

# Implementierung der Cut & Count-Technik am Beispiel Steiner tree

Levin von Hollen, Tilman Beck

*{stu127560-, stu127568-}@informatik.uni-kiel.de*

Christian-Albrechts Universität Kiel

7. November 2016

- 1 Cut & Count-Technik
  - Allgemeines
  - (Nice) Tree Decomposition
- 2 Cut & Count mit Steiner Tree
  - Cut
  - Count
- 3 Implementierung

# Cut & Count-Technik

# Cut & Count-Technik

- Technik um connectivity-type Probleme mithilfe von Randomisierung zu lösen (Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan van Rooij, Jakub Onufry Wojtaszczyk)

- Technik um connectivity-type Probleme mithilfe von Randomisierung zu lösen (Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan van Rooij, Jakub Onufry Wojtaszczyk)
- angewendet auf viele verschiedene Probleme (z.B. Longest Path, Steiner Tree, Feedback Vertex Set, uvm.)

- Technik um connectivity-type Probleme mithilfe von Randomisierung zu lösen (Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan van Rooij, Jakub Onufry Wojtaszczyk)
- angewendet auf viele verschiedene Probleme (z.B. Longest Path, Steiner Tree, Feedback Vertex Set, uvm.)
- Randomisierung durch Isolation-Lemma

- Technik um connectivity-type Probleme mithilfe von Randomisierung zu lösen (Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan van Rooij, Jakub Onufry Wojtaszczyk)
- angewendet auf viele verschiedene Probleme (z.B. Longest Path, Steiner Tree, Feedback Vertex Set, uvm.)
- Randomisierung durch Isolation-Lemma
- als Ergebnis ein einseitiger Monte-Carlo-Algorithmus mit Laufzeit  $c^{tw(G)} |V|^{\mathcal{O}(1)}$



## Theorem

There exist Monte-Carlo algorithms that given a tree decomposition of the (underlying undirected graph of the) input graph of width  $t$  solve the following problems:

- Steiner Tree in  $3^t |V|^{\mathcal{O}(1)}$
- Feedback Vertex Set in  $3^t |V|^{\mathcal{O}(1)}$
- ...

The algorithms cannot give false positives and may give false negatives with probability at most  $1/2$ .

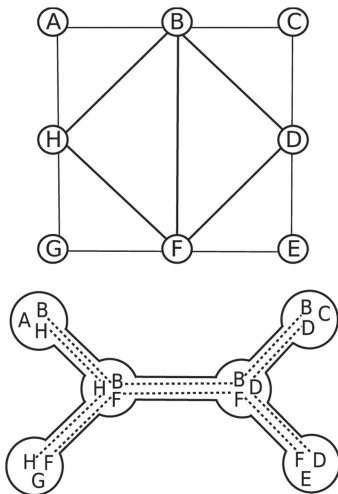
## Tree Decomposition

A tree decomposition of a graph  $G$  is a tree  $\mathbb{T}$  in which each vertex  $x \in \mathbb{T}$  has an assigned set of vertices  $B_x \subseteq V$  (called a bag) such that  $\bigcup_{x \in \mathbb{T}} B_x = V$  with the following properties:

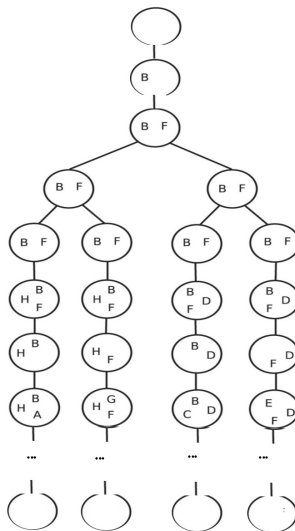
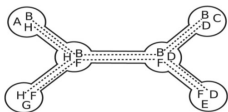
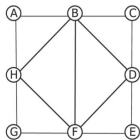
- for any  $uv \in E$ , there exists an  $x \in \mathbb{T}$  such that  $u, v \in B_x$
- if  $v \in B_x$  and  $v \in B_y$ , then  $v \in B_z$  for all  $z$  on the path from  $x$  to  $y$  in  $\mathbb{T}$

