r})|)$$

- **Idea: Perform expensive cuts as far down as possible.**
- **Cost function proposed by [Dasgupta 16]:**
  - $g(a,b) := a + b \rightarrow$ **total number of leaves below** N

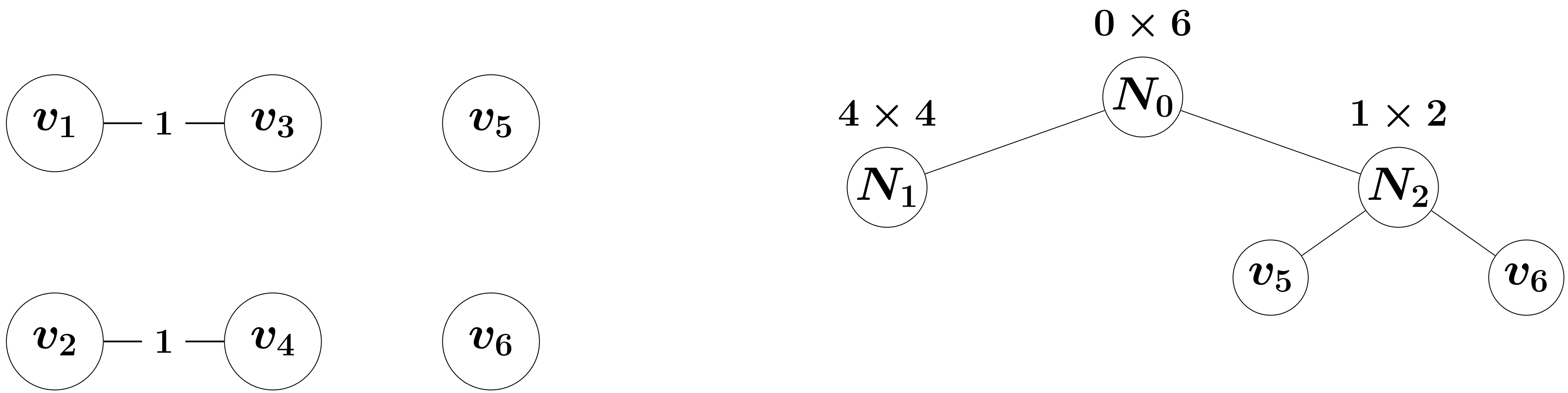


Figure: **Example of clustering using cost function $\tilde{g}$ and unit weights.**

Cost Function: Example

$$\gamma(N) = \sum_{u \in L(N_\mathsf{l}), v \in L(N_\mathsf{r})} w(u,v) \cdot (|L(N_\mathsf{l})| + |L(N_\mathsf{r})|)$$

- **Idea: Perform expensive cuts as far down as possible.**
- **Cost function proposed by [Dasgupta 16]:**
  - $g(a,b) := a + b \rightarrow$ **total number of leaves below** N

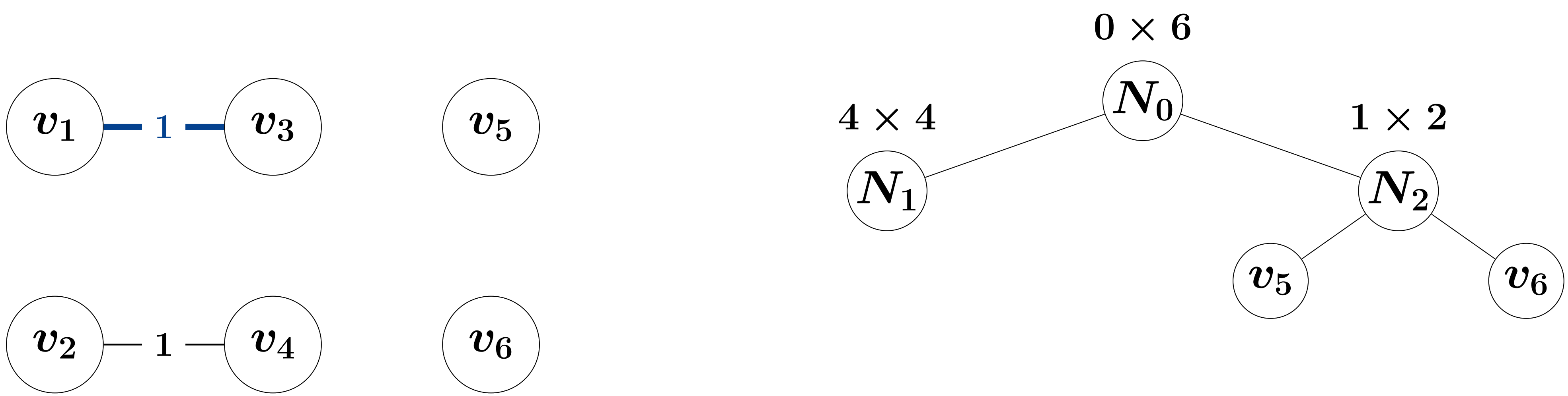


Figure: **Example of clustering using cost function $\tilde{g}$ and unit weights.**

RWTH AACHEN UNIVERSITY

Cost Function: Example

$$\gamma(N) = \sum_{u \in L(N_l), v \in L(N_r)} w(u,v) \cdot (|L(N_l)| + |L(N_r)|)$$

- **Idea: Perform expensive cuts as far down as possible.**
- **Cost function proposed by [Dasgupta 16]:**
  - $g(a,b) := a + b \rightarrow$ **total number of leaves below N**

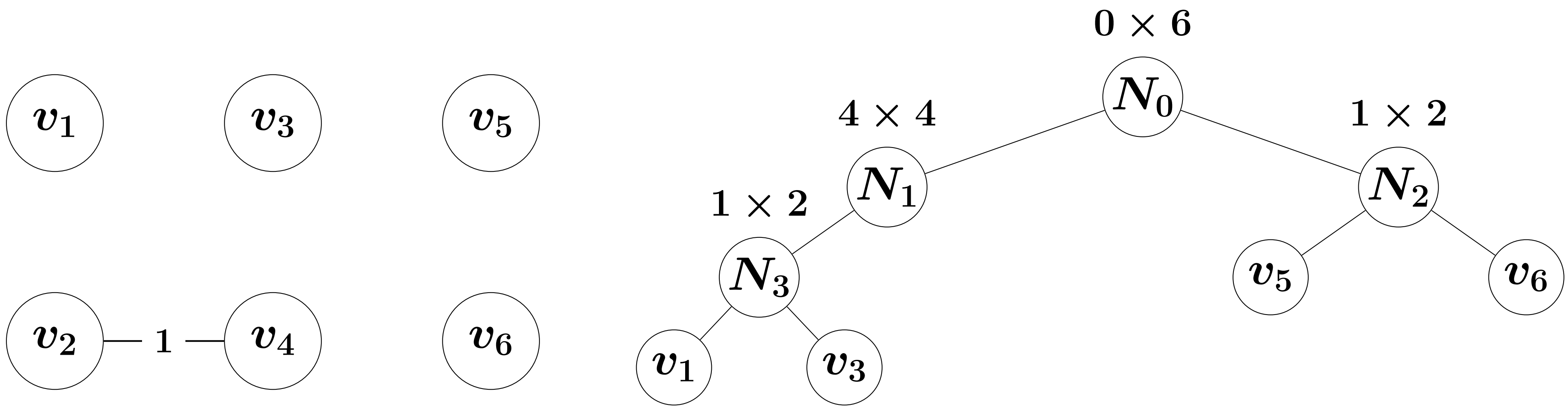


Figure: **Example of clustering using cost function $\tilde{g}$ and unit weights.**

Cost Function: Example

$$\gamma(N) = \sum_{u \in L(N_\mathsf{l}), v \in L(N_\mathsf{r})} w(u,v) \cdot (|L(N_\mathsf{l})| + |L(N_\mathsf{r})|)$$

- **Idea: Perform expensive cuts as far down as possible.**
- **Cost function proposed by [Dasgupta 16]:**
  - $g(a,b) := a + b \rightarrow$ **total number of leaves below** N

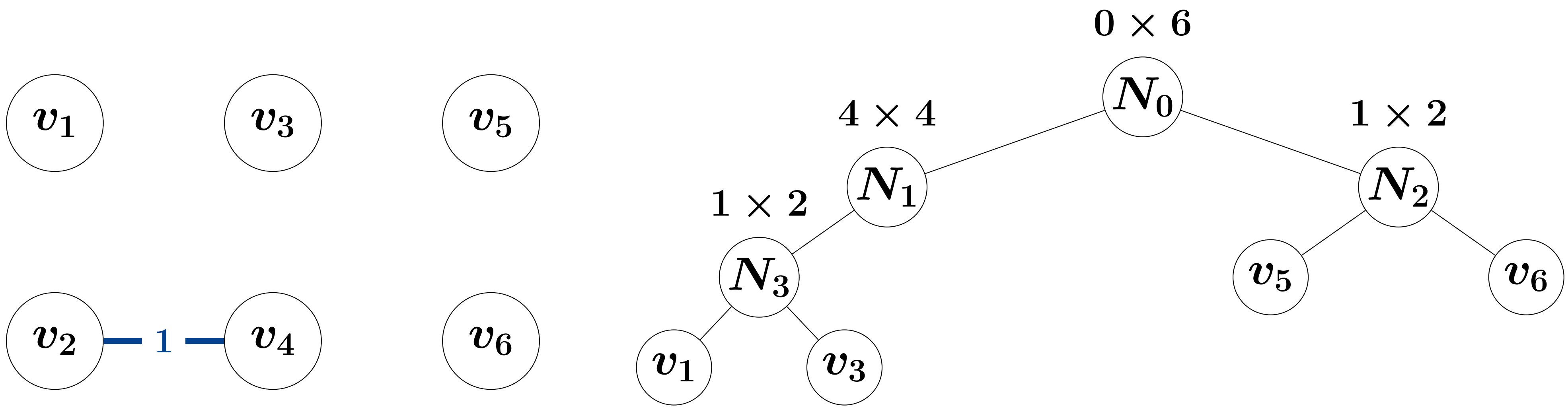


Figure: **Example of clustering using cost function $\tilde{g}$ and unit weights.**

Cost Function: Example

$$\gamma(N) = \sum_{u \in L(N_\mathsf{l}), v \in L(N_\mathsf{r})} w(u,v) \cdot (|L(N_\mathsf{l})| + |L(N_\mathsf{r})|)$$

- **Idea: Perform expensive cuts as far down as possible.**
- **Cost function proposed by [Dasgupta 16]:**
  - $g(a,b) := a + b \rightarrow$ **total number of leaves below** N

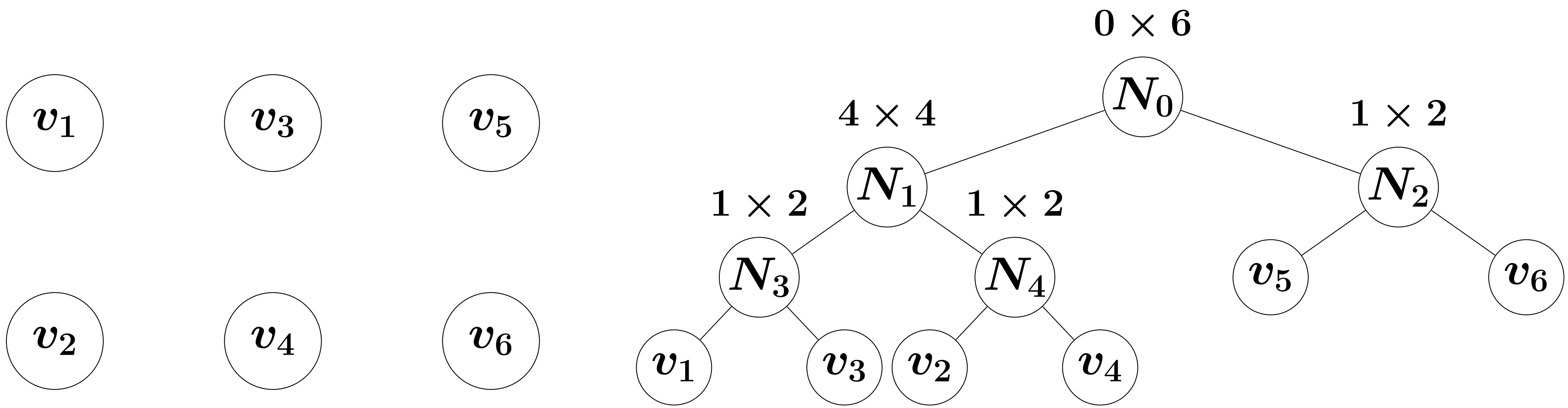


Figure: **Example of clustering using cost function $\tilde{g}$ and unit weights.**

Outline

Introduction

Objective Function for Hierarchical Clustering

Admissible Cost Function

Algorithms

Conclusion and Discussion

Ultrametrics

Definition (Ultrametric)

An ultrametric is a metric space (X,d) with

$$d(x,y) \leq \max\{d(x,z),d(y,z)\} \; \forall x,y,z \in X$$

$\rightarrow$ Isosceles triangles with two sides longer than one.

