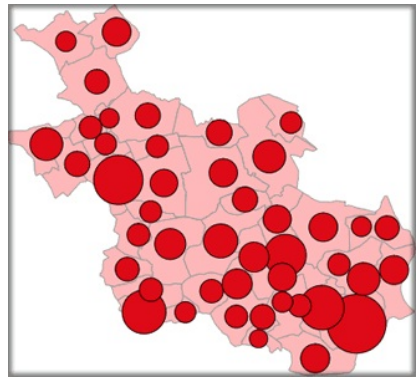


Proportional Symbol Maps

Florian Simon

8. Dezember, 2009

Proportional Symbol Maps

- Gegeben: Punkte $p_1, \ldots, p_n \in \mathbb{R}^2$ mit zugeordneten Werten $w_1, \ldots, w_n \in \mathbb{R}$

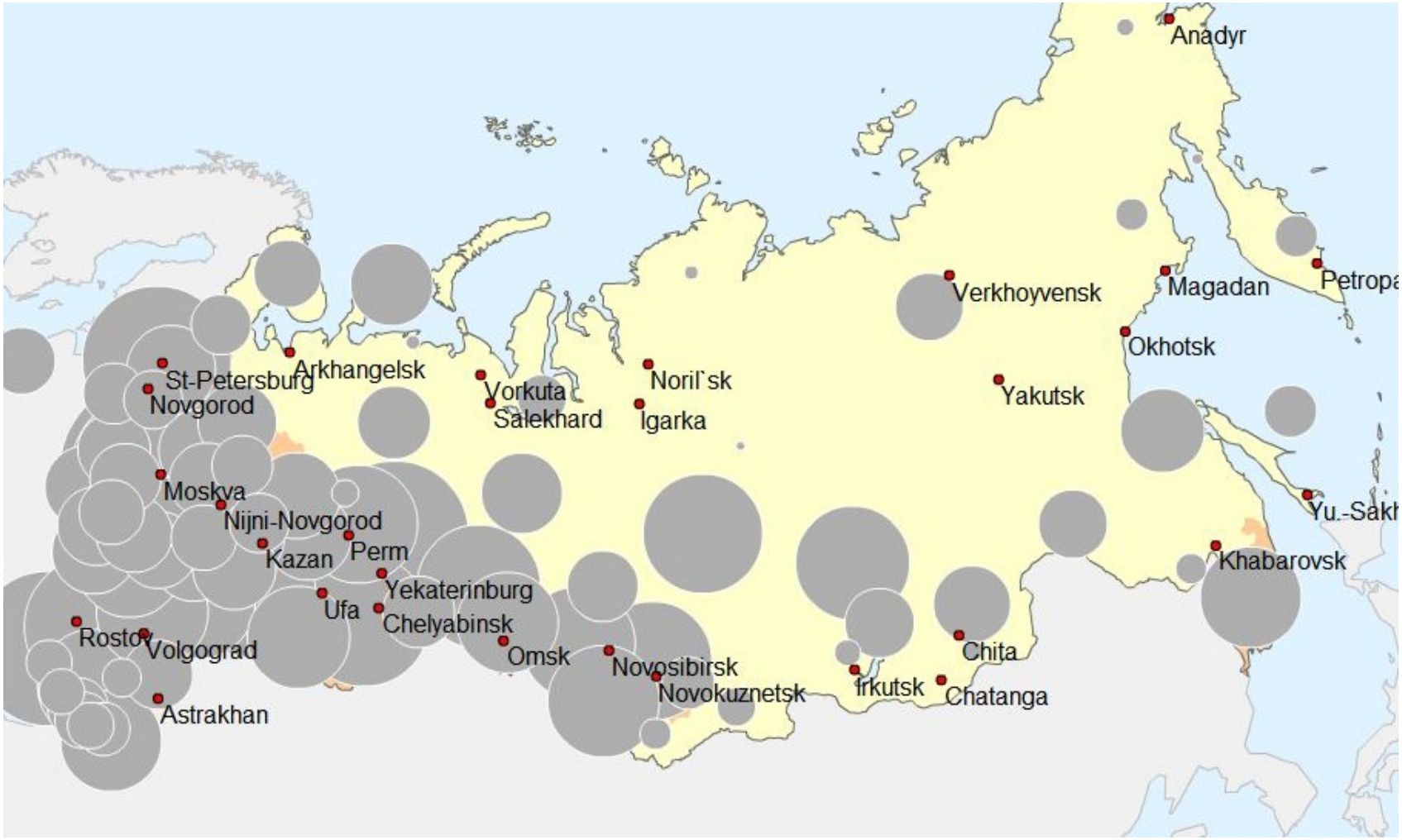
Proportional Symbol Maps

- Gegeben: Punkte $p_1, \ldots, p_n \in \mathbb{R}^2$ mit zugeordneten Werten $w_1, \ldots, w_n \in \mathbb{R}$
- Beispiel:
  - p_i Ort der Rathäuser eines Landes
  - w_i Anzahl dort gemeldeter Einwohner

Proportional Symbol Maps

- Gegeben: Punkte $p_1, \ldots, p_n \in \mathbb{R}^2$ mit zugeordneten Werten $w_1, \ldots, w_n \in \mathbb{R}$
- Beispiel:
  - p_i Ort der Rathäuser eines Landes
  - w_i Anzahl dort gemeldeter Einwohner
- Visualisierung mithilfe geometischer Objekte

Proportional Symbol Maps



Proportional Symbol Maps

Ziel:

Eine gültige und

möglichst gute

Zeichnung

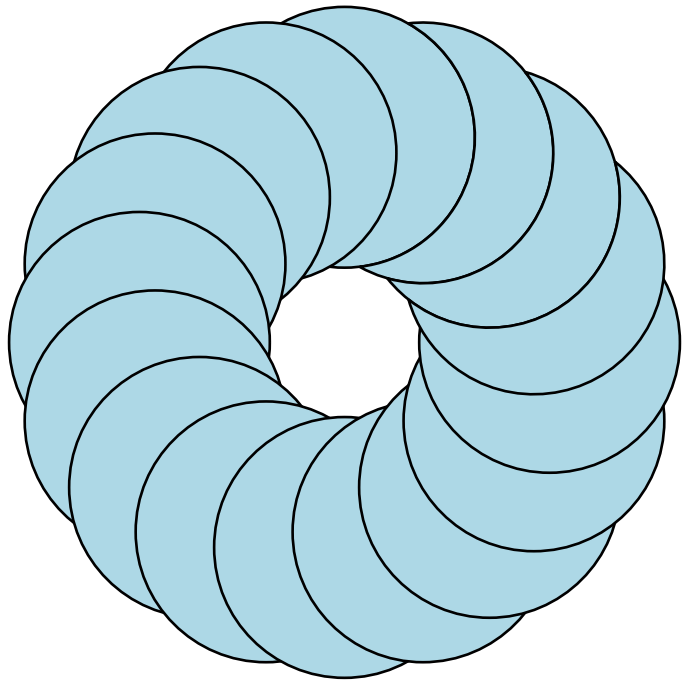
Proportional Symbol Maps

Ziel:

Eine gültige und	$\Rightarrow$ zwei Klassen von Zeichnungen
möglichst gute Zeichnung	$\Rightarrow$ zwei Optimierungsprobleme

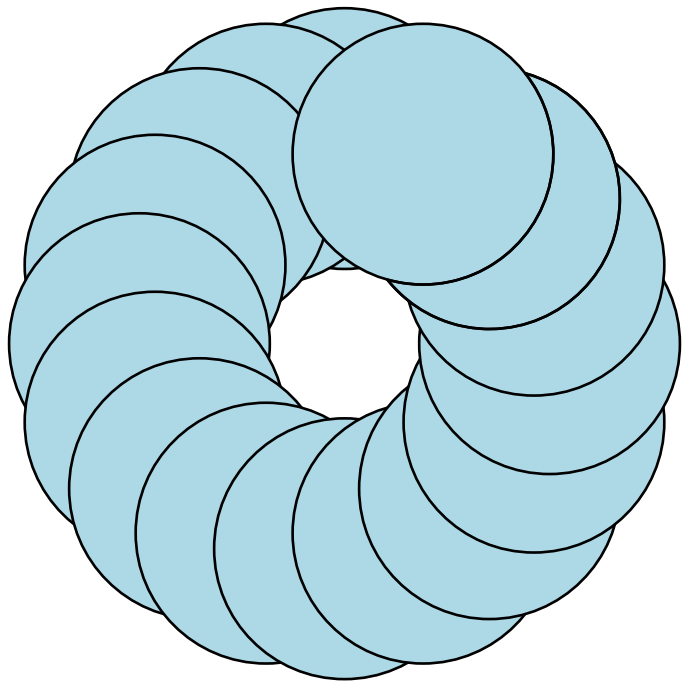
physikalisch-realisierbare Zeichnungen

Eine Zeichnung heißt physikalisch-realisierbar, wenn sie mithilfe von Münzen nachgebaut werden kann.