Kann in polynomieller Zeit berechnet werden (Beweis: Ton Kloks. Treewidth, Computations and Approximations, volume 842 of Lecture Notes in Computer Science. Springer, 1994)

# Einschub: Tree Decomposition (2)



# Einschub: Nice Tree Decomposition (NTD)



# Cut & Count mit Steiner Tree

## Problem

**Input:** An undirected graph  $G = (V, E)$ , a set of terminals  $T \subseteq V$  and an integer  $k$ .

**Question:** Is there a set  $X \subseteq V$  of cardinality  $k$  such that  $T \subseteq X$  and  $G[X]$  is connected?

# Cut (1)

- definiere zufällige Gewichtsfunktion  $\omega : V \rightarrow \{1, \dots, N\}$



# Cut (1)

- definiere zufällige Gewichtsfunktion  $\omega : V \rightarrow \{1, \dots, N\}$
- sei  $\mathcal{R}_W$  die Menge aller Teilmengen von  $X$  aus  $V$  mit  $T \subseteq X$ ,  
 $\omega(X) = W$  und  $|X| = k$

- definiere zufällige Gewichtsfunktion  $\omega : V \rightarrow \{1, \dots, N\}$
- sei  $\mathcal{R}_W$  die Menge aller Teilmengen von  $X$  aus  $V$  mit  $T \subseteq X$ ,  $\omega(X) = W$  und  $|X| = k$
- sei  $\mathcal{S}_W = \{X \in \mathcal{R}_W \mid G[X] \text{ ist zusammenhängend}\}$

- definiere zufällige Gewichtungsfunktion  $\omega : V \rightarrow \{1, \dots, N\}$
- sei  $\mathcal{R}_W$  die Menge aller Teilmengen von  $X$  aus  $V$  mit  $T \subseteq X$ ,  $\omega(X) = W$  und  $|X| = k$
- sei  $\mathcal{S}_W = \{X \in \mathcal{R}_W \mid G[X] \text{ ist zusammenhängend}\}$
- $\cup_W \mathcal{S}_W$  ist die Menge der Lösungen

- definiere zufällige Gewichtungsfunktion  $\omega : V \rightarrow \{1, \dots, N\}$
- sei  $\mathcal{R}_W$  die Menge aller Teilmengen von  $X$  aus  $V$  mit  $T \subseteq X$ ,  $\omega(X) = W$  und  $|X| = k$
- sei  $\mathcal{S}_W = \{X \in \mathcal{R}_W \mid G[X] \text{ ist zusammenhängend}\}$
- $\cup_W \mathcal{S}_W$  ist die Menge der Lösungen
- gibt es ein  $W$  für das die Menge nichtleer ist, so gibt der Algorithmus eine positive Antwort

# Cut (2)

- einen beliebigen Terminalknoten  $v \in T$  als  $v_1$  festlegen

## Cut (2)

- einen beliebigen Terminalknoten  $v \in T$  als  $v_1$  festlegen
- sei  $\mathcal{C}_W$  die Menge aller Subgraphen, die einen konsistenten Cut  $(X, (X_1, X_2))$  bilden, wobei  $X \in \mathcal{R}_W$  und  $v_1 \in X_1$

- einen beliebigen Terminalknoten  $v \in T$  als  $v_1$  festlegen
- sei  $\mathcal{C}_W$  die Menge aller Subgraphen, die einen konsistenten Cut  $(X, (X_1, X_2))$  bilden, wobei  $X \in \mathcal{R}_W$  und  $v_1 \in X_1$

### Lemma 3.3

Let  $G = (V, E)$  be a graph and let  $X$  be a subset of vertices such that  $v_1 \in X \subseteq V$ . The number of consistently cut subgraphs  $(X, (X_1, X_2))$  such that  $v_1 \in X_1$  is equal to  $2^{cc(G[X])-1}$ .