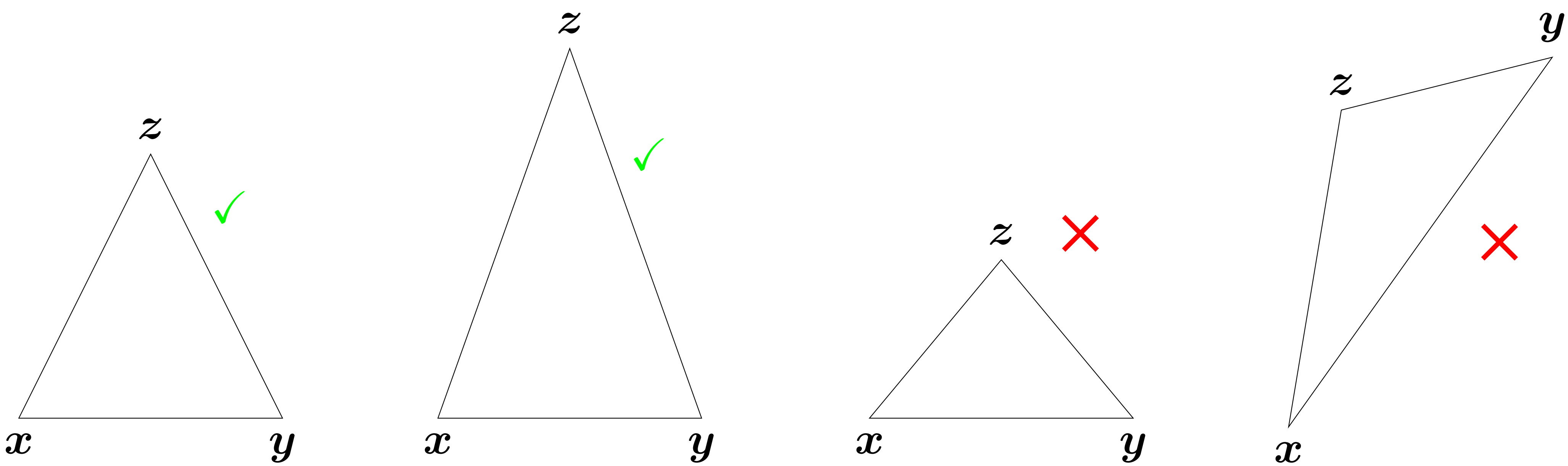


Figure: **The triangles 3 and 4 violate the condition $d(x,y) \leq \max\{d(x,z),d(y,z)\}$**

Graph Generated By Ultrametric

- Similarity graph $G = (V,E,w)$
- G generated from ultrametric (X,d) if:
  - $V \subseteq X$ and
  - $w(u,v) = f(d(u,v))$ for every $u,v \in V$ with $u \neq v$
  - $f : \mathbb{R}_+ \to \mathbb{R}_+$ non-increasing function
    - e.g.: $f(x) = \frac{1}{x}$
- G is then called ground-truth input

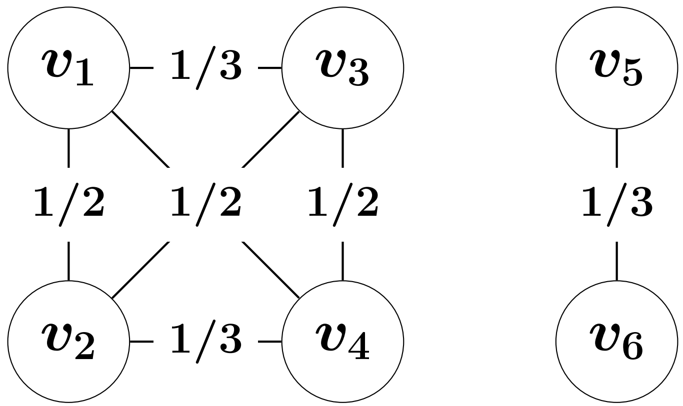


Figure: Example of a graph generated by an ultrametric.

Graph Generated By Ultrametric

- Similarity graph $G = (V,E,w)$
- G generated from ultrametric (X,d) if:
  - $V \subseteq X$ and
  - $w(u,v) = f(d(u,v))$ for every $u,v \in V$ with $u \neq v$
  - $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ non-increasing function
    - e.g.: $f(x) = \frac{1}{x}$
- G is then called ground-truth input

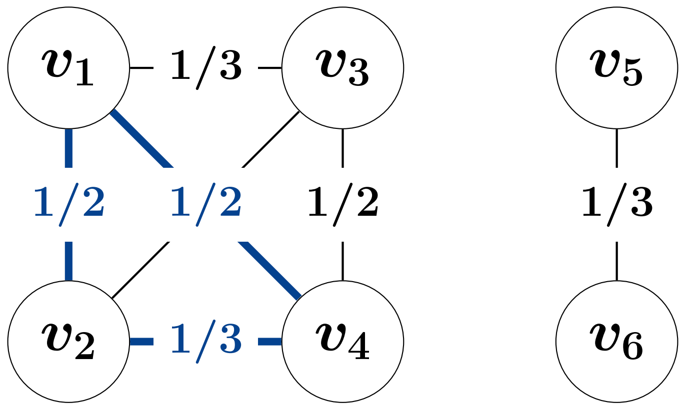


Figure: Example of a graph generated by an ultrametric.

Graph Generated By Ultrametric

- Similarity graph $G = (V,E,w)$
- G generated from ultrametric (X,d) if:
  - $V \subseteq X$ and
  - $w(u,v) = f(d(u,v))$ for every $u,v \in V$ with $u \neq v$
  - $f : \mathbb{R}_+ \to \mathbb{R}_+$ non-increasing function
    - e.g.: $f(x) = \frac{1}{x}$
- G is then called ground-truth input

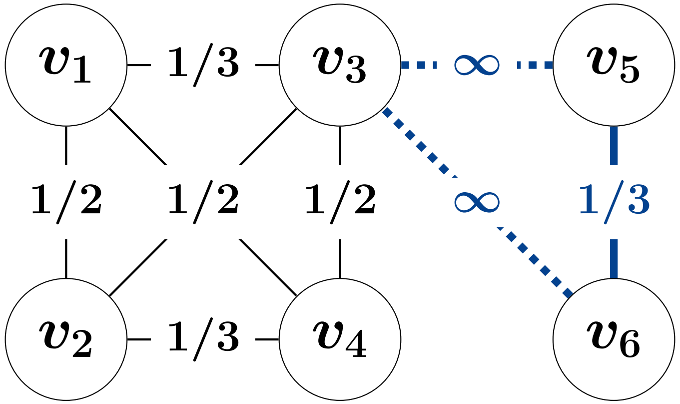


Figure: Example of a graph generated by an ultrametric.

Graph Generated By Ultrametric

- Similarity graph $G = (V,E,w)$
- G generated from ultrametric (X,d) if:
  - $V \subseteq X$ and
  - $w(u,v) = f(d(u,v))$ for every $u,v \in V$ with $u \neq v$
  - $f : \mathbb{R}_+ \to \mathbb{R}_+$ non-increasing function
    - e.g.: $f(x) = \frac{1}{x}$
- G is then called ground-truth input

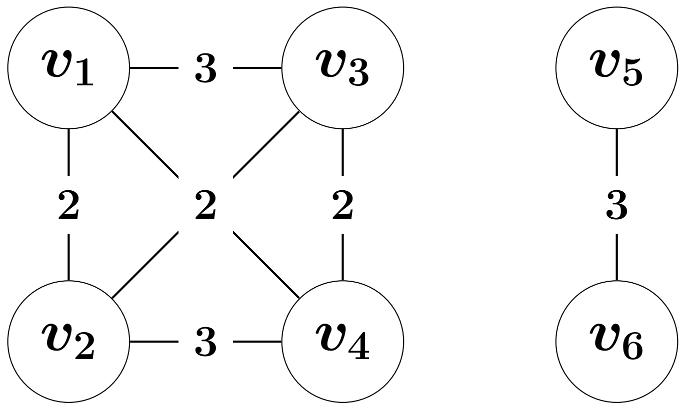


Figure: Example of a similarity graph generated by an ultrametric.

Generating Tree

Definition (Generating Tree)

- Rooted binary tree T with $|V|$ leaves $\mathcal{L}$ and $|V| - 1$ internal nodes $\mathcal{N}$
- Weight function $W : \mathcal{N} \to \mathbb{R}_+$ s.t.:
  - $W(N_i) \leq W(N_j)$ for $N_i, N_j \in \mathcal{N}$ if N_i ancestor of N_j
  - $w(v_i, v_j) = W(\text{LCA}_T(v_i, v_j))$ for every $v_i, v_j \in V$

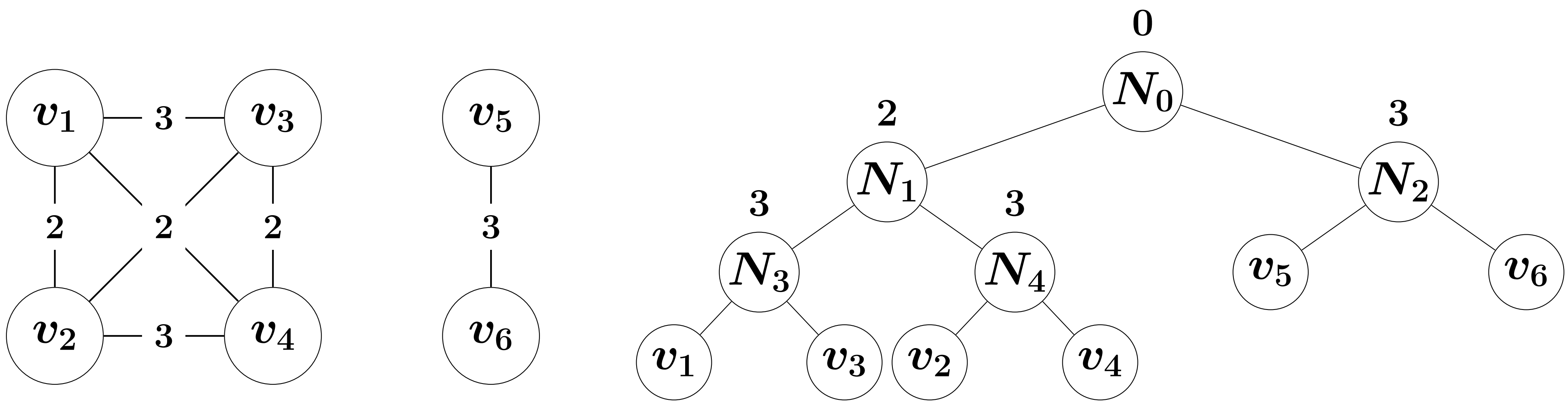


Figure: Similarity graph and corresponding generating tree with weights on every node.

Generating Tree

Definition (Generating Tree)

- **Rooted binary tree T with $|V|$ leaves $\mathcal{L}$ and $|V|-1$ internal nodes $\mathcal{N}$**
- **Weight function $W : \mathcal{N} \to \mathbb{R}_+$ s.t.:**
  - $W(N_i) \leq W(N_j)$ **for** $N_i, N_j \in \mathcal{N}$ **if** N_i **ancestor of** N_j
  - $w(v_i, v_j) = W(\text{LCA}_T(v_i, v_j))$ **for every** $v_i, v_j \in V$

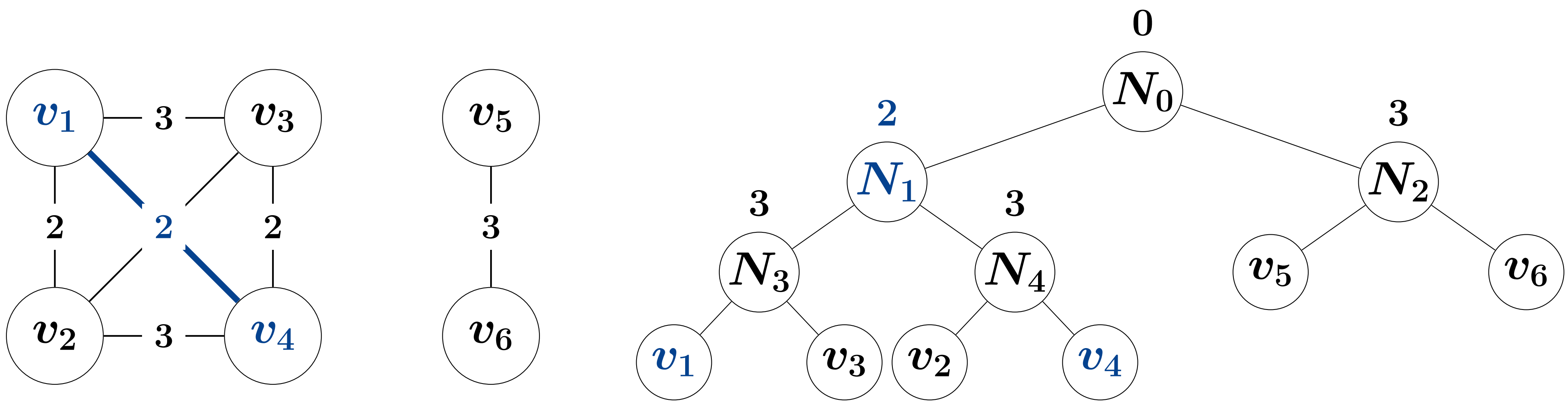


Figure: **Similarity graph and corresponding generating tree with weights on every node.**

Generating Tree

Definition (Generating Tree)

- Rooted binary tree T with $|V|$ leaves $\mathcal{L}$ and $|V|-1$ internal nodes $\mathcal{N}$
- Weight function $W : \mathcal{N} \to \mathbb{R}_+$ s.t.:
  - $W(N_i) \leq W(N_j)$ for $N_i, N_j \in \mathcal{N}$ if N_i ancestor of N_j
  - $w(v_i, v_j) = W(\mathrm{LCA}_T(v_i, v_j))$ for every $v_i, v_j \in V$

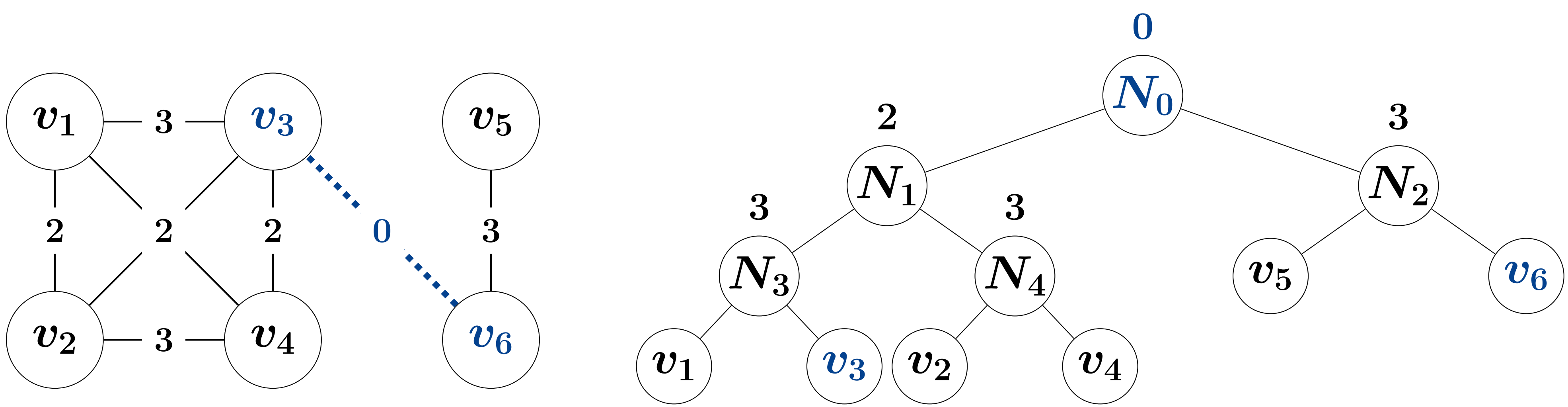


Figure: Similarity graph and corresponding generating tree with weights on every node.