Stackings

Eine Zeichnung S heißt Stacking, wenn es eine totale Ordnung unter allen Disks gibt, sodass S entsteht in dem man die Disks gemäß dieser Ordnung aufeinander stapelt.



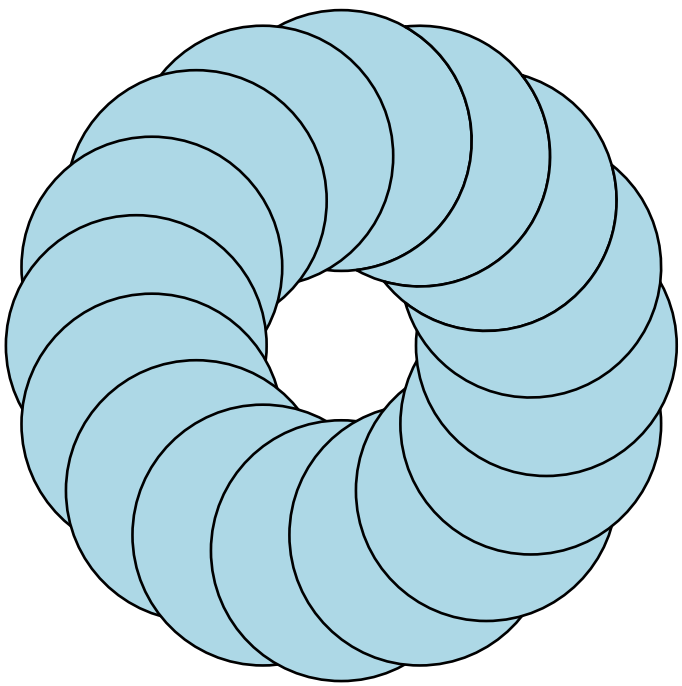
Zwei Optimierungsprobleme

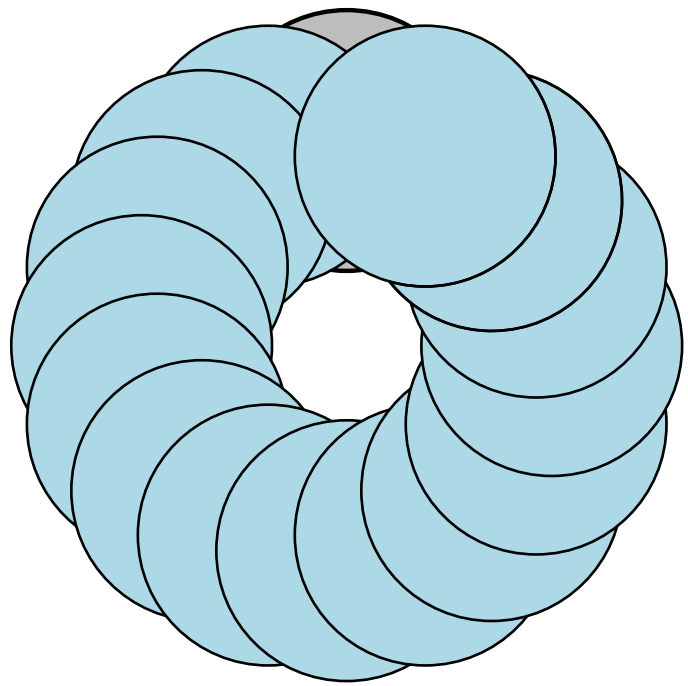
Gegeben eine Menge von Disks S

- Max-Min: Finde eine physikalisch-realisierbare Zeichnung (ein Stacking), die den minimal sichtbaren Rand unter allen Disks maximiert.
- Max-Total: Finde eine physikalisch-realisierbare Zeichnung (ein Stacking), die den insgesamt sichtbaren Rand aller Disks maximiert.

- Der in einer Zeichnung minimal sichtbare Rand unter allen Disks aus S wird im Folgenden mit $\min(S)$ bezeichnet.

- Der in einer Zeichnung minimal sichtbare Rand unter allen Disks aus S wird im Folgenden mit $\min(S)$ bezeichnet.
- $\min(S)$ kann für physikalisch-realisierbare Zeichnungen größer sein als für Stackings. Beispiel:





Überblick

	Max-Min	Max-Total
physikalisch realisierbar		
Stacking		

Überblick

	Max-Min	Max-Total
physikalisch realisierbar	$\mathcal{NP}$-schwer	
Stacking		

planares 3-*SAT* (*p*-3-*SAT*)

- Aussagenlogische Formel F mit Variablen $\{x_1, \ldots, x_n\}$ und Klauseln $\{c_1, \ldots, c_m\}$

planares 3-*SAT* (*p*-3-*SAT*)

- Aussagenlogische Formel F mit Variablen $\{x_1, \ldots, x_n\}$ und Klauseln $\{c_1, \ldots, c_m\}$

- $G(F) := (V, E)$ mit $V = \{p_1, \ldots, p_n, q_1, \ldots, q_m\}$ und $(p_i, q_j) \in E$ genau dann, wenn x_i in c_j vorkommt $(i = 1, \ldots, n; j = 1, \ldots, m)$

planares 3-*SAT* (*p*-3-*SAT*)

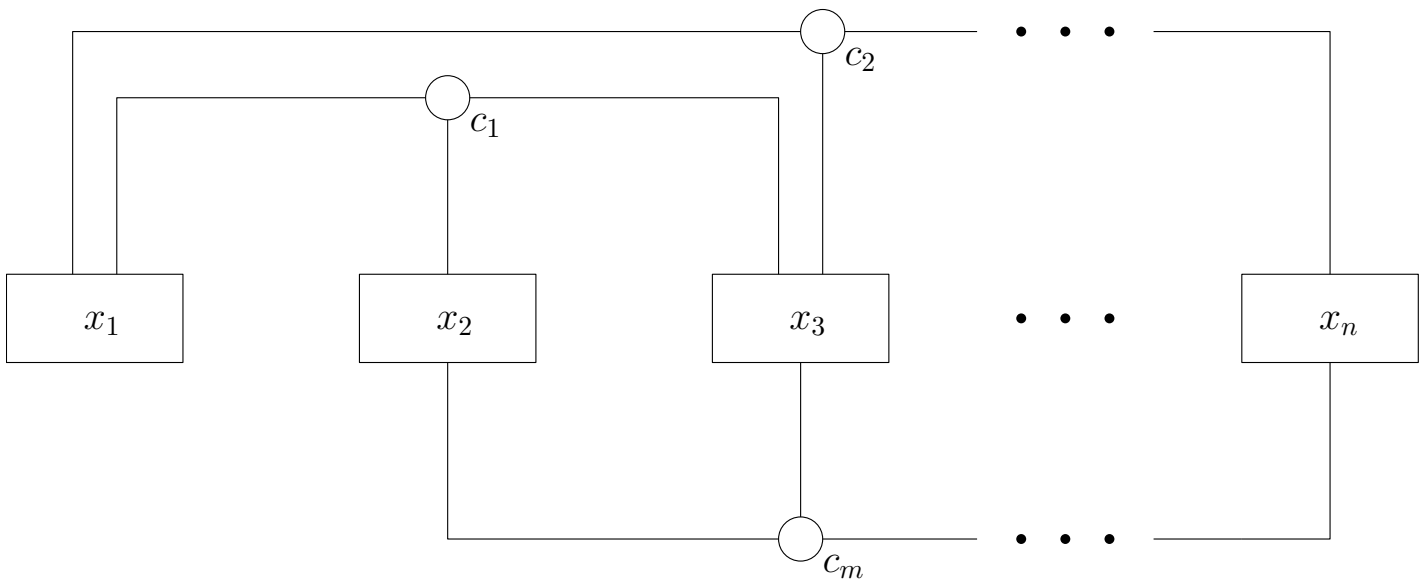
- Aussagenlogische Formel F mit Variablen $\{x_1, \ldots, x_n\}$ und Klauseln $\{c_1, \ldots, c_m\}$
- $G(F) := (V, E)$ mit $V = \{p_1, \ldots, p_n, q_1, \ldots, q_m\}$ und $(p_i, q_j) \in E$ genau dann, wenn x_i in c_j vorkommt $(i = 1, \ldots, n; j = 1, \ldots, m)$
- $F \in p\text{-3-}SAT \Leftrightarrow F \in 3\text{-}SAT$ und $G(F)$ planar

planares 3-*SAT* (*p*-3-*SAT*)

- planares 3-*SAT* ist $\mathcal{NP}$-schwer (ohne Beweis)

planares 3-*SAT* (*p*-3-*SAT*)

- planares 3-*SAT* ist $\mathcal{NP}$-schwer (ohne Beweis)
- Jeder Graph zu einer *p*-3-*SAT* Instanz kann nach folgendem Schema gezeichnet werden:



Max-Min - $\mathcal{NP}$-schwer

- Das Max-Min Problem ist für physikalisch-realisierbare Zeichnungen $\mathcal{NP}$-schwer.
- Vorgehen: Zu einer Instanz $\mathcal{I} \in p\text{-}3\text{-}SAT$ konstruieren wir eine Menge von Disks S so, dass $\mathcal{I}$ genau dann erfüllbar ist, wenn es eine physikalisch-realisierbare Zeichnung von S mit $\min(S) \geq \frac{3}{4}$ gibt.