# Cut (3) - wozu $\omega$ ?

## Cut (3) - wozu $\omega$ ?

### Isolation Lemma: Definition 2.4

A function  $\omega : U \rightarrow \mathbb{Z}$  isolates a set family  $\mathcal{F} \subseteq 2^U$  if there is a unique  $S' \in \mathcal{F}$  with  $\omega(S') = \min_{S \in \mathcal{F}} \omega(S)$

## Cut (3) - wozu $\omega$ ?

### Isolation Lemma: Definition 2.4

A function  $\omega : U \rightarrow \mathbb{Z}$  isolates a set family  $\mathcal{F} \subseteq 2^U$  if there is a unique  $S' \in \mathcal{F}$  with  $\omega(S') = \min_{S \in \mathcal{F}} \omega(S)$

### Isolation Lemma: Lemma 2.5

Let  $\mathcal{F} \subseteq 2^U$  be a set family over a universe  $U$  with  $|\mathcal{F}| > 0$ . For each  $u \in U$ , choose a weight  $\omega(u) \in 1, 2, \dots, N$  uniformly and independently at random. Then

$$\text{prob}[\omega \text{ isolates } \mathcal{F}] \geq 1 - \frac{|U|}{N}$$

## Cut (3) - wozu $\omega$ ?

### Isolation Lemma: Definition 2.4

A function  $\omega : U \rightarrow \mathbb{Z}$  isolates a set family  $\mathcal{F} \subseteq 2^U$  if there is a unique  $S' \in \mathcal{F}$  with  $\omega(S') = \min_{S \in \mathcal{F}} \omega(S)$

### Isolation Lemma: Lemma 2.5

Let  $\mathcal{F} \subseteq 2^U$  be a set family over a universe  $U$  with  $|\mathcal{F}| > 0$ . For each  $u \in U$ , choose a weight  $\omega(u) \in 1, 2, \dots, N$  uniformly and independently at random. Then

$$\text{prob}[\omega \text{ isolates } \mathcal{F}] \geq 1 - \frac{|U|}{N}$$

- der Algorithmus (ohne  $\omega$ ) ist korrekt, sofern es **genau eine** oder **keine** Lösung gibt

## Cut (3) - wozu $\omega$ ?

### Isolation Lemma: Definition 2.4

A function  $\omega : U \rightarrow \mathbb{Z}$  isolates a set family  $\mathcal{F} \subseteq 2^U$  if there is a unique  $S' \in \mathcal{F}$  with  $\omega(S') = \min_{S \in \mathcal{F}} \omega(S)$

### Isolation Lemma: Lemma 2.5

Let  $\mathcal{F} \subseteq 2^U$  be a set family over a universe  $U$  with  $|\mathcal{F}| > 0$ . For each  $u \in U$ , choose a weight  $\omega(u) \in 1, 2, \dots, N$  uniformly and independently at random. Then

$$\text{prob}[\omega \text{ isolates } \mathcal{F}] \geq 1 - \frac{|U|}{N}$$

- der Algorithmus (ohne  $\omega$ ) ist korrekt, sofern es **genau eine** oder **keine** Lösung gibt
- mithilfe des Isolation Lemma kann mit hoher Wahrscheinlichkeit eine große Lösungsmenge auf eine einzige reduziert werden

# Count (1)

- aus Lemma 3.3 ist bekannt:  $|\mathcal{C}| = \sum_{X \in \mathcal{R}} 2^{\text{cc}(G[X])-1}$

- aus Lemma 3.3 ist bekannt:  $|\mathcal{C}| = \sum_{X \in \mathcal{R}} 2^{\text{cc}(G[X])-1}$
- wir legen  $W$  fest und ignorieren die Indices:  
 $|\mathcal{C}| \equiv |\{X \in \mathcal{R} | \text{cc}(G[X]) = 1\}| = |\mathcal{S}|$