Strict Generating Tree

Definition (Strict Generating Tree)

- Rooted binary tree T with $|V|$ leaves $\mathcal{L}$ and $|V| - 1$ internal nodes $\mathcal{N}$
- Weight function $W : \mathcal{N} \rightarrow \mathbb{R}_+$ s.t.:
  - $W(N_i) < W(N_j)$ for $N_i, N_j \in \mathcal{N}$ if N_i ancestor of N_j
  - $w(v_i, v_j) = W(\text{LCA}_T(v_i, v_j))$ for every $v_i, v_j \in V$

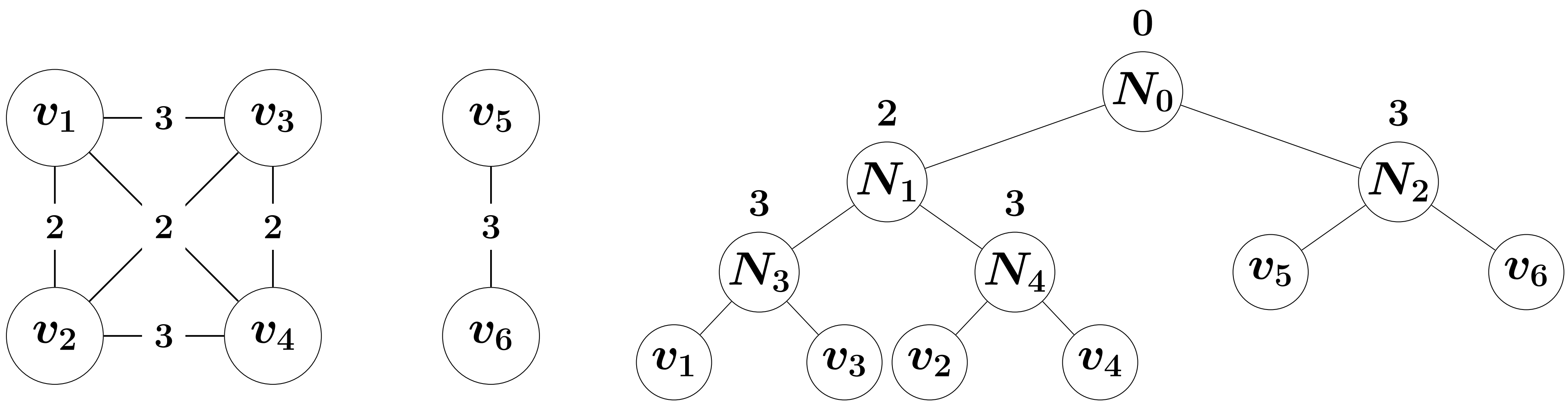


Figure: Similarity graph and corresponding generating tree with weights on every node.

Admissible Cost Function

$$\Gamma(T) = \sum_{N \in \mathcal{N}} \left(\sum_{u \in L(N_\mathsf{l}), v \in L(N_\mathsf{r})} w(u,v) \cdot g(|L(N_\mathsf{l})|, |L(N_\mathsf{r})|) \right)$$

Definition (Admissible Cost Function)

Γ admissible if a cluster tree T for G generated from an ultrametric, achieves minimum cost if and only if it is a generating tree for G.

Admissible Cost Function

$$\Gamma(T) = \sum_{N \in \mathcal{N}} \left(\sum_{u \in L(N_{\mathsf{l}}), v \in L(N_{\mathsf{r}})} w(u,v) \cdot g(|L(N_{\mathsf{l}})|, |L(N_{\mathsf{r}})|) \right)$$

Theorem

Γ ***admissible if and only if it satifies the follwoing conditions:***

1. ***Let*** G ***be a clique, i.e.*** $w(u,v) = 1 \; \forall u,v \in V$***:***
   $\Gamma(T)$ ***is identical for every cluster tree*** T ***of*** G
2. ***Symmetry:*** $g(a,b) = g(b,a)$ ***for every*** $a,b \in \mathbb{N}$
3. ***Monotonicity:*** $g(a+1,b) > g(a,b)$ ***for every*** $a,b \in \mathbb{N}$

Note: Dasgupta's cost function with $g(a,b) = a + b$ is admissible.

► Use this cost function for all following proofs and examples

Admissible Cost Function

$$\Gamma(T) = \sum_{N \in \mathcal{N}} \left(\sum_{u \in L(N_{\mathsf{l}}), v \in L(N_{\mathsf{r}})} w(u,v) \cdot g(|L(N_{\mathsf{l}})|, |L(N_{\mathsf{r}})|) \right)$$

Theorem

Γ ***admissible if and only if it satifies the follwoing conditions:***

1. ***Let*** G ***be a clique, i.e.*** $w(u,v) = 1\ \forall u,v \in V$***:***
   $\Gamma(T)$ ***is identical for every cluster tree*** T ***of*** G
2. ***Symmetry:*** $g(a,b) = g(b,a)$ ***for every*** $a,b \in \mathbb{N}$
3. ***Monotonicity:*** $g(a+1,b) > g(a,b)$ ***for every*** $a,b \in \mathbb{N}$

Note: Dasgupta's cost function with $g(a,b) = a + b$ **is admissible.**

- **Use this cost function for all following proofs and examples**

Outline

Introduction

Objective Function for Hierarchical Clustering

Algorithms

- Clustering Arbitrary Inputs
- Clustering Perfect Inputs

Conclusion and Discussion

Outline

Recursive ϕ-Sparsest-Cut Algorithm for Hierarchical Clustering

- **Idea: Minimize sparsity $sp(\{A,V\setminus A\}) := \frac{w(A,V\setminus A)}{|A||V\setminus A|}$ in each cut.**
  - **Approximation in $\mathcal{O}(\sqrt{\log(|V|)})$ possible [Arora & Rao$^+$ 09]**

1. Recursive ϕ-Sparsest-Cut Algorithm

1: **Input:** $G = (V,E,w)$
2: **Find** $\{A,V\setminus A\}$ **with**

$$sp(\{A,V\setminus A\}) \leq \phi \cdot \min_{S\subset V} sp(\{S,V\setminus S\})$$

3: **Repeat recursively on $G[A]$ and $G[V\setminus A]$ to obtain trees T_A and $T_{V\setminus A}$**
4: **Return: union of T_A and $T_{V\setminus A}$**

Where $G[A]$ is subgraph of G induced by $A \subseteq V$.

Recursive ϕ-Sparsest-Cut Algorithm for Hierarchical Clustering

Theorem

For any graph $G = (V,E,w)$ with $w : E \to \mathbb{R}_+$, the ϕ-sparsest-cut algorithm outputs a solution T of cost $\Gamma(T) \leq \frac{27}{4}\phi\Gamma(T^*)$ for any (in particular the optimal) clustering T^*

Recursive ϕ-Sparsest-Cut Algorithm: Proof

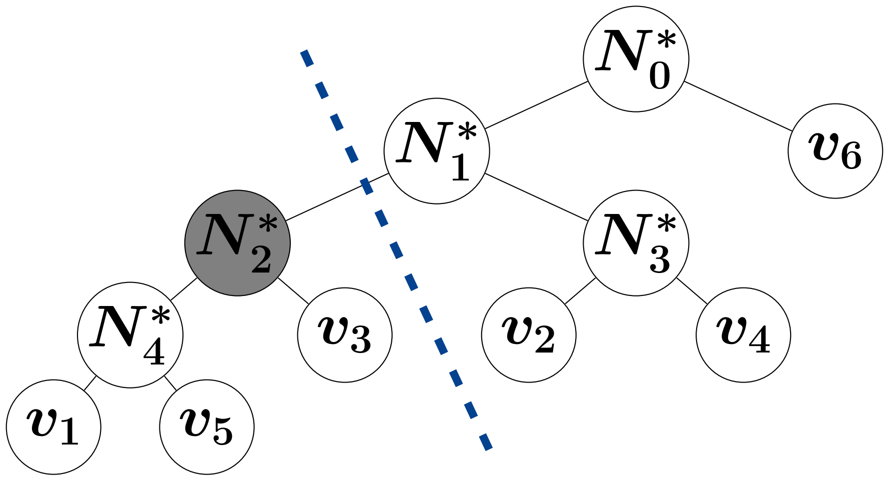


Figure: Cut of T^* ($N_2^* = N_{\text{BC}}$)

- **Descent through T^* in direction of most leaves**
- **N_{BC} first node found in T^* with $|L(N_{\text{BC}})| < \frac{2n}{3} = \frac{2|V|}{3}$**
  $\rightarrow$ Balanced cut of T^*: $(L(N_{\text{BC}}), V \setminus L(N_{\text{BC}}))$

Recursive ϕ-Sparsest-Cut Algorithm: Proof

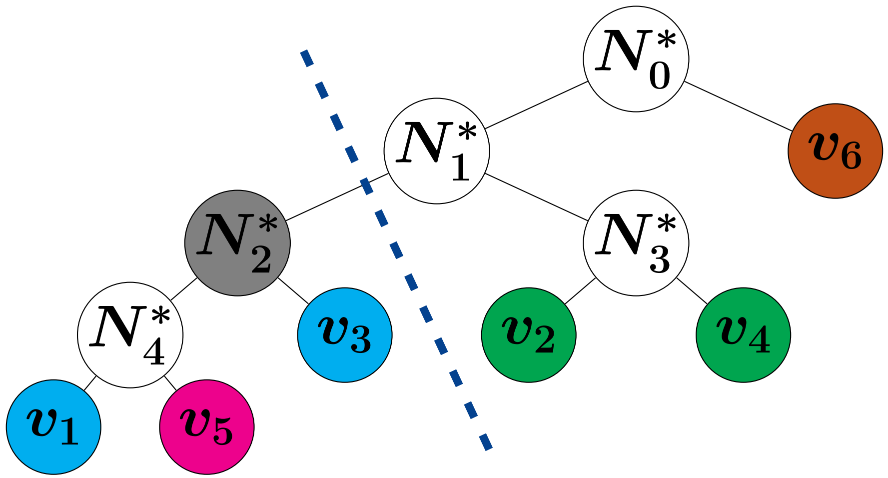


Figure: Cut of T^* ($N_2^* = N_{\text{BC}}$)

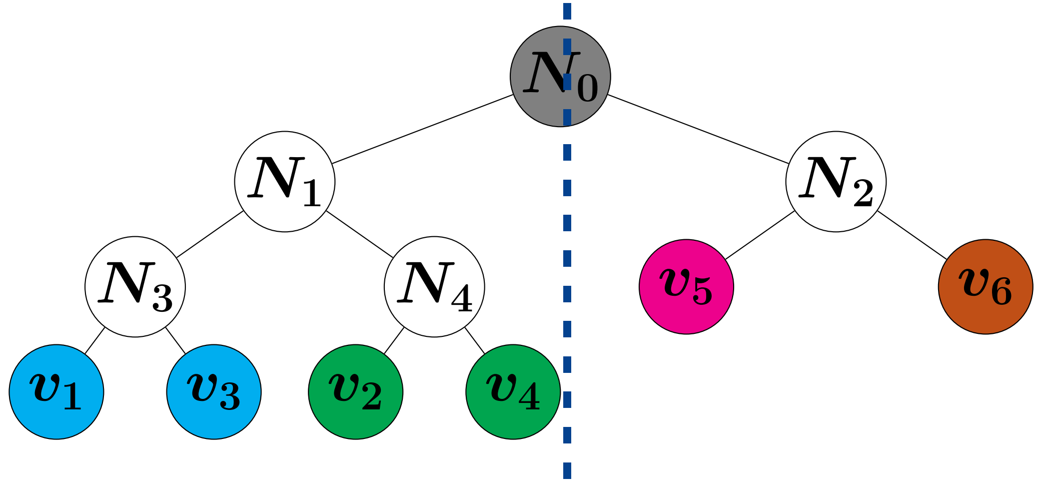


Figure: Cut of T

- **Balanced cut of T^*: $(L(N_{\text{BC}}), V \setminus L(N_{\text{BC}}))$**
- **Pick $A, B, C, D \subseteq V$ s.t.**
  - $(A \cup B, C \cup D) := (L(N_{\text{BC}}), V \setminus L(N_{\text{BC}}))$ **in T^* and**
  - $(A \cup C, B \cup D) := (L(N_0), V \setminus L(N_0))$ **cut induced by root of T**
- **In our example: $A = \{v_1, v_3\}$, $B = \{v_5\}$, $C = \{v_2, v_4\}$, $D = \{v_6\}$**