Konstruktion

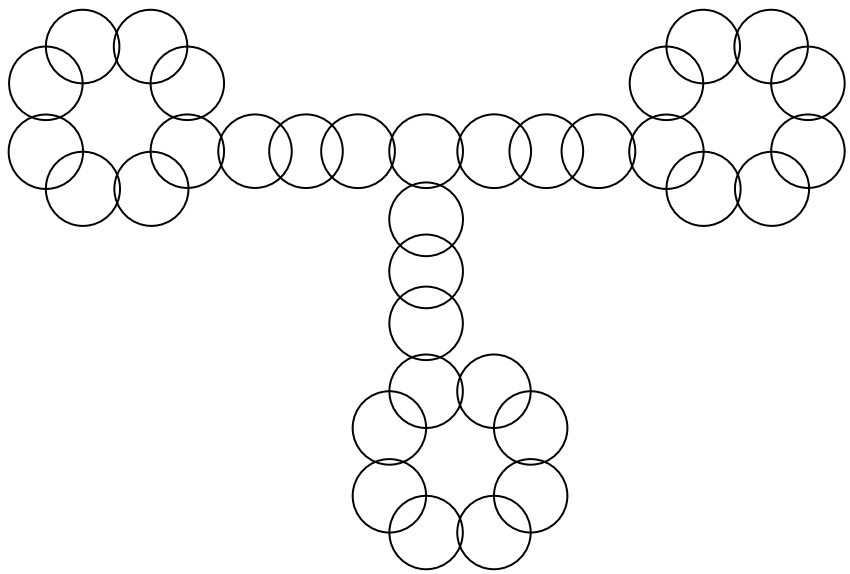
S besteht aus Disks mit Umfang 1 die sich in folgende Komponenten unterteilen lassen:

- Variablen-Gadgets - für jede in $\mathcal{I}$ vorkommende Variable
- Klausel-Gadgets - für jede in $\mathcal{I}$ vorkommende Klausel
- Kanäle - repräsentieren die in einer Klausel vorkommenden Literale

Konstruktion

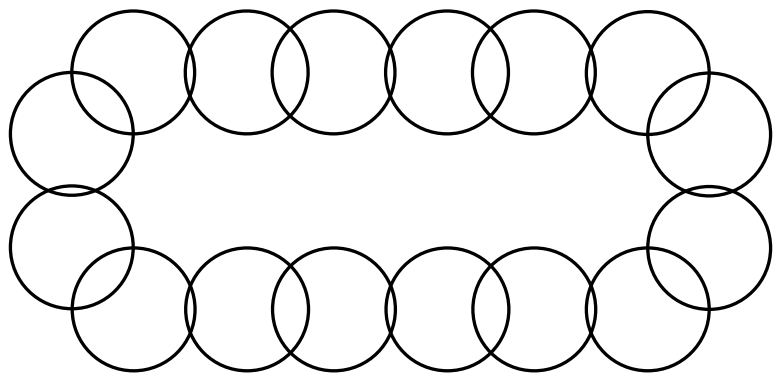
S besteht aus Disks mit Umfang 1 die sich in folgende Komponenten unterteilen lassen:

- Variablen-Gadgets - für jede in $\mathcal{I}$ vorkommende Variable
- Klausel-Gadgets - für jede in $\mathcal{I}$ vorkommende Klausel
- Kanäle - repräsentieren die in einer Klausel vorkommenden Literale



Variablen-Gadgets

Für jede Variable x_i aus $\mathcal{I}$. Zyklisch angeordnete Menge von Disks $S(x_i)$ mit alternierenden Überlappungen von $\frac{1}{4}$ und $\frac{1}{8}$

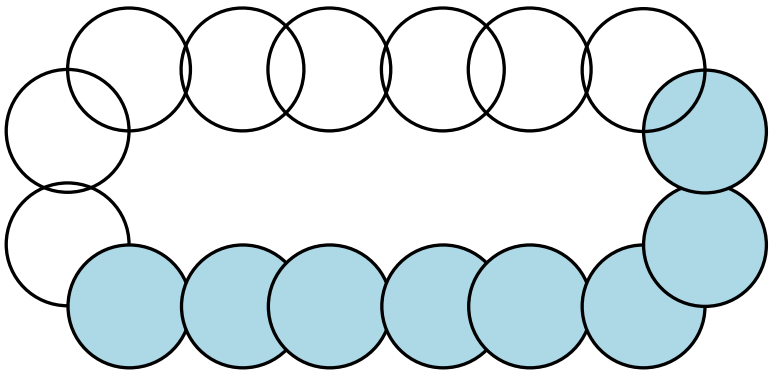


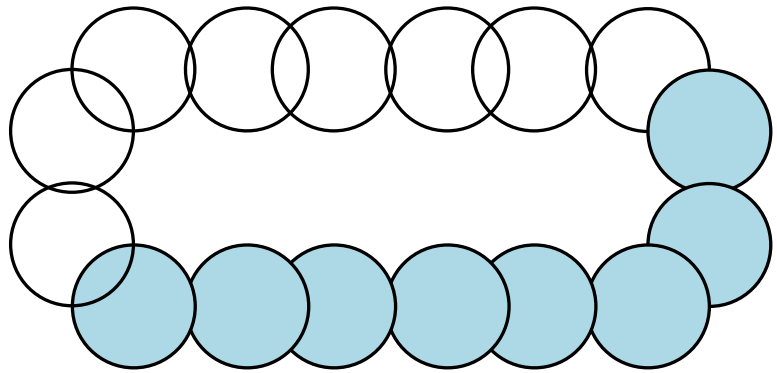
Variablen-Gadgets

Für $\min(S) \geq \frac{3}{4}$ muss dann jede Disk zwischen ihren Nachbarn liegen.

Variablen-Gadgets

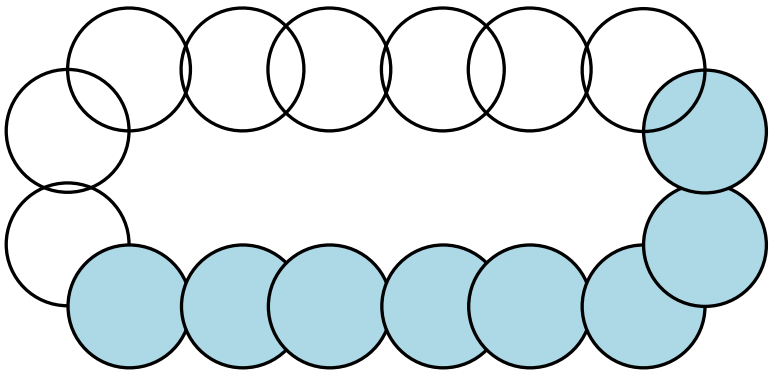
Für $\min(S) \geq \frac{3}{4}$ muss dann jede Disk zwischen ihren Nachbarn liegen.

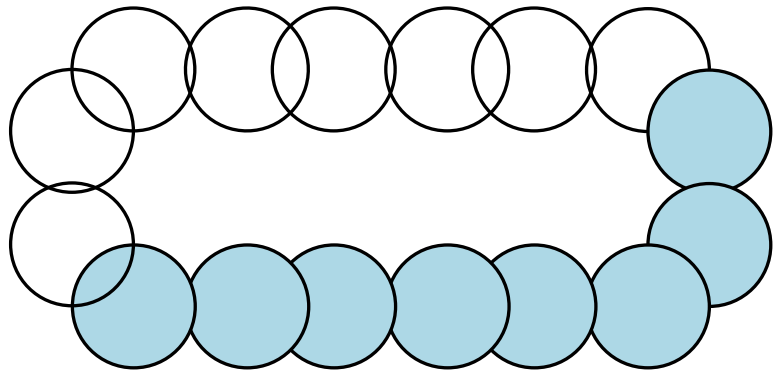




Variablen-Gadgets

Für $\min(S) \geq \frac{3}{4}$ muss dann jede Disk zwischen ihren Nachbarn liegen.