- aus Lemma 3.3 ist bekannt:  $|\mathcal{C}| = \sum_{X \in \mathcal{R}} 2^{\text{cc}(G[X])-1}$
- wir legen  $W$  fest und ignorieren die Indices:  
 $|\mathcal{C}| \equiv |\{X \in \mathcal{R} \mid \text{cc}(G[X]) = 1\}| = |\mathcal{S}|$

## Lemma 3.4

Let  $G$ ,  $\omega$ ,  $\mathcal{C}_W$  and  $\mathcal{S}_W$  be as defined above. Then for every  $W$ ,  
 $|\mathcal{S}_W| \equiv |\mathcal{C}_W|$ .

# Count (2)

- $|\mathcal{C}_W|$  modulo 2 kann mit dynamischen Programm auf der NTD  $\mathbb{T}$  berechnet werden

## Count (2)

- $|\mathcal{C}_W|$  modulo 2 kann mit dynamischen Programm auf der NTD  $\mathbb{T}$  berechnet werden
- für jeden Bag  $x \in \mathbb{T}$ , integers  $0 \leq i \leq k, 0 \leq w \leq kN$  und Färbung  $s \in \{0, 1_1, 1_2\}^{B_x}$  definiere:

## Count (2)

- $|\mathcal{C}_W|$  modulo 2 kann mit dynamischen Programm auf der NTD  $\mathbb{T}$  berechnet werden
- für jeden Bag  $x \in \mathbb{T}$ , integers  $0 \leq i \leq k, 0 \leq w \leq kN$  und Färbung  $s \in \{0, 1_1, 1_2\}^{B_x}$  definiere:
  - $\mathcal{R}_x(i, w) = \{X \subseteq V_x \mid (T \cap V_x) \subseteq X \wedge |X| = i \wedge \omega(X) = w\}$

- $|\mathcal{C}_W|$  modulo 2 kann mit dynamischen Programm auf der NTD  $\mathbb{T}$  berechnet werden
- für jeden Bag  $x \in \mathbb{T}$ , integers  $0 \leq i \leq k, 0 \leq w \leq kN$  und Färbung  $s \in \{0, 1_1, 1_2\}^{B_x}$  definiere:
  - $\mathcal{R}_x(i, w) = \{X \subseteq V_x \mid (T \cap V_x) \subseteq X \wedge |X| = i \wedge \omega(X) = w\}$
  - $\mathcal{C}_x(i, w) = \{(X, (X_1, X_2)) \mid X \in \mathcal{R}_x(i, w) \wedge (X, (X_1, X_2)) \text{ is a consistently cut subgraph of } G_x \wedge (v_1 \in V_x \Rightarrow v_1 \in X_1)\}$

- $|\mathcal{C}_W|$  modulo 2 kann mit dynamischen Programm auf der NTD  $\mathbb{T}$  berechnet werden
- für jeden Bag  $x \in \mathbb{T}$ , integers  $0 \leq i \leq k, 0 \leq w \leq kN$  und Färbung  $s \in \{0, 1_1, 1_2\}^{B_x}$  definiere:
  - $\mathcal{R}_x(i, w) = \{X \subseteq V_x \mid (\mathcal{T} \cap V_x) \subseteq X \wedge |X| = i \wedge \omega(X) = w\}$
  - $\mathcal{C}_x(i, w) = \{(X, (X_1, X_2)) \mid X \in \mathcal{R}_x(i, w) \wedge (X, (X_1, X_2)) \text{ is a consistently cut subgraph of } G_x \wedge (v_1 \in V_x \Rightarrow v_1 \in X_1)\}$
  - $\mathcal{A}_x(i, w, s) = |\{(X, (X_1, X_2)) \in \mathcal{C}_x(i, w) \mid (s(v) = 1_j \Rightarrow v \in X_j) \wedge (s(v) = 0 \Rightarrow v \notin X)\}|$