Recursive ϕ-Sparsest-Cut Algorithm: Proof

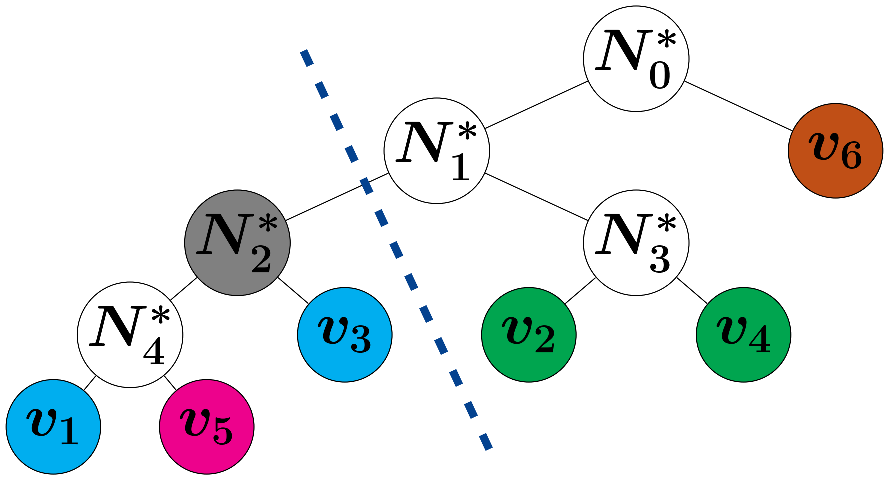


Figure: Cut of T^* ($N_2^* = N_{\mathrm{BC}}$)

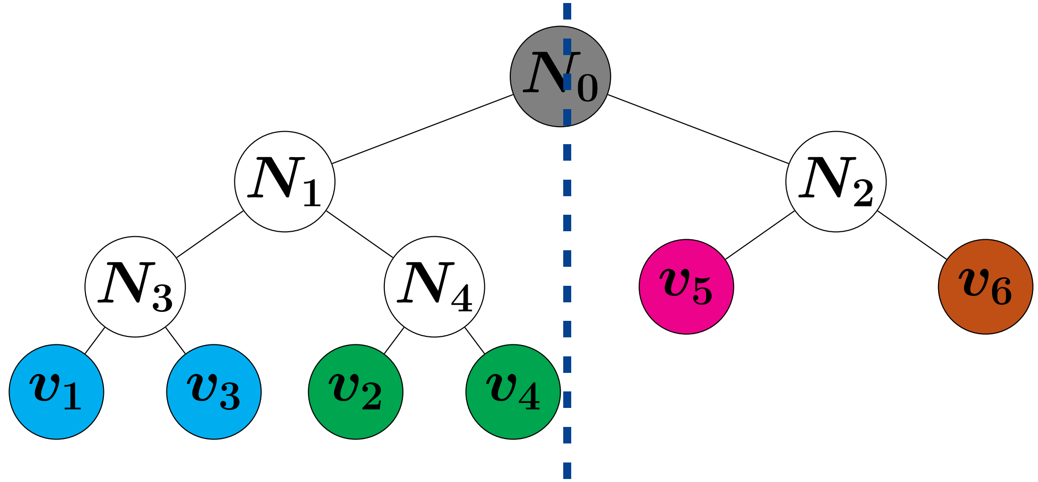


Figure: Cut of T

T output of Algo 1 $\Rightarrow$ $(A \cup C, B \cup D)$ is ϕ-approximate sparsest cut:

$$\frac{w(A \cup C, B \cup D)}{|A \cup C| \cdot |B \cup D|} \leq \phi \frac{w(A \cup B, C \cup D)}{|A \cup B| \cdot |C \cup D|}$$

Weight between $A \cup B$ and $C \cup D \leq$ summed weights between all subsets

$$\Rightarrow w(A \cup C, B \cup D) \leq \phi \frac{|A \cup C| \cdot |B \cup D|}{|A \cup B| \cdot |C \cup D|} \cdot (w(A,C) + w(A,D) + w(B,C) + w(B,D))$$

Recursive ϕ-Sparsest-Cut Algorithm: Proof

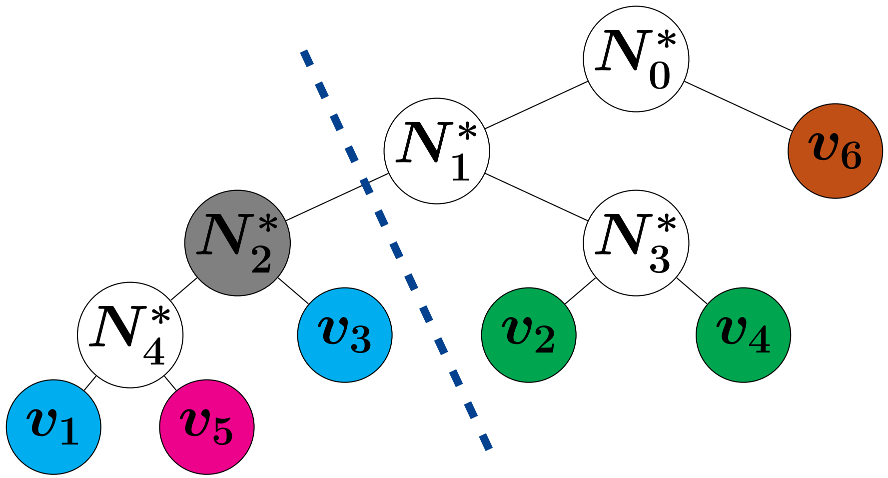


Figure: Cut of T^* ($N_2^* = N_{\text{BC}}$)

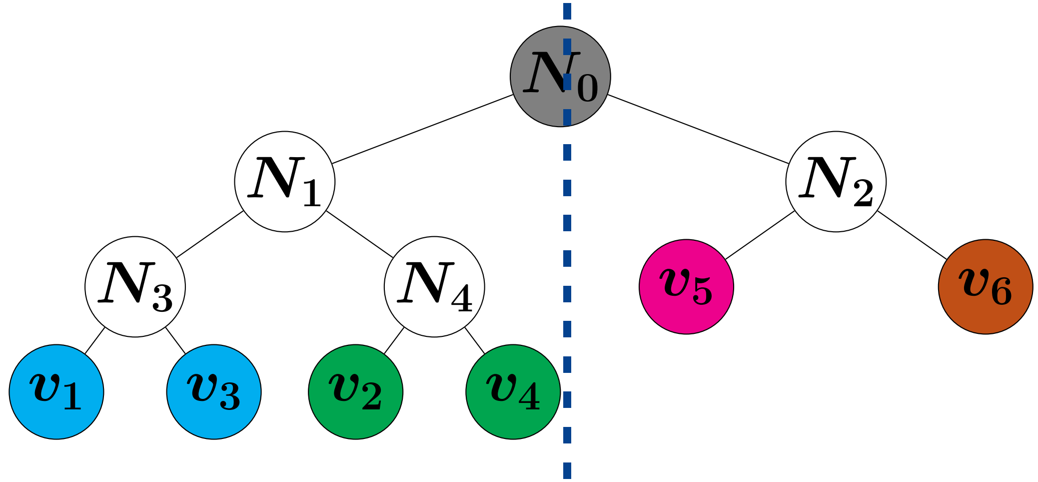


Figure: Cut of T

T output of Algo 1 $\Rightarrow$ $(A \cup C, B \cup D)$ is ϕ-approximate sparsest cut:

$$\frac{w(A \cup C, B \cup D)}{|A \cup C| \cdot |B \cup D|} \leq \phi \frac{w(A \cup B, C \cup D)}{|A \cup B| \cdot |C \cup D|}$$

Weight between $A \cup B$ and $C \cup D$ $\leq$ summed weights between all subsets

$$\Rightarrow w(A \cup C, B \cup D) \leq \phi \frac{|A \cup C| \cdot |B \cup D|}{|A \cup B| \cdot |C \cup D|} \cdot (w(A,C) + w(A,D) + w(B,C) + w(B,D))$$

Recursive ϕ-Sparsest-Cut Algorithm: Proof

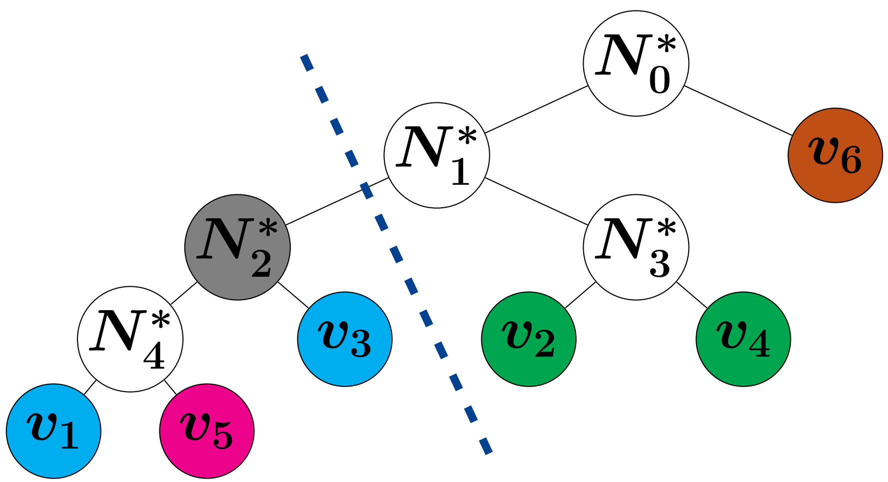


Figure: Cut of T^* ($N_2^* = N_{\text{BC}}$)

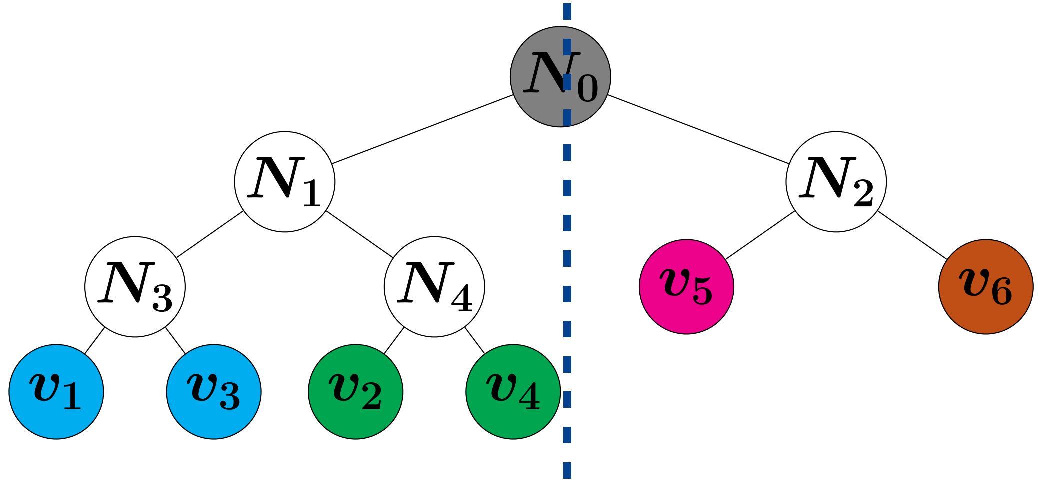


Figure: Cut of T

$$w(A \cup C, B \cup D) \leq \phi \frac{|A \cup C| \cdot |B \cup D|}{|A \cup B| \cdot |C \cup D|} \cdot (w(A,C) + w(A,D) + w(B,C) + w(B,D))$$

By defintion of N_{BC}:

- $\frac{n}{3} \leq |A \cup B| \leq \frac{2n}{3}$
- $\frac{n}{3} \leq |C \cup D| \leq \frac{2n}{3}$

Recursive ϕ-Sparsest-Cut Algorithm: Proof

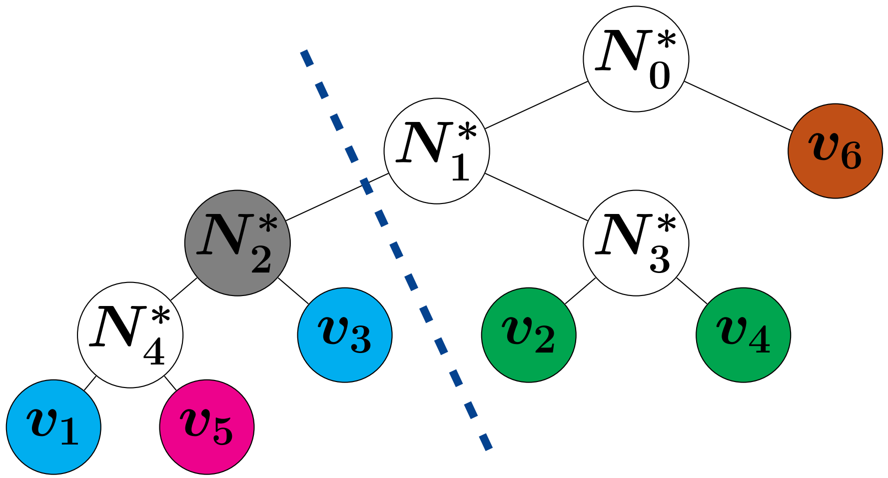


Figure: Cut of T^* ($N_2^* = N_{\text{BC}}$)