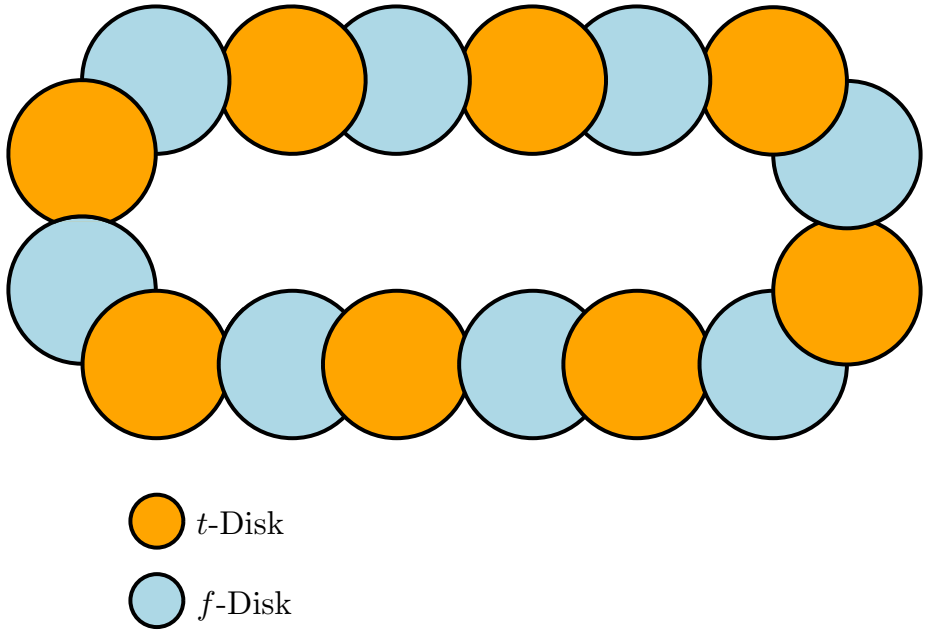




- Überlappungen im Uhrzeigersinn ($S(x_i) = \mathit{TRUE}$) entspricht $x_i = \mathit{TRUE}$
- Überlappungen gegen den Uhrzeigersinn ($S(x_i) = \mathit{FALSE}$) entspricht $x_i = \mathit{FALSE}$

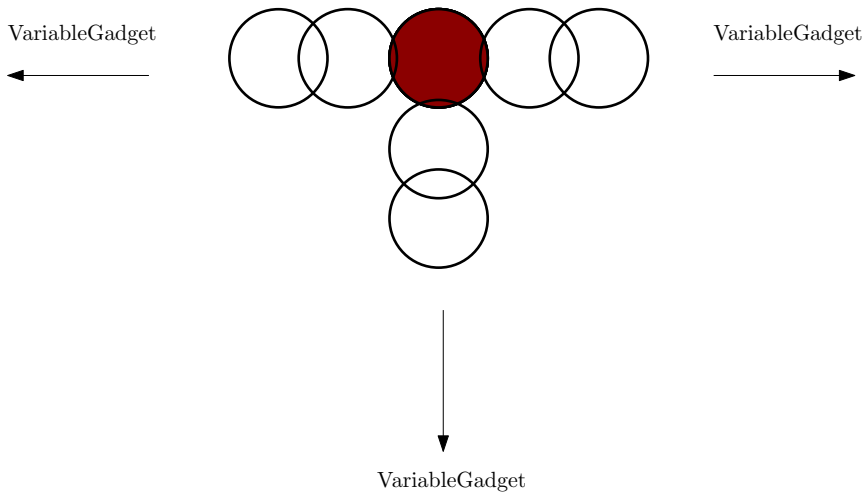
Variablen-Gadgets

Disks deren Rand für $S(x_i) = \mathit{TRUE}$ nur zu $\frac{1}{8}$ überdeckt sind heißen t-Disks die anderen f-Disks



Klausel-Gadgets

Ein Klausel-Gadget besteht aus einer Disk, die drei Kanalenden zu je $\frac{1}{8}$ überlappt.



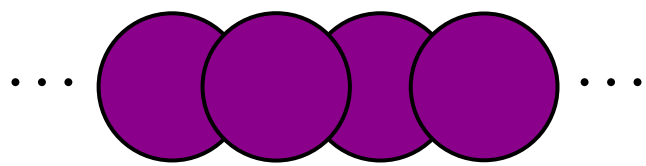
Damit $\min(S) \geq \frac{3}{4}$ ist, muss eine Disk der Kanalenden von der Klausel-Disk überlappt werden.

Kanäle

- Kette von sich jeweils zu $\frac{1}{4}$ überlappenden Disks
- Repräsentieren die Literale in den Klauseln
- Ein Kanal beginnt an einem Variablen-Gadget und endet an einem Klausel-Gadget
- Für $min(S) \geq \frac{3}{4}$ muss jede Disks zwischen ihren Nachbarn liegen

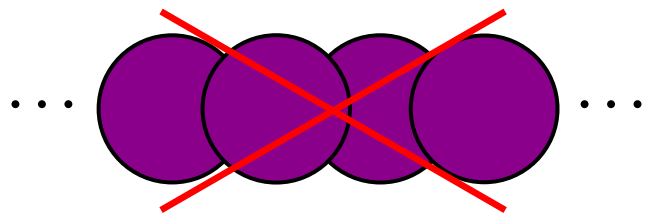
Kanäle

- Kette von sich jeweils zu $\frac{1}{4}$ überlappenden Disks
- Repräsentieren die Literale in den Klauseln
- Ein Kanal beginnt an einem Variablen-Gadget und endet an einem Klausel-Gadget
- Für $min(S) \geq \frac{3}{4}$ muss jede Disks zwischen ihren Nachbarn liegen



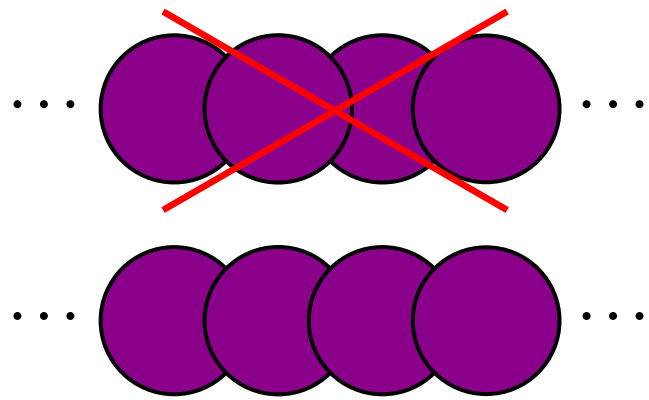
Kanäle

- Kette von sich jeweils zu $\frac{1}{4}$ überlappenden Disks
- Repräsentieren die Literale in den Klauseln
- Ein Kanal beginnt an einem Variablen-Gadget und endet an einem Klausel-Gadget
- Für $min(S) \geq \frac{3}{4}$ muss jede Disks zwischen ihren Nachbarn liegen



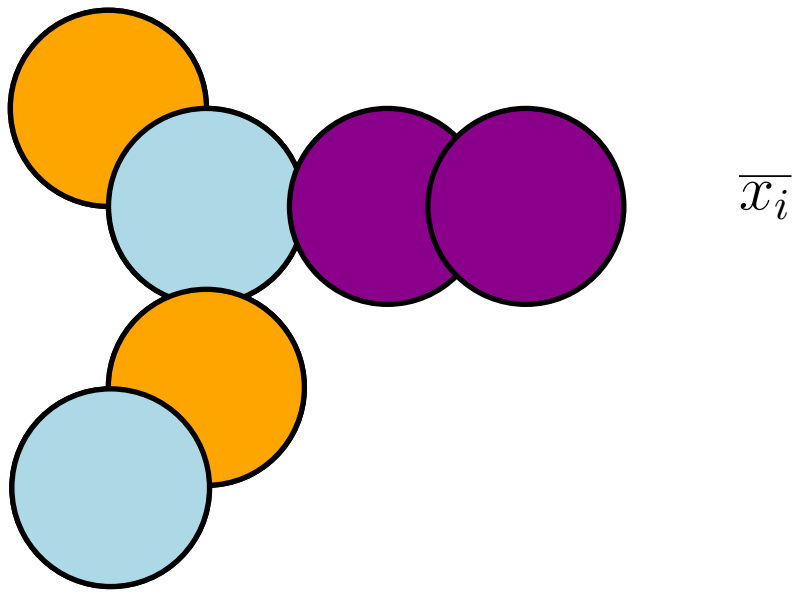
Kanäle

- Kette von sich jeweils zu $\frac{1}{4}$ überlappenden Disks
- Repräsentieren die Literale in den Klauseln
- Ein Kanal beginnt an einem Variablen-Gadget und endet an einem Klausel-Gadget
- Für $min(S) \geq \frac{3}{4}$ muss jede Disks zwischen ihren Nachbarn liegen