- $|\mathcal{C}_W|$  modulo 2 kann mit dynamischen Programm auf der NTD  $\mathbb{T}$  berechnet werden
- für jeden Bag  $x \in \mathbb{T}$ , integers  $0 \leq i \leq k, 0 \leq w \leq kN$  und Färbung  $s \in \{0, 1_1, 1_2\}^{B_x}$  definiere:
  - $\mathcal{R}_x(i, w) = \{X \subseteq V_x \mid (\mathcal{T} \cap V_x) \subseteq X \wedge |X| = i \wedge \omega(X) = w\}$
  - $\mathcal{C}_x(i, w) = \{(X, (X_1, X_2)) \mid X \in \mathcal{R}_x(i, w) \wedge (X, (X_1, X_2)) \text{ is a consistently cut subgraph of } G_x \wedge (v_1 \in V_x \Rightarrow v_1 \in X_1)\}$
  - $\mathcal{A}_x(i, w, s) = |\{(X, (X_1, X_2)) \in \mathcal{C}_x(i, w) \mid (s(v) = 1_j \Rightarrow v \in X_j) \wedge (s(v) = 0 \Rightarrow v \notin X)\}|$
- Färbung  $s \in \{0, 1_1, 1_2\}^{B_x}$  der Knoten aus  $B_x$  bzgl. der Menge  $\mathcal{C}_x$ 
  - $s[v] = 0 \Rightarrow v \notin X$
  - $s[v] = 1_1 \Rightarrow v \in X_1$
  - $s[v] = 1_2 \Rightarrow v \in X_2$



# Count (3)

- $\mathcal{A}_x(i, w, s)$  zählt verschiedenen Färbungen  $\mathcal{C}_x(i, w)$ , die entsprechend gefärbt sind

- $\mathcal{A}_x(i, w, s)$  zählt verschiedenen Färbungen  $\mathcal{C}_x(i, w)$ , die entsprechend gefärbt sind
- $$\sum_{s \in \{0,1_1,1_2\}^{B_x}} A_x(i, w, s) = |\mathcal{C}_x(i, w)|$$

- $\mathcal{A}_x(i, w, s)$  zählt verschiedenen Färbungen  $\mathcal{C}_x(i, w)$ , die entsprechend gefärbt sind
- $$\sum_{s \in \{0, 1_1, 1_2\}^{B_x}} A_x(i, w, s) = |\mathcal{C}_x(i, w)|$$
- für die Lösung nur  $A_r(k, W, \emptyset) = |\mathcal{C}_r(k, W)| = |\mathcal{C}_W|$  interessant (für alle  $W$ )

- $\mathcal{A}_x(i, w, s)$  zählt verschiedenen Färbungen  $\mathcal{C}_x(i, w)$ , die entsprechend gefärbt sind
- $$\sum_{s \in \{0,1_1,1_2\}^{B_x}} A_x(i, w, s) = |\mathcal{C}_x(i, w)|$$
- für die Lösung nur  $A_r(k, W, \emptyset) = |\mathcal{C}_r(k, W)| = |\mathcal{C}_W|$  interessant (für alle  $W$ )
- ausgehend von Wurzel-Knoten rekursiver Abstieg zu den Blatt-Knoten der NTD

## Theorem 3.6

There exists a Monte-Carlo algorithm that given a tree decomposition of width  $t$  solves STEINER TREE in  $3^t |V|^{\mathcal{O}(1)}$  time. The algorithm cannot give false positives and may give false negatives with probability at most  $1/2$ .

## Theorem 3.6

There exists a Monte-Carlo algorithm that given a tree decomposition of width  $t$  solves STEINER TREE in  $3^t |V|^{\mathcal{O}(1)}$  time. The algorithm cannot give false positives and may give false negatives with probability at most  $1/2$ .