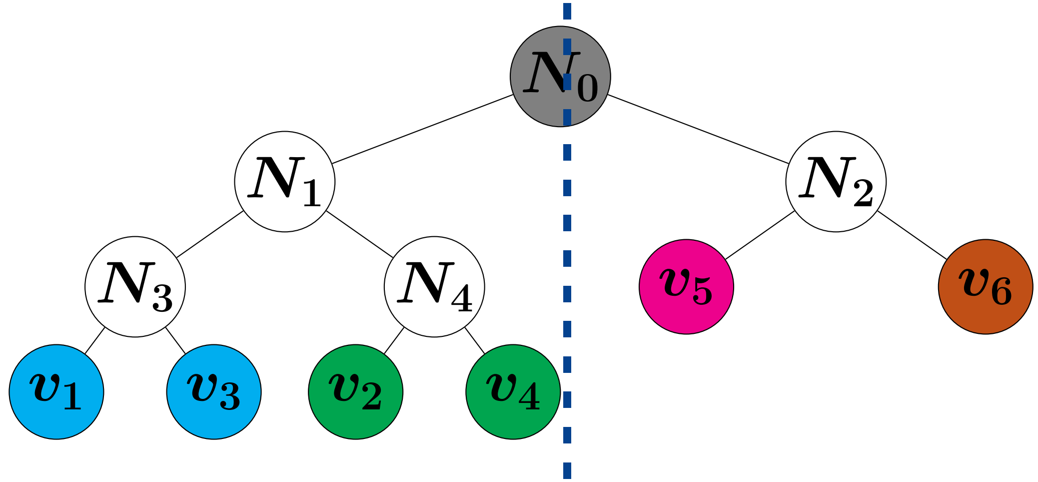


Figure: Cut of T

$$w(A \cup C, B \cup D) \leq \phi \frac{|A \cup C| \cdot |B \cup D|}{|A \cup B| \cdot |C \cup D|} \cdot (w(A,C) + w(A,D) + w(B,C) + w(B,D))$$

Lowerbound $|A \cup B| \cdot |C \cup D| \geq \frac{2n^2}{9}$ and use in the denominator:

$$\leq \phi \frac{9}{2n^2} |A \cup C| \cdot |B \cup D| \cdot (w(A,C) + w(A,D) + w(B,C) + w(B,D))$$

Recursive ϕ-Sparsest-Cut Algorithm: Proof

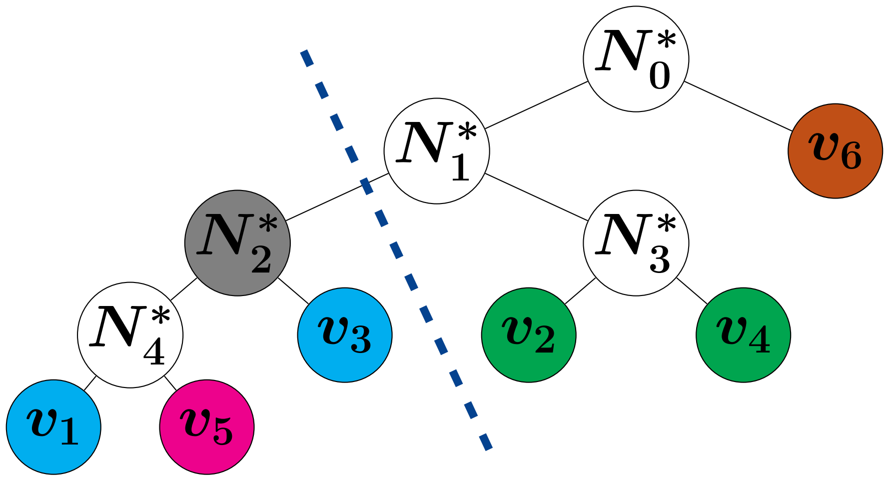


Figure: Cut of T^* ($N_2^* = N_{\mathrm{BC}}$)

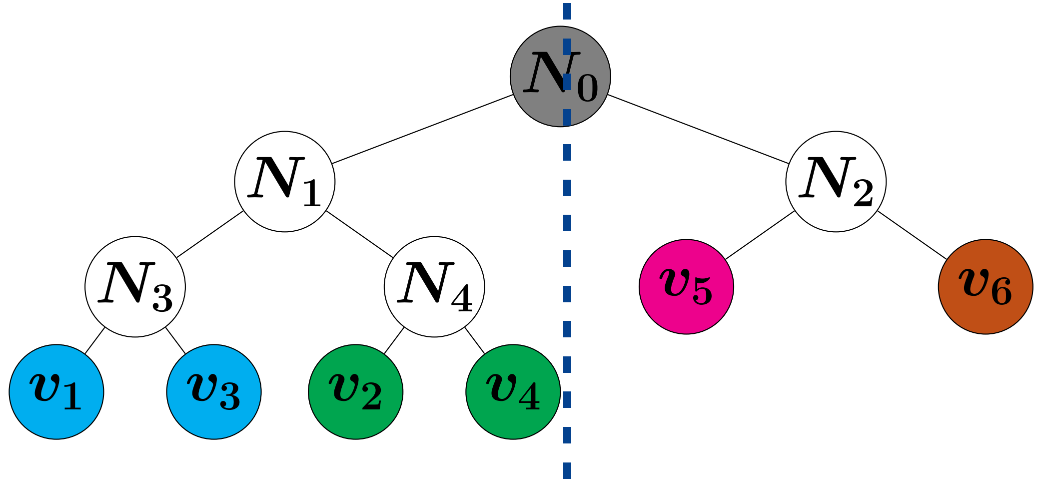


Figure: Cut of T

$$
\begin{aligned}
w(A \cup C, B \cup D) \leq & \phi \frac{9}{2n^2} |A \cup C| \cdot |B \cup D| \\
& \cdot (w(A,C) + w(A,D) + w(B,C) + w(B,D))
\end{aligned}
$$

Exploiting the fact that $\frac{|X|}{n} \leq 1$ for any $X \subseteq V$:

$$
\begin{aligned}
\leq & \phi \frac{9}{2} \bigg(\frac{|B \cup D|}{n} w(A,C) + w(A,D) \\
& + w(B,C) + \frac{|A \cup C|}{n} w(B,D) \bigg)
\end{aligned}
$$

Recursive ϕ-Sparsest-Cut Algorithm: Proof

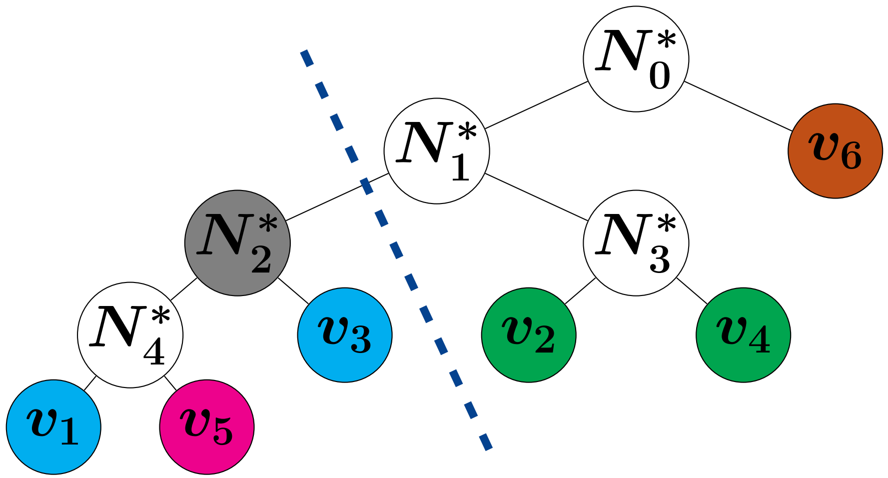


Figure: Cut of T^* ($N_2^* = N_{\mathrm{BC}}$)

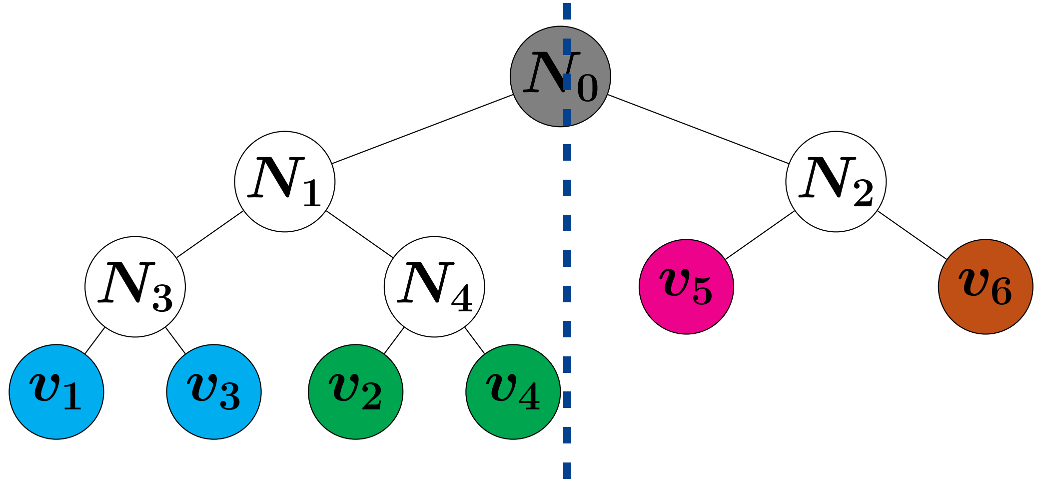


Figure: Cut of T

Cost induced by the root N_0 satisfies:

$$\begin{aligned}\gamma(N_0) = nw(A \cup C, B \cup D) \leq\ & \frac{9}{2}\phi|A \cup C|w(B,D) \\ & + \frac{9}{2}\phi|B \cup D|w(A,C) \\ & + \frac{9}{2}n\phi[w(A,D) + w(B,C)]\end{aligned}$$

Recursive ϕ-Sparsest-Cut Algorithm: Proof

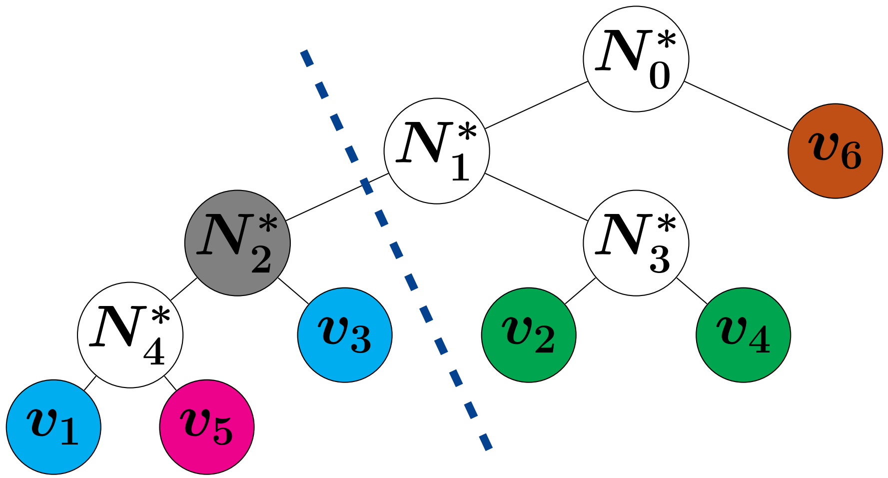


Figure: Cut of T^* ($N_2^* = N_{\text{BC}}$)

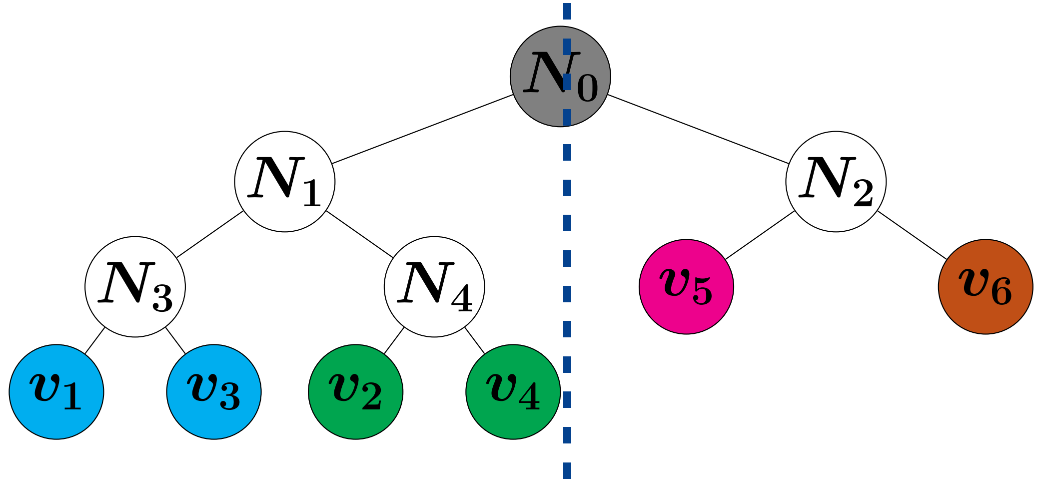


Figure: Cut of T

Cost induced by the root N_0 satisfies:

$$\begin{aligned}\gamma(N_0) = nw(A \cup C, B \cup D) \leq & \frac{9}{2}\phi|A \cup C|w(B,D) \\ & + \frac{9}{2}\phi|B \cup D|w(A,C) \\ & + \frac{9}{2}n\phi[w(A,D) + w(B,C)]\end{aligned}$$