Kanäle

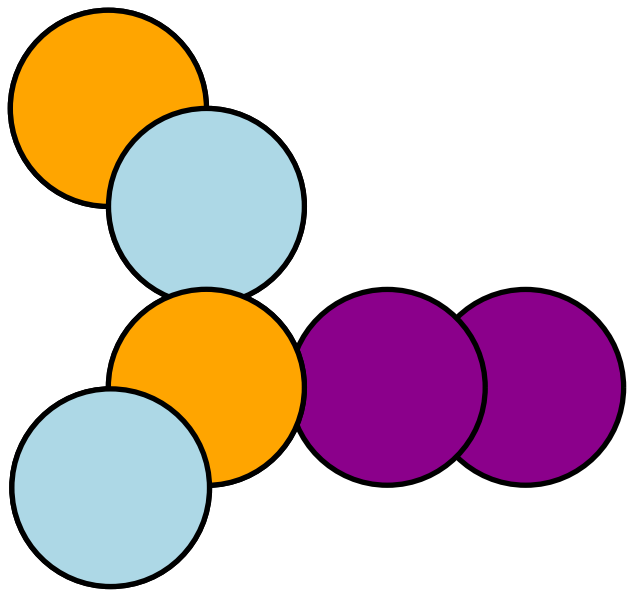
Repräsentiert ein Kanal das Literal $\overline{x_i}$, so überlappt der Anfang des Kanals eine f-Disk andernfalls eine t-Disk von $S(x_i)$



$\overline{x_i}$

Kanäle

Repräsentiert ein Kanal das Literal $\overline{x_i}$, so überlappt der Anfang des Kanals eine f-Disk andernfalls eine t-Disk von $S(x_i)$



x_i

Kanäle

- Ein Kanal hat den Wert *TRUE*, falls die Kanal-Disks in Richtung Klausel-Gadget aufsteigend aufeinander liegen
- Ist $\min(S) \geq \frac{3}{4}$ so kann die Disk am Ende des Kanals von der Klausel-Disk genau dann überlappt werden, wenn der Kanal den Wert *TRUE* hat.

Beispiel

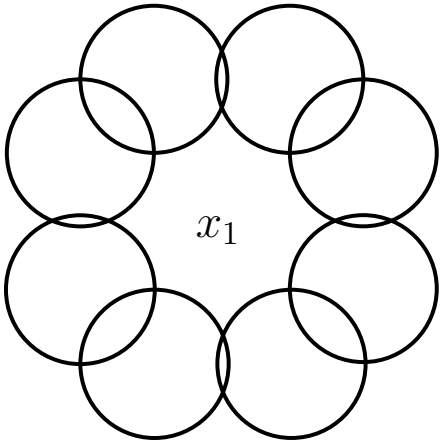
Eine erfüllende Belegung der p-3-SAT Instanz

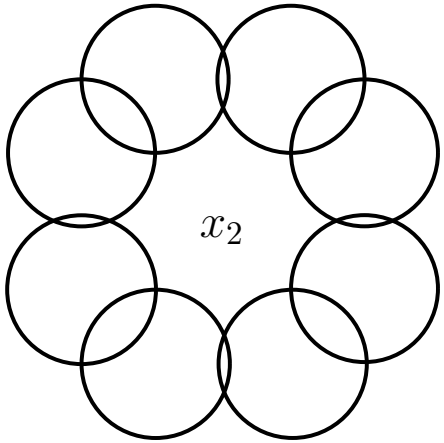
$$(x_1 \vee \overline{x_2} \vee \overline{x_3})$$

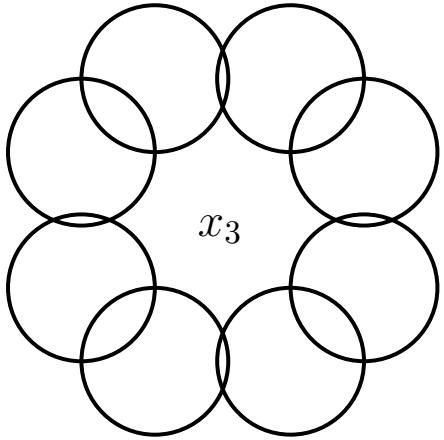
ist z.B.

$$x_1 = FALSE, x_2 = TRUE, x_3 = FALSE$$

Beispiel







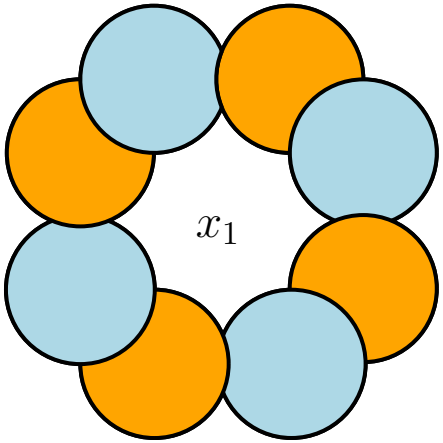
$x_1 \vee \overline{x_2} \vee \overline{x_3}$

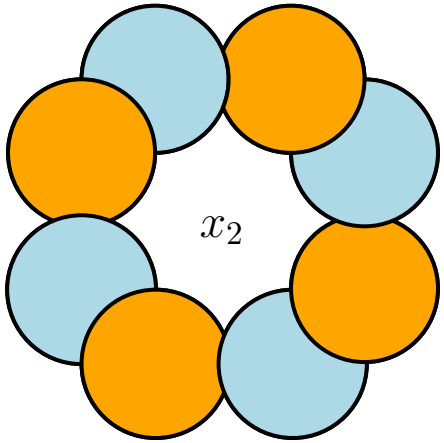
$x_1 = FALSE$

$x_2 = TRUE$

$x_3 = FALSE$

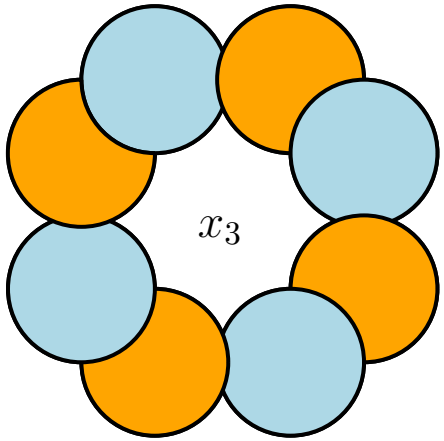
Beispiel





t-Disk

f-Disk



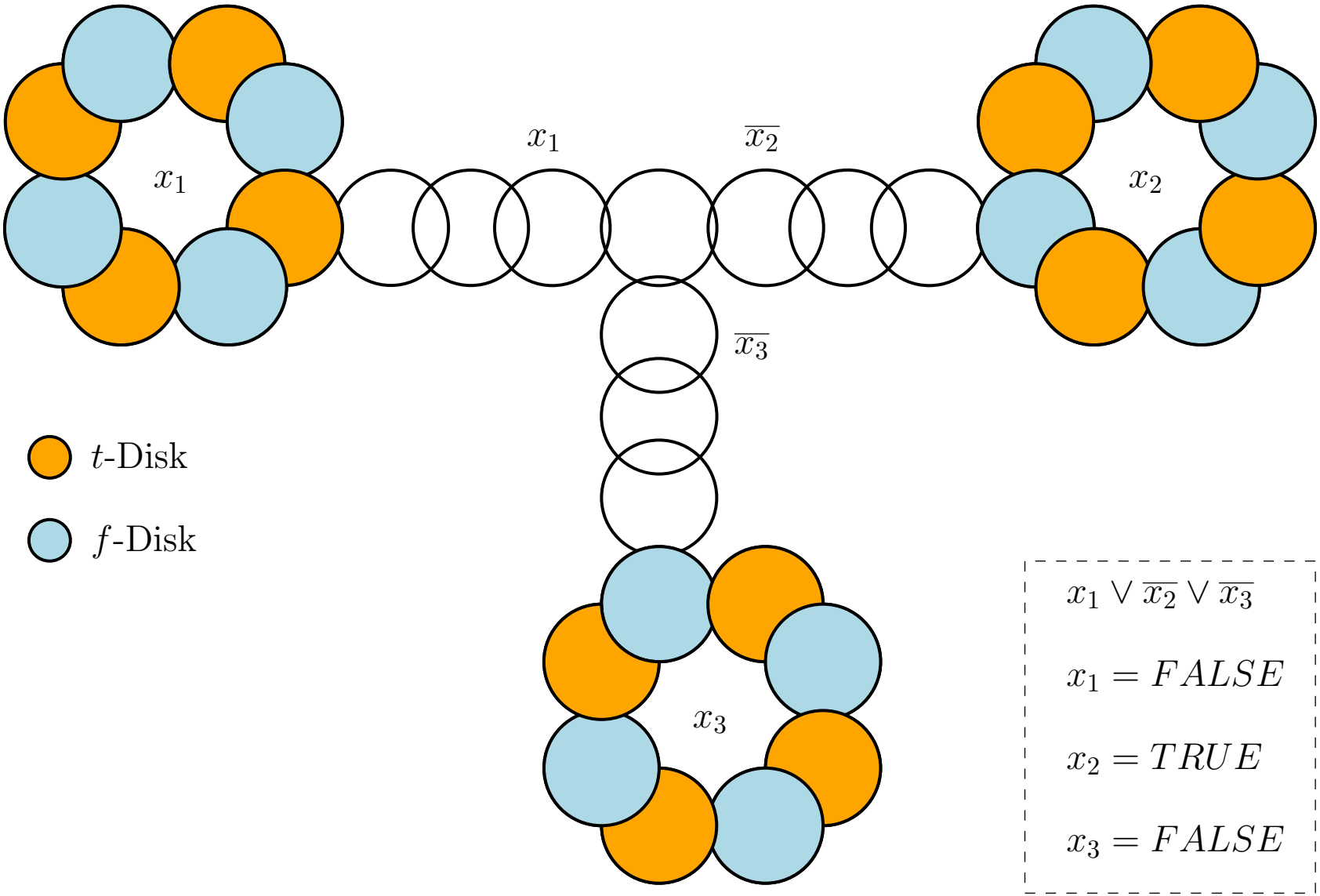
$x_1 \vee \overline{x_2} \vee \overline{x_3}$