- $3^t$  kommt von 3 Färbungen im dynamischen Programm

## Theorem 3.6

There exists a Monte-Carlo algorithm that given a tree decomposition of width  $t$  solves STEINER TREE in  $3^t |V|^{\mathcal{O}(1)}$  time. The algorithm cannot give false positives and may give false negatives with probability at most  $1/2$ .

- $3^t$  kommt von 3 Färbungen im dynamischen Programm
- $|V|^{\mathcal{O}(1)}$  kommt aus  $k$  und  $N$  als Input



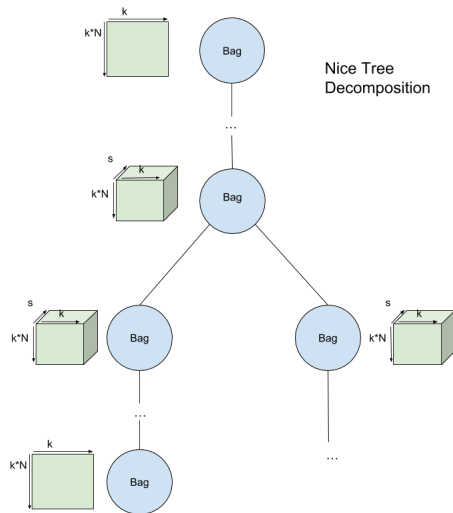
## Theorem 3.6

There exists a Monte-Carlo algorithm that given a tree decomposition of width  $t$  solves STEINER TREE in  $3^t |V|^{\mathcal{O}(1)}$  time. The algorithm cannot give false positives and may give false negatives with probability at most  $1/2$ .

- $3^t$  kommt von 3 Färbungen im dynamischen Programm
- $|V|^{\mathcal{O}(1)}$  kommt aus  $k$  und  $N$  als Input
- Wahrscheinlichkeit  $1/2$  durch Gewichtsfunktion  $\omega : V \rightarrow 1, \dots, N$  und Isolation Lemma

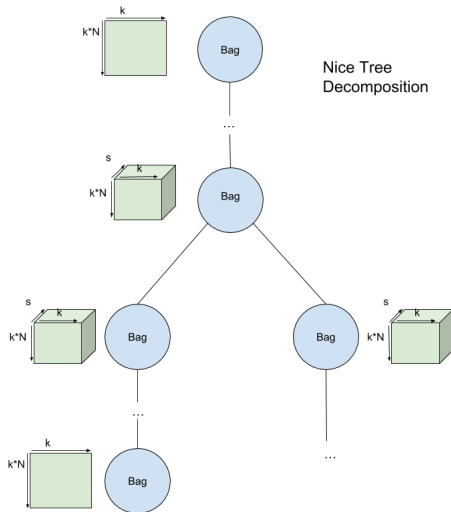
# Implementierung

# Implementierung (1)



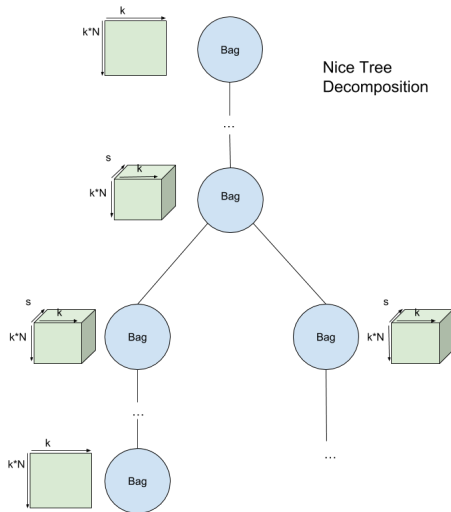
# Implementierung (1)