Recursive ϕ-Sparsest-Cut Algorithm: Proof

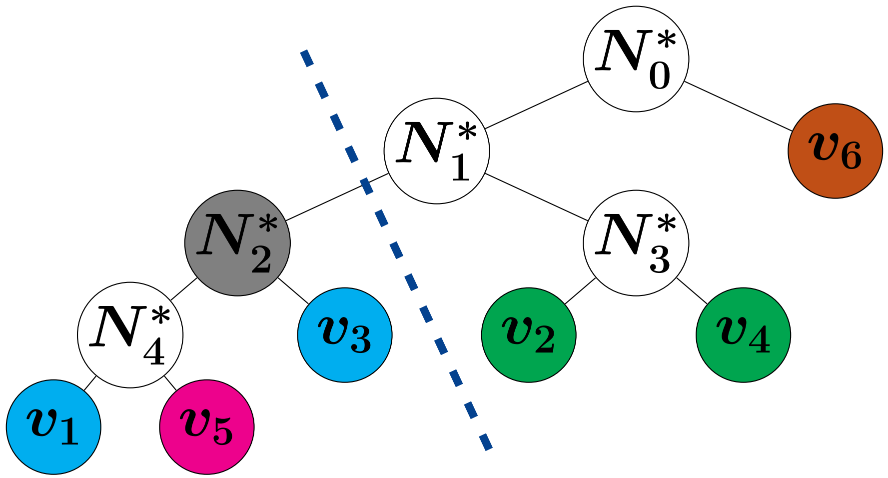


Figure: Cut of T^* ($N_2^* = N_{\text{BC}}$)

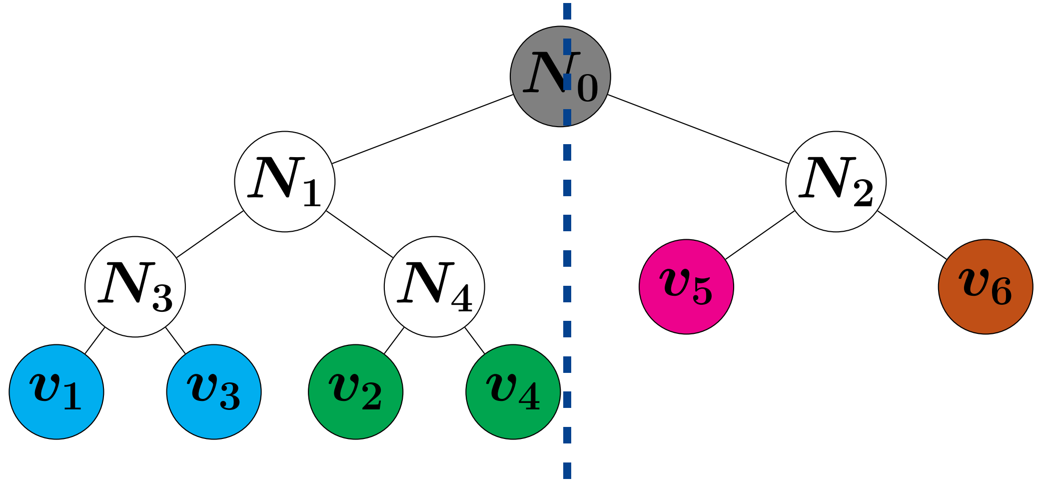


Figure: Cut of T

Induction over the nodes proves the theorem:

$$\Gamma(T) = \sum_{N_i \in \mathcal{N}} \gamma(N_i) \leq \frac{27}{4} \phi \Gamma(T^*)$$

$\square$

Recursive ϕ-Sparsest-Cut Algorithm

Adapting balanced cut and rescaling at some point by a factor $\beta(\Gamma)$:

Remark

For any admissible cost function Γ:

Algorithm 1 achieves an $\mathcal{O}(\beta(\Gamma) \cdot \phi)$-approximation.

Outline

Fast and Simple Algorithm for Clustering on Perfect Ground-Truth Inputs

2. Fast and Simple Clustering

Input: $G = (V,E,w)$
$p \leftarrow$ **random vertex of** V
$w_1 > \ldots > w_k$ **edge weights of edges** $\{\cdot,p\}$
Let $B_i = \{v|w(p,v) = w_i\}$ **for** $1 \leq i \leq k$
Recurse on each $G[B_i]$ **and obtain** $T_1,\ldots,T_k$
$T_0^* \leftarrow$ **tree with** p **as single vertex**
$T_i^* \leftarrow$ **union of** T_{i-1}^* **and** T_i
Return: T_k^*

Fast and Simple Clustering: Example

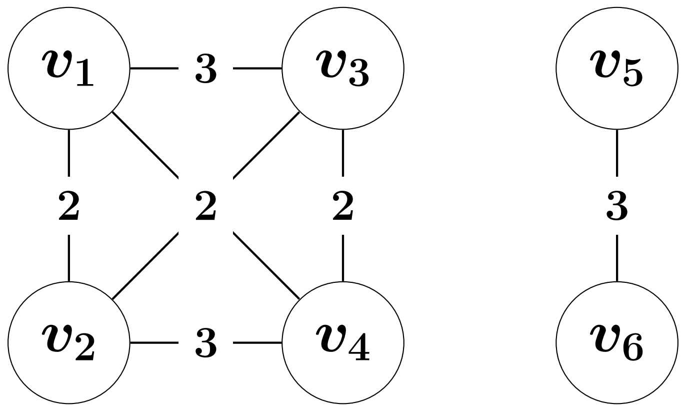


Figure: Example of clustering using the fast and simple clustering algorithm.

1. **Select:** $p \leftarrow v_3$ **(randomly)**
   - **Sort edge weights:** $w_{13} > w_{23} = w_{34} > w_{35} = w_{36}$
   - **Partition into buckets:** $B_1 = \{v_1\}$, $B_2 = \{v_2, v_4\}$, $B_3 = \{v_5, v_6\}$

Fast and Simple Clustering: Example

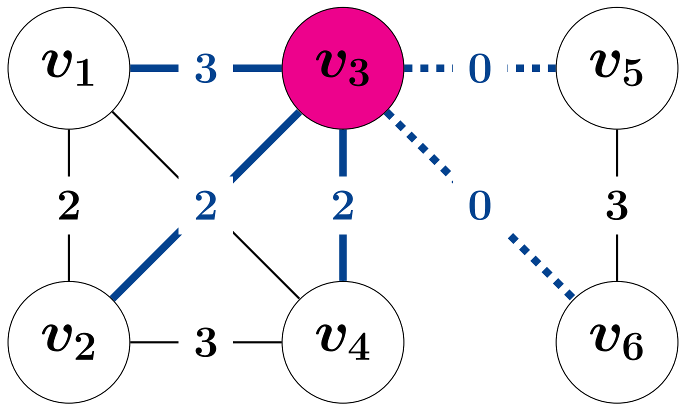


Figure: Example of clustering using the fast and simple clustering algorithm.

1. **Select:** $p \leftarrow v_3$ **(randomly)**
   - ▷ **Sort edge weights:** $w_{13} > w_{23} = w_{34} > w_{35} = w_{36}$
   - ▷ **Partition into buckets:** $B_1 = \{v_1\}$, $B_2 = \{v_2, v_4\}$, $B_3 = \{v_5, v_6\}$

Fast and Simple Clustering: Example

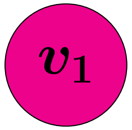


Figure: Example of clustering using the fast and simple clustering algorithm.

1. **Select:** $p \leftarrow v_3$ **(randomly)**
   - ▷ **Sort edge weights:** $w_{13} > w_{23} = w_{34} > w_{35} = w_{36}$
   - ▷ **Partition into buckets:** $B_1 = \{v_1\}$, $B_2 = \{v_2, v_4\}$, $B_3 = \{v_5, v_6\}$

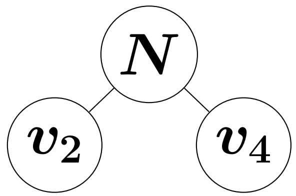
1.1 **Recurse on** $G[B_1] \rightarrow$ **Return:** $T_1 :=$ v_1

Fast and Simple Clustering: Example

Figure: Example of clustering using the fast and simple clustering algorithm.

1. **Select:** $p \leftarrow v_3$ **(randomly)**
   - ▷ **Sort edge weights:** $w_{13} > w_{23} = w_{34} > w_{35} = w_{36}$
   - ▷ **Partition into buckets:** $B_1 = \{v_1\}$, $B_2 = \{v_2, v_4\}$, $B_3 = \{v_5, v_6\}$

1.2 Recurse on $G[B_2] \rightarrow$ **Return:** $T_2 :=$

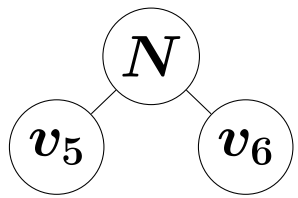


Fast and Simple Clustering: Example

Figure: **Example of clustering using the fast and simple clustering algorithm.**

1. **Select:** $p \leftarrow v_3$ **(randomly)**
   - ▷ **Sort edge weights:** $w_{13} > w_{23} = w_{34} > w_{35} = w_{36}$
   - ▷ **Partition into buckets:** $B_1 = \{v_1\}$, $B_2 = \{v_2, v_4\}$, $B_3 = \{v_5, v_6\}$

1.3 Recurse on $G[B_3] \rightarrow$ **Return:** $T_3 :=$



Fast and Simple Clustering: Example

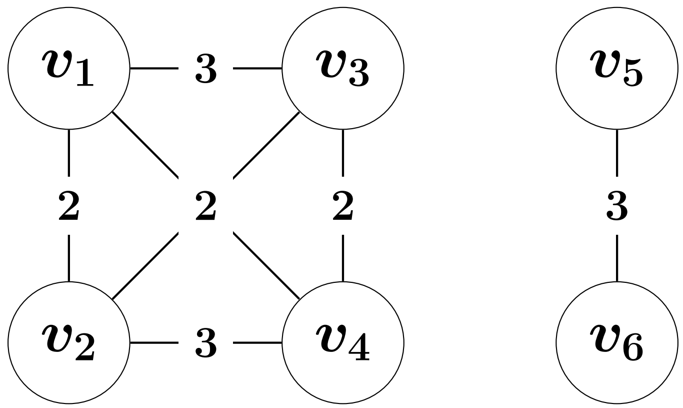


Figure: Example of clustering using the fast and simple clustering algorithm.

1. **Select:** $p \leftarrow v_3$ **(randomly)**
   - **Sort edge weights:** $w_{13} > w_{23} = w_{34} > w_{35} = w_{36}$
   - **Partition into buckets:** $B_1 = \{v_1\}$, $B_2 = \{v_2, v_4\}$, $B_3 = \{v_5, v_6\}$
2. **Merge all subtrees:**

Fast and Simple Clustering: Example

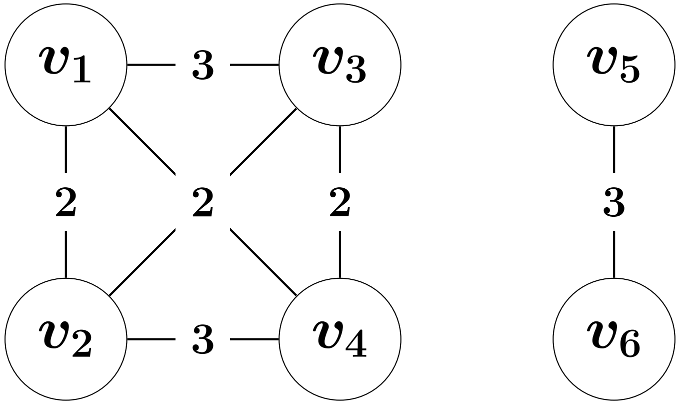


Figure: Example of clustering using the fast and simple clustering algorithm.

1. **Select:** $p \leftarrow v_3$ **(randomly)**
   - ▷ **Sort edge weights:** $w_{13} > w_{23} = w_{34} > w_{35} = w_{36}$
   - ▷ **Partition into buckets:** $B_1 = \{v_1\}$, $B_2 = \{v_2, v_4\}$, $B_3 = \{v_5, v_6\}$
2. **Merge all subtrees:**

2.0 Start with $T_0^* =$

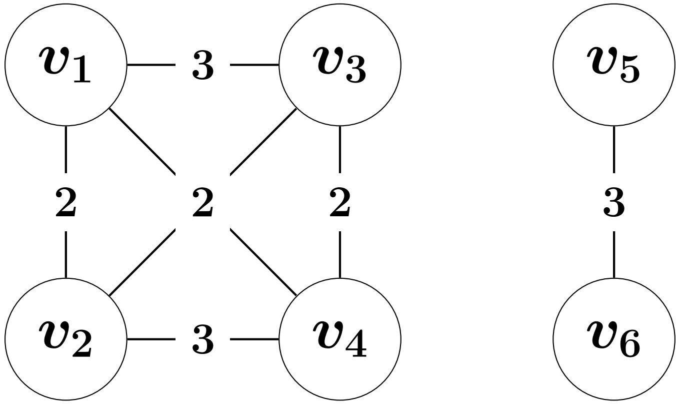
Fast and Simple Clustering: Example

Figure: Example of clustering using the fast and simple clustering algorithm.

1. **Select:** $p \leftarrow v_3$ (randomly)
   - ▷ **Sort edge weights:** $w_{13} > w_{23} = w_{34} > w_{35} = w_{36}$
   - ▷ **Partition into buckets:** $B_1 = \{v_1\}$, $B_2 = \{v_2, v_4\}$, $B_3 = \{v_5, v_6\}$
2. **Merge all subtrees:**

2.1 **Start with** $T_1^* \leftarrow$ **union of** T_0^* **and** T_1

Fast and Simple Clustering: Example



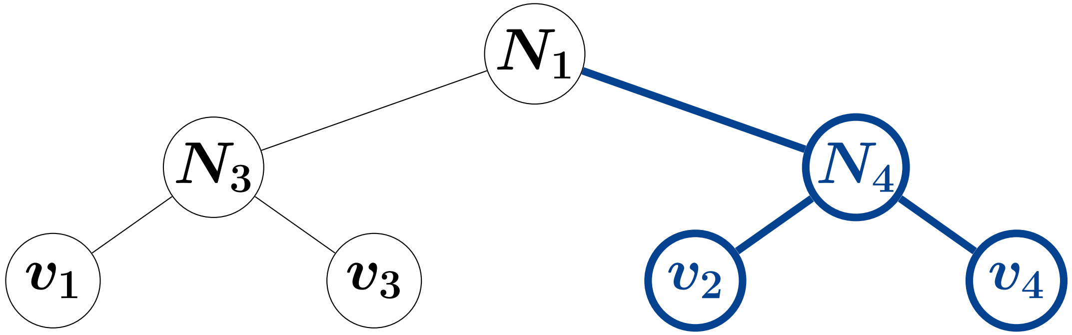


Figure: Example of clustering using the fast and simple clustering algorithm.