$x_1 = FALSE$

$x_2 = TRUE$

$x_3 = FALSE$

Beispiel



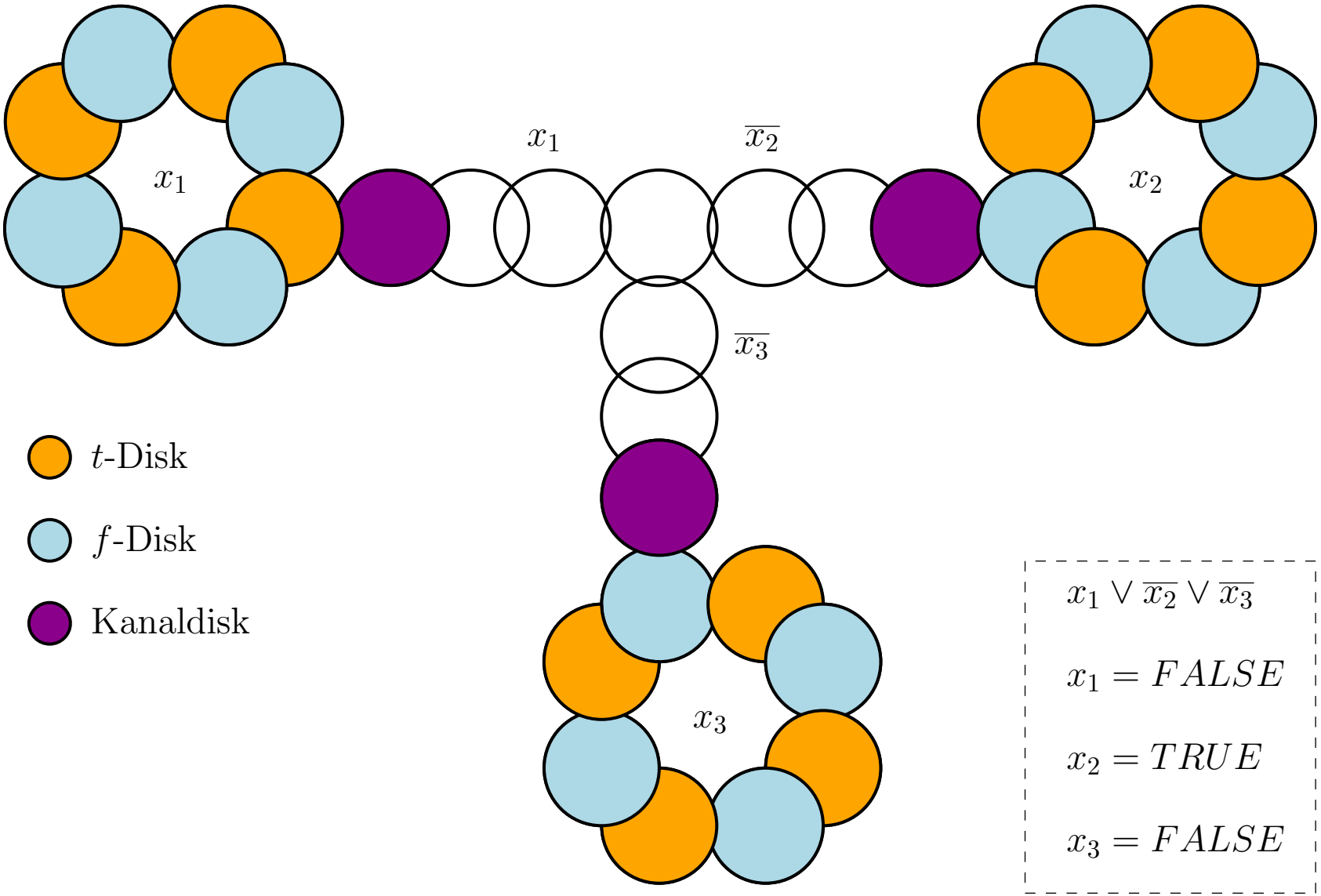
$x_1 \vee \overline{x_2} \vee \overline{x_3}$

$x_1 = FALSE$

$x_2 = TRUE$

$x_3 = FALSE$

Beispiel



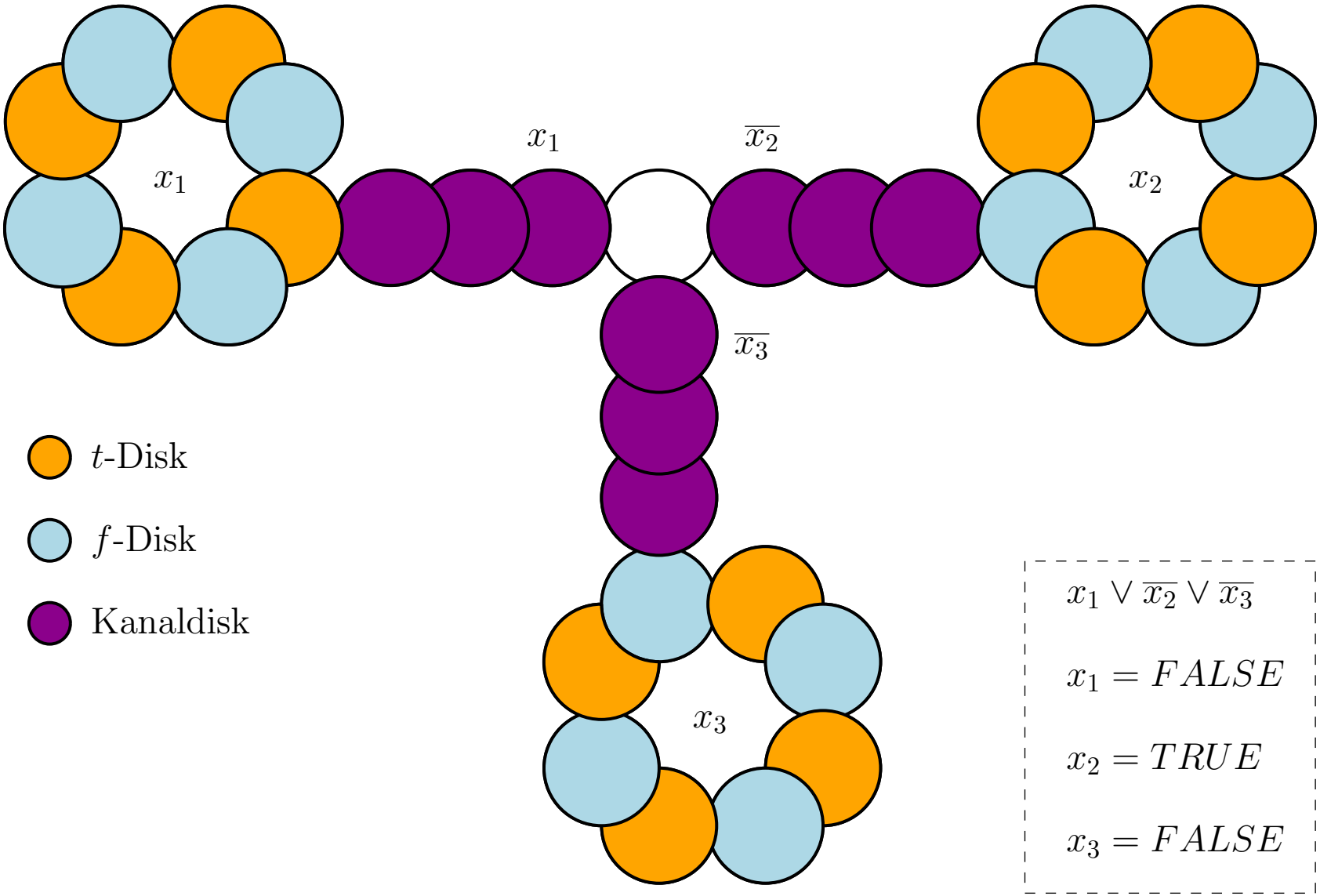
$x_1 \vee \overline{x_2} \vee \overline{x_3}$