- Inorder-Traversierung der NTD



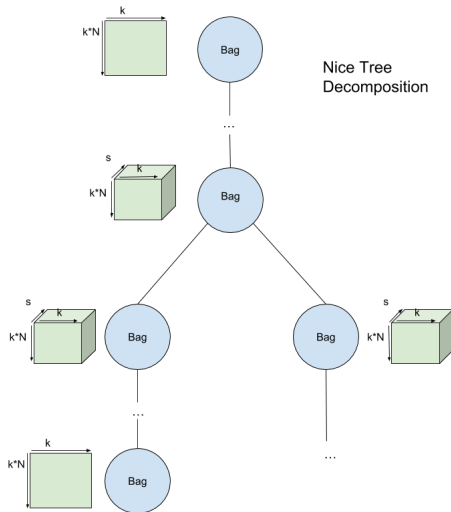
# Implementierung (1)

- Inorder-Traversierung der NTD
- Berechnung einer Datenmatrix basierend auf dynamischen Programm und Kind-Knoten



# Implementierung (1)

- Inorder-Traversierung der NTD
- Berechnung einer Datenmatrix basierend auf dynamischen Programm und Kind-Knoten
- Wurzel- und alle Blattknoten enthalten 2D Datenmatrix



# Implementierung (2)

- Entwicklung einer Umformung von TD  $\rightarrow$  NTD

# Implementierung (2)

- Entwicklung einer Umformung von TD  $\rightarrow$  NTD
- anpassen der Färbungs-Dimension mittels ternärer Kodierung



# Implementierung (2)

- Entwicklung einer Umformung von TD  $\rightarrow$  NTD
- anpassen der Färbungs-Dimension mittels ternärer Kodierung
- Tests mit verschiedenen Input-Größen

Input	$\emptyset \text{ T in } s$
( $k=2$ , $N=6$ , $ V =3$ )	$\sim 0.002$
( $k=3$ , $N=14$ , $ V =7$ )	$\sim 0.83$
( $k=4$ , $N=32$ , $ V =16$ )	$\sim 14.23$

# Implementierung (2)

- Entwicklung einer Umformung von TD  $\rightarrow$  NTD
- anpassen der Färbungs-Dimension mittels ternärer Kodierung
- Tests mit verschiedenen Input-Größen
- Vorteil:
  - $\mathcal{O}(1)$  Zugriffszeit für alle  $i < k$
  - Wahrscheinlichkeit über N steuerbar, aber Datenmatrix wächst schnell !

Input	$\emptyset$ T in s
(k=2, N= 6,  V =3)	$\sim 0.002$
(k=3, N=14,  V =7)	$\sim 0.83$
(k=4, N=32,  V =16)	$\sim 14.23$

# Fragen?



Marek Cygan and Jesper Nederlof and Marcin Pilipczuk and Michal Pilipczuk and Johan M. M. van Rooij and Jakub Onufry Wojtaszczyk (2011)

Solving connectivity problems parameterized by treewidth in single exponential time

*CoRR* abs/1103.0534.

## Berechnungen

- **Leaf bag:**

- $A_x = (0, 0, \emptyset) = 1$

- **Introduce vertex  $v$ :**

- $A_x = (i, w, s[v \rightarrow 0]) = [v \notin T] A_y(i, w, s)$

- $A_x = (i, w, s[v \rightarrow 1_1]) = A_y(i - 1, w - w(v), s)$

- $A_x = (i, w, s[v \rightarrow 1_2]) = [v \neq v_1] A_y(i - 1, w - w(v), s)$

- **Introduce edge  $uv$**

- $A_x(i, w, s) = [s(u) = 0 \vee s(v) = 0 \vee s(u) = s(v)] A_y(i, w, s)$

- **Forget vertex  $v$**

- $A_x(i, w, s) = \sum_{\alpha \in \{0, 1_1, 1_2\}} A_x(i, w, s[v \rightarrow \alpha])$

- **Join bag**

- $A_x(i, w, s) = \sum_{i_1 + i_2 = i + |s^{-1}(1_1, 1_2)|} \sum_{w_1 + w_2 = w + w(s^{-1}(1_1, 1_2))} A_y(i_1, w_1, s) A_z(i_2, w_2, s)$