1. **Select:** $p \leftarrow v_3$ **(randomly)**
   - ▷ **Sort edge weights:** $w_{13} > w_{23} = w_{34} > w_{35} = w_{36}$
   - ▷ **Partition into buckets:** $B_1 = \{v_1\}$, $B_2 = \{v_2, v_4\}$, $B_3 = \{v_5, v_6\}$
2. **Merge all subtrees:**

2.2 **Start with** $T_2^* \leftarrow$ **union of** T_1^* **and** T_2

Fast and Simple Clustering: Example

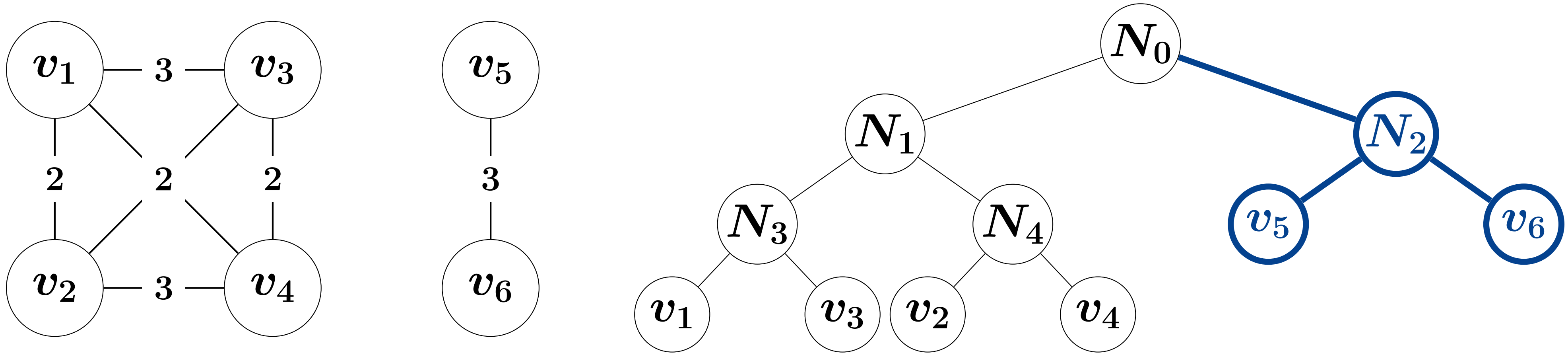


Figure: Example of clustering using the fast and simple clustering algorithm.

1. **Select:** $p \leftarrow v_3$ **(randomly)**
   - **Sort edge weights:** $w_{13} > w_{23} = w_{34} > w_{35} = w_{36}$
   - **Partition into buckets:** $B_1 = \{v_1\}$, $B_2 = \{v_2, v_4\}$, $B_3 = \{v_5, v_6\}$
2. **Merge all subtrees:**

2.3 Start with $T_3^* \leftarrow$ **union of** T_2^* **and** T_3

Fast and Simple Clustering: Example

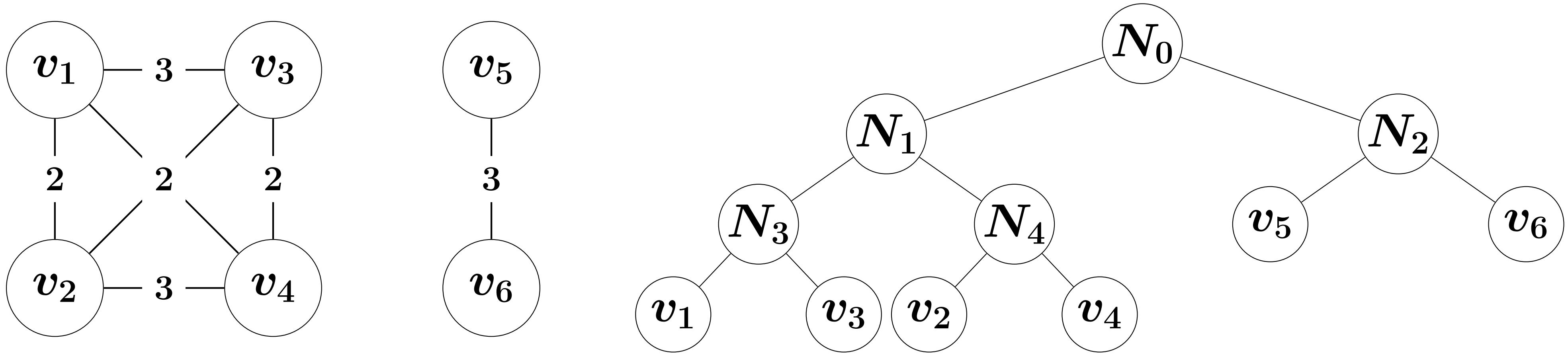


Figure: Example of clustering using the fast and simple clustering algorithm.

1. **Select:** $p \leftarrow v_3$ **(randomly)**
   - ▷ **Sort edge weights:** $w_{13} > w_{23} = w_{34} > w_{35} = w_{36}$
   - ▷ **Partition into buckets:** $B_1 = \{v_1\}$, $B_2 = \{v_2, v_4\}$, $B_3 = \{v_5, v_6\}$
2. **Merge all subtrees:**
3. **Return** T_3^*

Fast and Simple Algorithm for Clustering on Perfect Ground-Truth Inputs

Theorem

For any admissible objective function, the fast and simple clustering algorithm computes a tree of optimal cost

- ***in $\mathcal{O}(n \log^2 n)$ with high probability if the input is a strict ground-truth input or***
- ***in $\mathcal{O}(n^2)$ if the input is a (non-necessarily strict) ground-truth input***

Fast and Simple Clustering: Proof of Correctness

- Pivot element $p \in V$
- Let $u \in B_i$, $v \in B_j$ for partitions based on p with $j > i$
- Let T be a generating tree
- Define $N_u :=$ LCA$_T(p,u)$ and $N_v :=$ LCA$_T(p,v)$
- Observe: $w(p,u) > w(p,v)$ (from construction of B_i,B_j)
  $\Rightarrow N_v$ ancestor of N_u in T

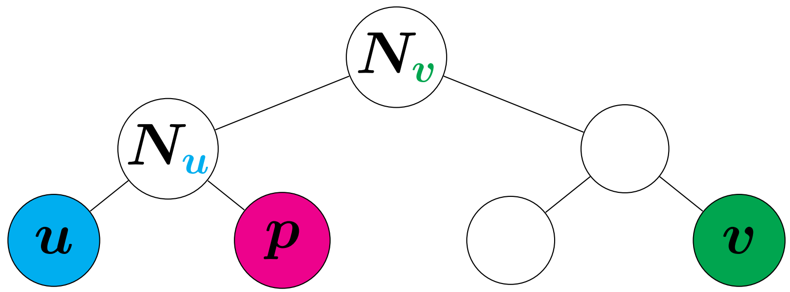


Figure: Visualizing the proof.

- Obeserve: LCA$_T(u,v) = N_v$
  $\Rightarrow w(u,v) = W(N_v) = w(p,v)$ for all $u \in B_i$ and $v \in B_j$
- Apply inductively to all $G[B_i] \Rightarrow$ output is a generating tree for G $\square$

Fast and Simple Clustering: Proof of Correctness

- Pivot element $p \in V$
- Let $u \in B_i$, $v \in B_j$ for partitions based on p with $j > i$
- Let T be a generating tree
- Define $N_u :=$ LCA$_T(p,u)$ and $N_v :=$ LCA$_T(p,v)$
- Observe: $w(p,u) > w(p,v)$ (from construction of B_i,B_j)
  $\Rightarrow N_v$ ancestor of N_u in T

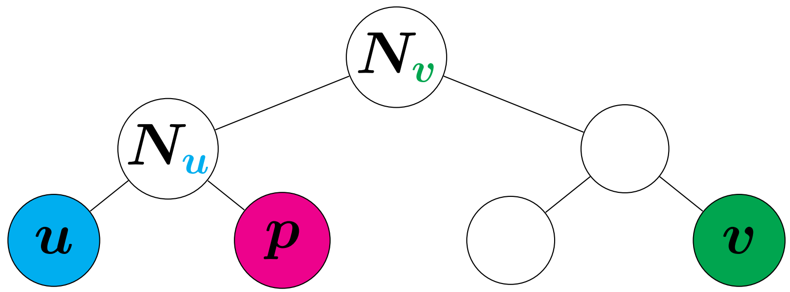


Figure: Visualizing the proof.

- Obeserve: LCA$_T(u,v) = N_v$
  $\Rightarrow w(u,v) = W(N_v) = w(p,v)$ for all $u \in B_i$ and $v \in B_j$
- Apply inductively to all $G[B_i] \Rightarrow$ output is a generating tree for G $\square$

Fast and Simple Clustering: Proof of Complexity

- Let input G be strongly generated by some T
- Recursive call on $G[B_0]$ with $|B_0| = n_0$.

If $\frac{|B_i|}{n_0-1} \leq \frac{2}{3} \, \forall i \in [1,k]$:

- Apply the Master theorem [Cormen & Leiserson⁺ 09]:

$$\text{Time}(n_0) = \sum_{i=1}^{k} \text{Time}(\alpha_i(n_0 - 1)) + f(n_0)$$

- $f(n_0)$: Time for recursive call before further recursion
  - Fulfills: $f(n_0) \in \mathcal{O}(n_0)$
- $\alpha_i = \frac{|B_i|}{n_0-1}$
  - Fulfills: $\sum_{i=1}^{k} \alpha_i = 1$ and $\alpha_i < 1$

$\Rightarrow$ Time$(n) \in \mathcal{O}(n \log n)$.

Fast and Simple Clustering: Proof of Complexity

- Let input G be strongly generated by some T
- Recursive call on $G[B_0]$ with $|B_0| = n_0$.

If $\frac{|B_i|}{n_0 - 1} \leq \frac{2}{3} \; \forall i \in [1, k]$:

- Apply the Master theorem [Cormen & Leiserson^{+} 09]:

$$\text{Time}(n_0) = \sum_{i=1}^{k} \text{Time}(\alpha_i (n_0 - 1)) + f(n_0)$$

- $f(n_0)$: Time for recursive call before further recursion
  - Fulfills: $f(n_0) \in \mathcal{O}(n_0)$
- $\alpha_i = \frac{|B_i|}{n_0 - 1}$
  - Fulfills: $\sum_{i=1}^{k} \alpha_i = 1$ and $\alpha_i < 1$

$\Rightarrow$ Time$(n) \in \mathcal{O}(n \log n)$.

If $|B_i| = n_0 - 1$:
$\Rightarrow$ Time$(n) \in \mathcal{O}(n^2)$.

Fast and Simple Clustering: Proof of Complexity

- **Let input G be strongly generated by some T**
- **Recursive call on $G[B_0]$ with $|B_0| = n_0$.**

If $\frac{|B_i|}{n_0-1} \leq \frac{2}{3}\ \forall i \in [1,k]$:

- **Apply the Master theorem [Cormen & Leiserson+ 09]:**

$$\text{Time}(n_0) = \sum_{i=1}^{k} \text{Time}(\alpha_i(n_0 - 1)) + f(n_0)$$

- **$f(n_0)$: Time for recursive call before further recursion**
  - **Fulfills: $f(n_0) \in \mathcal{O}(n_0)$**
- $\alpha_i = \frac{|B_i|}{n_0-1}$
  - **Fulfills: $\sum_{i=1}^{k} \alpha_i = 1$ and $\alpha_i < 1$**

$\Rightarrow$ Time$(n) \in \mathcal{O}(n \log n)$.

If $|B_i| = n_0 - 1$: $\rightarrow$ This occurs with low probability.
$\Rightarrow$ Time$(n) \in \mathcal{O}(n^2)$.

Outline

Conclusion and Discussion

- **Framework for objective functions in hierarchical clustering**
- **Characterization of "good" objective functions**
  $\rightarrow$ admissible cost functions

- **Recursive ϕ-Sparsest Cut algorithm:**
  - **Proved $\mathcal{O}(\phi)$ approximation of an optimal output**

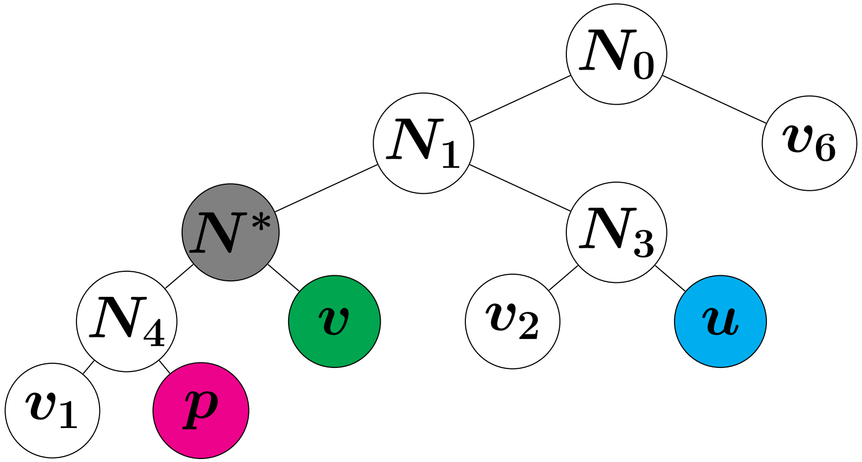
- **Fast and Simple Clustering algorithm:**
  - **Guarantees optimal clustering in $\mathcal{O}(n \log^2 n)$ time for perfect inputs**
  - **A modified version gives approximate output for unstructured inputs**

Thank you for your attention!