$x_1 = FALSE$

$x_2 = TRUE$

$x_3 = FALSE$

Beispiel



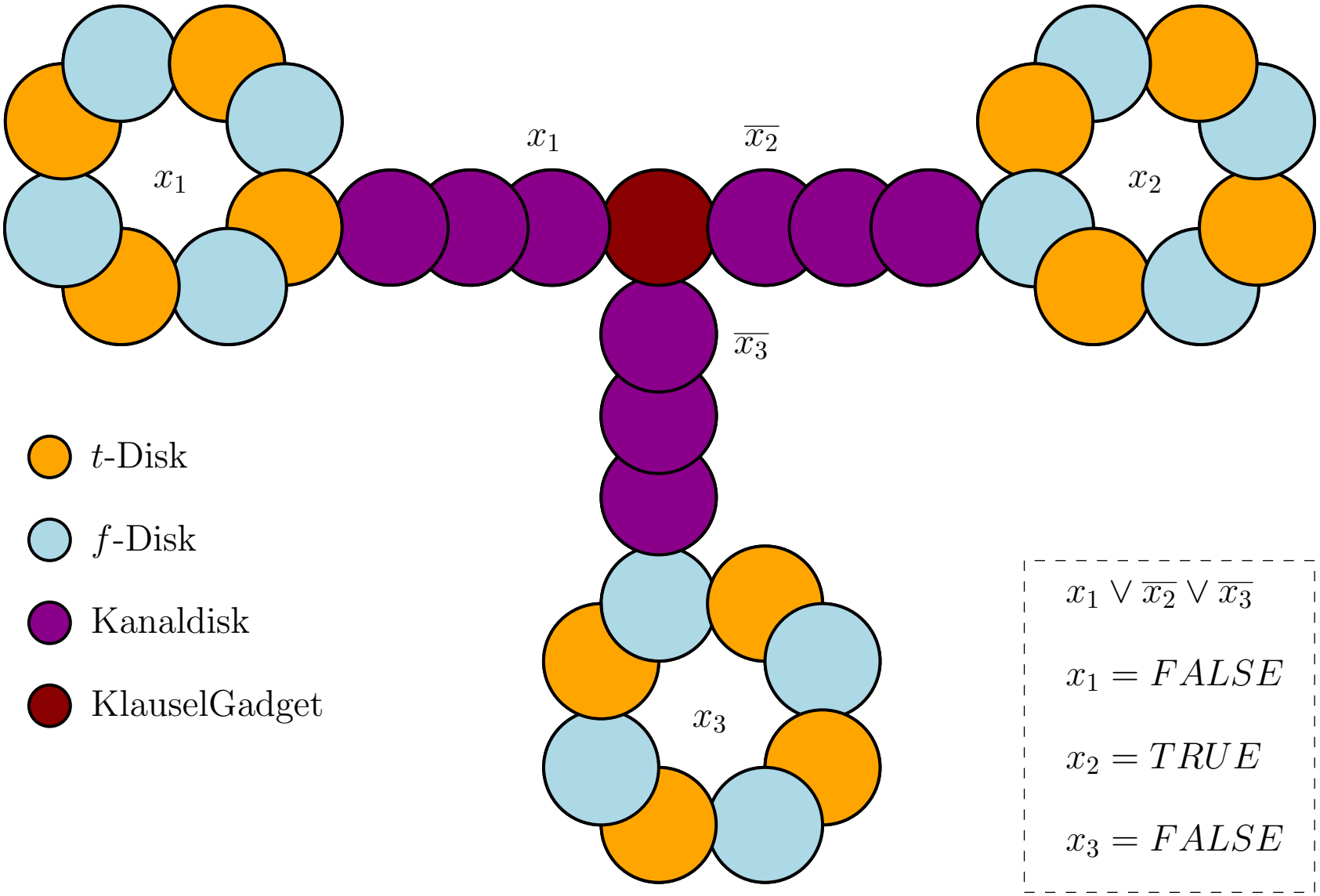
$x_1 \vee \overline{x_2} \vee \overline{x_3}$

$x_1 = FALSE$

$x_2 = TRUE$

$x_3 = FALSE$

Beispiel



$x_1 \vee \overline{x_2} \vee \overline{x_3}$

$x_1 = FALSE$

$x_2 = TRUE$

$x_3 = FALSE$

Resultat

Theorem

Es ist $\mathcal{NP}$-schwer zu entscheiden ob es für eine gegebene Menge von Disks S eine physikalisch-realisierbare Zeichnung mit $\min(S) \geq r$ gibt.

Beweis

Sei $\mathcal{I}$ eine Instanz von p-3-SAT und S die Menge von Disks zu $\mathcal{I}$ nach voriger Konstrukltion. Es gilt

$$\mathcal{I} \text{ ist erfüllbar} \Leftrightarrow \min(S) \geq \frac{3}{4}$$

$\Rightarrow$

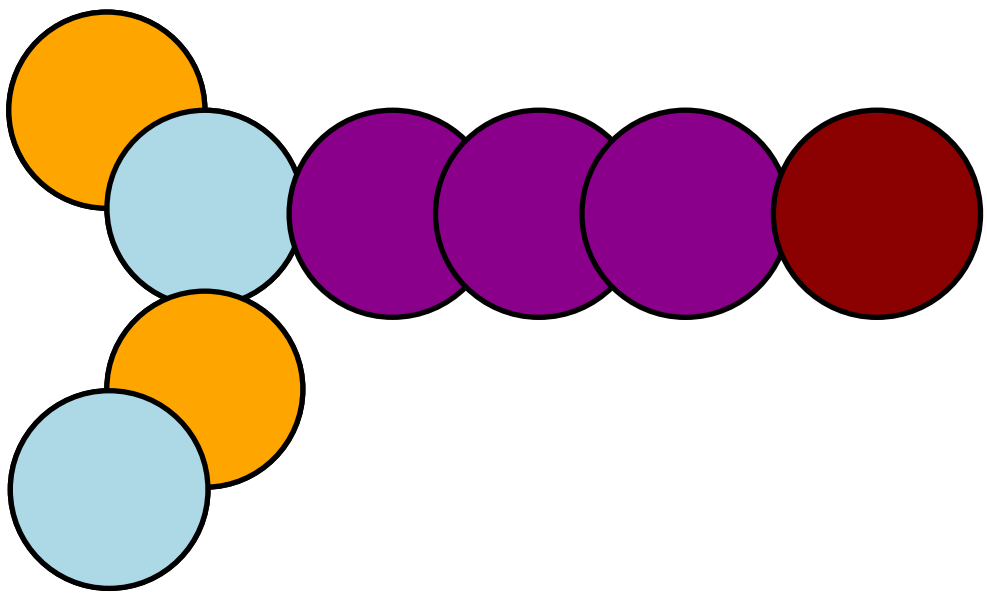
- Sei $\mathcal{I}$ erfüllbar.
- Fixiere eine erfüllende Belegung.
- Für jede Variable x_i der Belegung die *TRUE* ist zeichne $S(x_i)$ im *TRUE* Zustand, die Kanäle von x_i im *TRUE*-Zustand und die Kanäle von $\overline{x_i}$ im *FALSE*-Zustand
- Für jede Variable y_i der Belegung die *FALSE* ist zeichne umgekehrt