Arne Nix

arne.nix@rwth-aachen.de

Fast and Simple Clustering: Proof of Complexity



- Find N^*: descend T towards most leaves until $\frac{2n}{3} > |L(N^*)| \geq \frac{n}{3}$
- Let $p \in L(N^*)$
- Let $u \in V$ with LCA$_T(u,p)$ ancestor of N^*
  - For any $v \in L(N^*)$: $w(u,p) < w(v,p)$

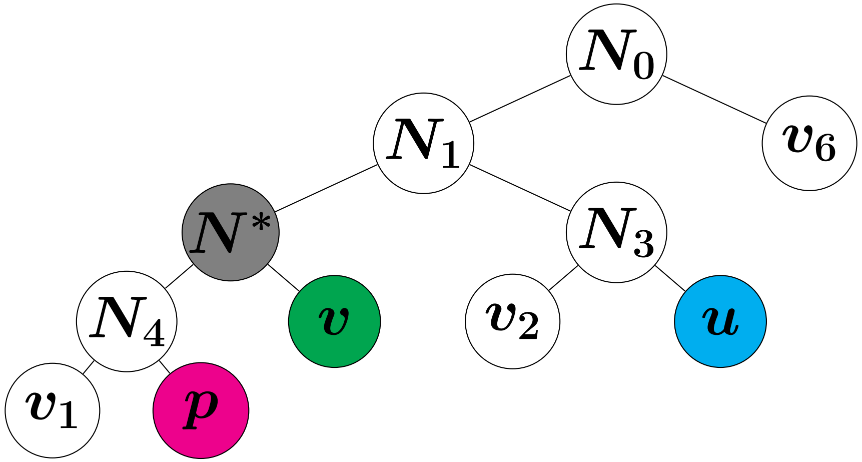
    $\Rightarrow$ v and u in different B_i

    $\Rightarrow$ $|B_i| < \frac{2n}{3}\ \forall i \in [1,k]$ for any partition induced by $p \in L(N^*)$

    $\Rightarrow$ Each $p \in L(N^*)$ would introduce “good” paritions.

    $\Rightarrow$ $Pr_{p\sim V}(p$ is “bad” pivot$) = Pr_{p\sim V}(p \notin L(N^*)) \in \mathcal{O}(\frac{2}{3})$.

Fast and Simple Clustering: Proof of Complexity



- Find N^*: descend T towards most leaves until $\frac{2n}{3} > |L(N^*)| \geq \frac{n}{3}$
- Let $p \in L(N^*)$
- Let $u \in V$ with LCA$_T(u,p)$ ancestor of N^*
  - For any $v \in L(N^*)$: $w(u,p) < w(v,p) \rightarrow$ **strictly generated!**

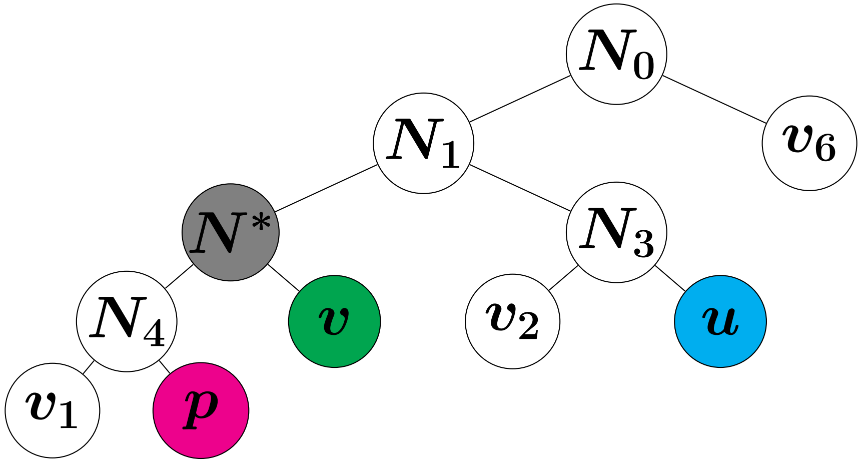
    $\Rightarrow$ v and u in different B_i

    $\Rightarrow$ $|B_i| < \frac{2n}{3}\ \forall i \in [1,k]$ for any partition induced by $p \in L(N^*)$

    $\Rightarrow$ Each $p \in L(N^*)$ would introduce “good” paritions.

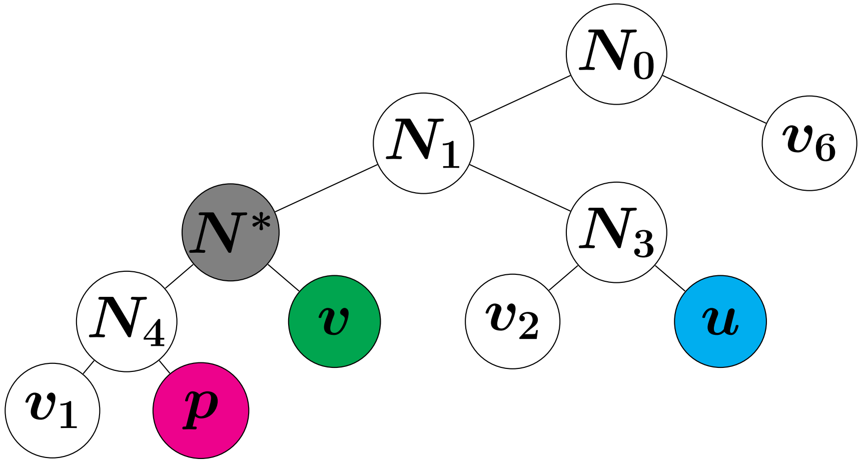
    $\Rightarrow$ $Pr_{p\sim V}(p$ is “bad” pivot$) = Pr_{p\sim V}(p \notin L(N^*)) \in \mathcal{O}(\frac{2}{3})$.

Fast and Simple Clustering: Proof of Complexity



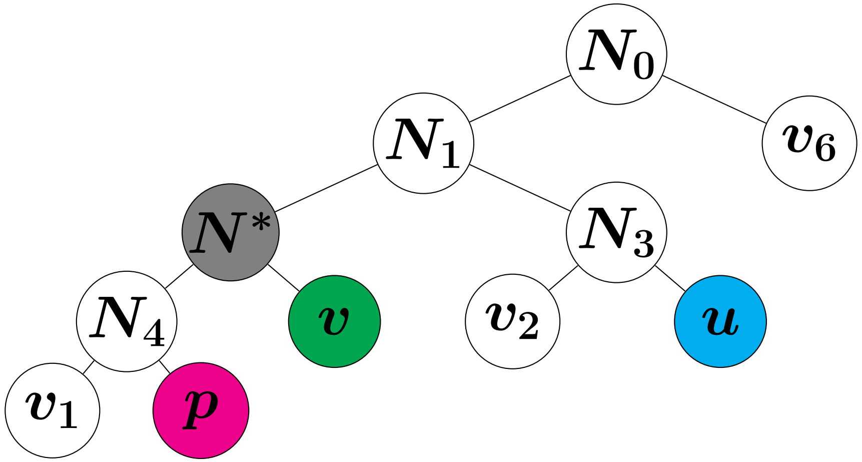
- Find N^*: descend T towards most leaves until $\frac{2n}{3} > |L(N^*)| \geq \frac{n}{3}$
- Let $p \in L(N^*)$
- Let $u \in V$ with LCA$_T(u,p)$ ancestor of N^*
  - For any $v \in L(N^*)$: $w(u,p) < w(v,p)$
    $\Rightarrow$ v and u in different B_i
    $\Rightarrow |B_i| < \frac{2n}{3} \; \forall i \in [1,k]$ for any partition induced by $p \in L(N^*)$
    $\Rightarrow$ Each $p \in L(N^*)$ would introduce "good" paritions.
    $\Rightarrow Pr_{p \sim V}(p$ is "bad" pivot$) = Pr_{p \sim V}(p \notin L(N^*)) \in \mathcal{O}(\frac{2}{3})$.

Fast and Simple Clustering: Proof of Complexity



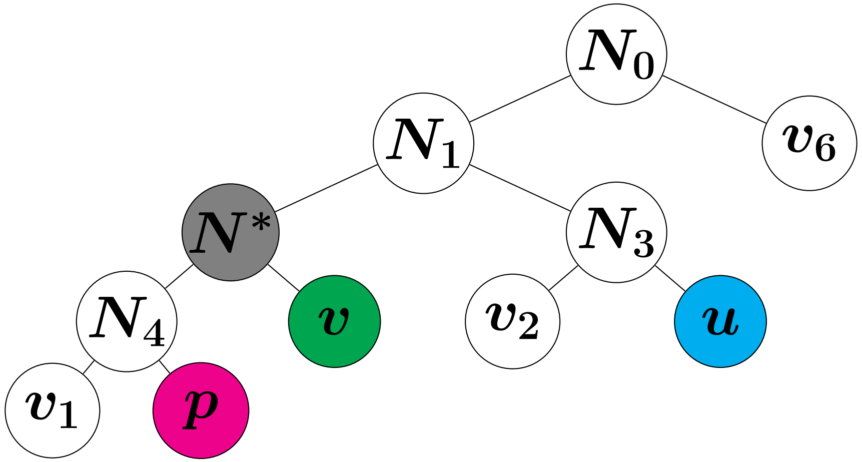
- Find N^*: descend T towards most leaves until $\frac{2n}{3} > |L(N^*)| \geq \frac{n}{3}$
- Let $p \in L(N^*)$
- Let $u \in V$ with LCA$_T(u,p)$ ancestor of N^*
  - For any $v \in L(N^*)$: $w(u,p) < w(v,p)$
    $\Rightarrow$ v and u in different B_i
    $\Rightarrow |B_i| < \frac{2n}{3}\ \forall i \in [1,k]$ for any partition induced by $p \in L(N^*)$
    $\Rightarrow$ Each $p \in L(N^*)$ would introduce “good” paritions.
    $\Rightarrow Pr_{p\sim V}(p$ is “bad” pivot$) = Pr_{p\sim V}(p \notin L(N^*)) \in \mathcal{O}(\frac{2}{3})$.

Fast and Simple Clustering: Proof of Complexity



- Find N^*: descend T towards most leaves until $\frac{2n}{3} > |L(N^*)| \geq \frac{n}{3}$
- Let $p \in L(N^*)$
- Let $u \in V$ with LCA$_T(u,p)$ ancestor of N^*
  - For any $v \in L(N^*)$: $w(u,p) < w(v,p)$
    $\Rightarrow$ v and u in different B_i
    $\Rightarrow |B_i| < \frac{2n}{3} \; \forall i \in [1,k]$ for any partition induced by $p \in L(N^*)$
    $\Rightarrow$ Each $p \in L(N^*)$ would introduce "good" paritions.
    $\Rightarrow Pr_{p \sim V}(p$ is "bad" pivot$) = Pr_{p \sim V}(p \notin L(N^*)) \in \mathcal{O}(\frac{2}{3})$.

Fast and Simple Clustering: Proof of Complexity



- Find N^*: descend T towards most leaves until $\frac{2n}{3} > |L(N^*)| \geq \frac{n}{3}$
- Let $p \in L(N^*)$
- Let $u \in V$ with LCA$_T(u,p)$ ancestor of N^*
  - For any $v \in L(N^*)$: $w(u,p) < w(v,p)$
    $\Rightarrow$ v and u in different B_i
    $\Rightarrow |B_i| < \frac{2n}{3}\ \forall i \in [1,k]$ for any partition induced by $p \in L(N^*)$
    $\Rightarrow$ Each $p \in L(N^*)$ would introduce “good” paritions.
    $\Rightarrow Pr_{p \sim V}(p$ is “bad” pivot$) = Pr_{p\sim V}(p \notin L(N^*)) \in \mathcal{O}(\frac{2}{3})$.

Fast and Simple Clustering: Proof of Complexity

- **Each p picked independently**
  $\Rightarrow$ Prob. of $p \notin L(N^*)$ after $c \log n$ calls $\in \mathcal{O}((\frac{2}{3})^{c \log n}) = \mathcal{O}(\frac{1}{n^c})$.

- **Not having $\log n$ "good" partitions after $\log(n) \cdot (c \log n)$ calls:**

$$\underset{p_1,\ldots,p_{\log n} \sim V}{Pr} \left(\bigvee_{i=1}^{\log n} p_i \notin L(N^*) \right) \leq \sum_{i=1}^{\log n} Pr_{p_i \sim V} \left(p_i \notin L(N^*)\right)$$

$$\in \mathcal{O}(\frac{\log n}{n^c}) = \mathcal{O}(\frac{1}{n^{c-1}})$$

$\Rightarrow |B_i| \leq \frac{2n}{3}$ for runtime $\mathcal{O}(n \log^2 n)$ holds with high probability. □

- **S. Arora, S. Rao, U. Vazirani:**
  Expander Flows, Geometric Embeddings and Graph Partitioning.
  J. ACM, Vol. 56, No. 2, pp. 5:1–5:37, April 2009.

- **V. Cohen-Addad, V. Kanade, F. Mallmann-Trenn, C. Mathieu:**
  ***Hierarchical Clustering: Objective Functions and Algorithms*, pp. 378–397.**
  2018.

- **T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein:**
  ***Introduction to Algorithms, Third Edition*.**
  The MIT Press, 3rd edition, 2009.

- **S. Dasgupta:**
  A Cost Function for Similarity-based Hierarchical Clustering.
  Proc. *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pp. 118–127, New York, NY, USA, 2016. ACM.