$\Rightarrow$

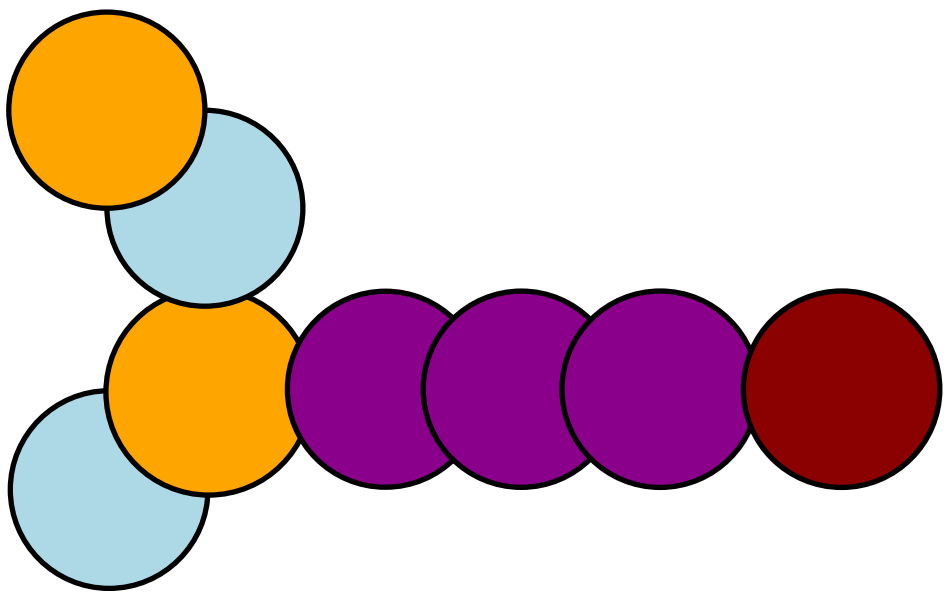
- Jede Klausel-Disk wird auf die Kanalenden im *TRUE* und unter die Kanalenden im *FALSE* Zustand gezeichnet
- Da jede Klausel in $\mathcal{I}$ *TRUE* ist gibt es für jede Klausel-Disk stets ein Kanalende im *TRUE* Zustand
- Daher sind $\frac{3}{4}$ des Randes jeder Klausel-Disk sichtbar
- Die Ränder der restlichen Disks sind nach Konstruktion zu mindestens $\frac{3}{4}$ sichtbar

- Angenommen es gibt eine physikalisch-realisierbare Zeichnung von S mit $\min(S) \geq \frac{3}{4}$
- Dann ist jedes Variablen-Gadget $S(x_i)$ entweder im Zustand *TRUE* oder *FALSE*
- Dies definiert eine boolsche Belegung der Variablen von $\mathcal{I}$

- Weiter muss jede Klausel-Disk auf einem Kanalende liegen
- Der zugehörige Kanal ist dann im Zustand *TRUE*, denn sonst wäre der Rand des Kanalanfangs nicht zu $\frac{3}{4}$ sichtbar



- Weiter muss jede Klausel-Disk auf einem Kanalende liegen
- Der zugehörige Kanal ist dann im Zustand *TRUE*, denn sonst wäre der Rand des Kanalanfangs nicht zu $\frac{3}{4}$ sichtbar



- Das durch den Kanal repräsentierte Literal erfüllt die Klausel
- Damit ist die Belegung eine erfüllende Belegung von $\mathcal{I}$

Anmerkungen

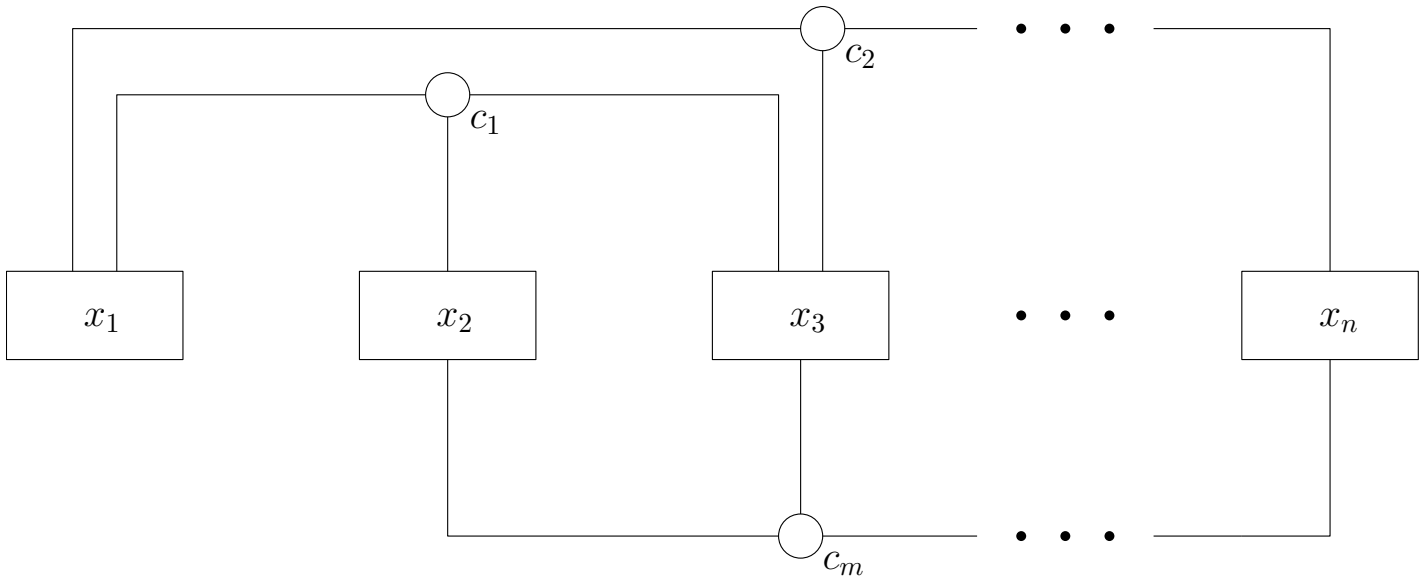
- Ist das Max-Min Problem in $\mathcal{NP}$?

Anmerkungen

- Ist das Max-Min Problem in $\mathcal{NP}$?
- Warum überhaupt planares 3-SAT?

Anmerkungen

- Ist das Max-Min Problem in $\mathcal{NP}$?
- Warum überhaupt planares 3-*SAT*?
- Ist die Konstruktion polynomiell?



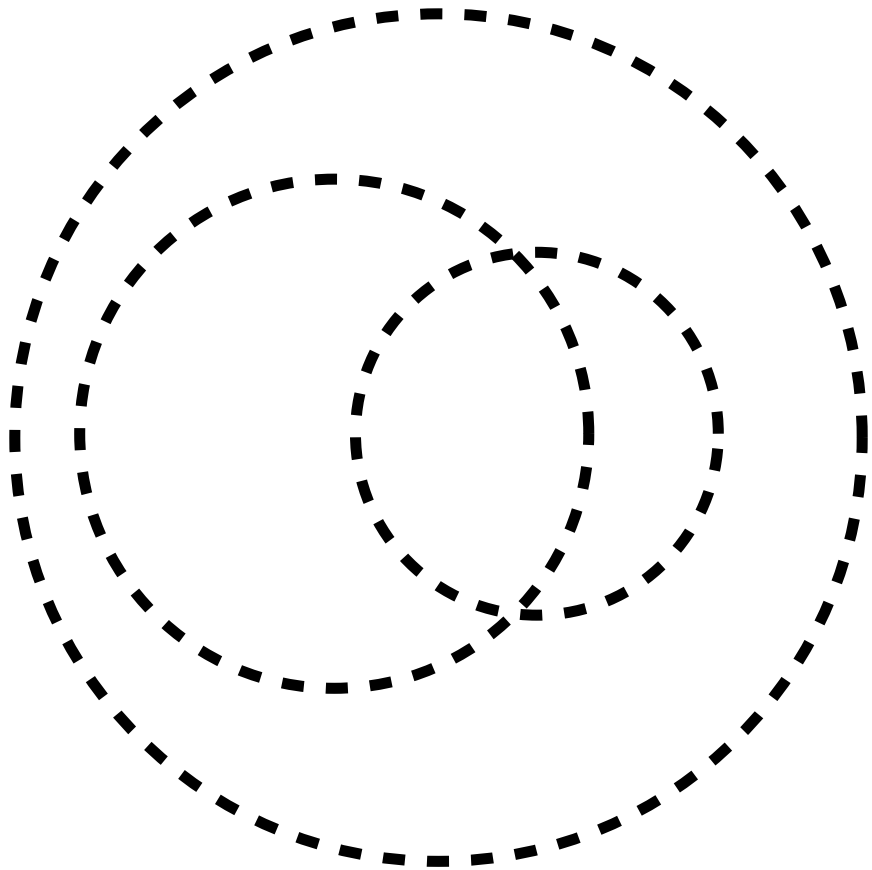
Überblick

	Max-Min	Max-Total
physikalisch realisierbar	$\mathcal{NP}$-schwer 	
Stacking		

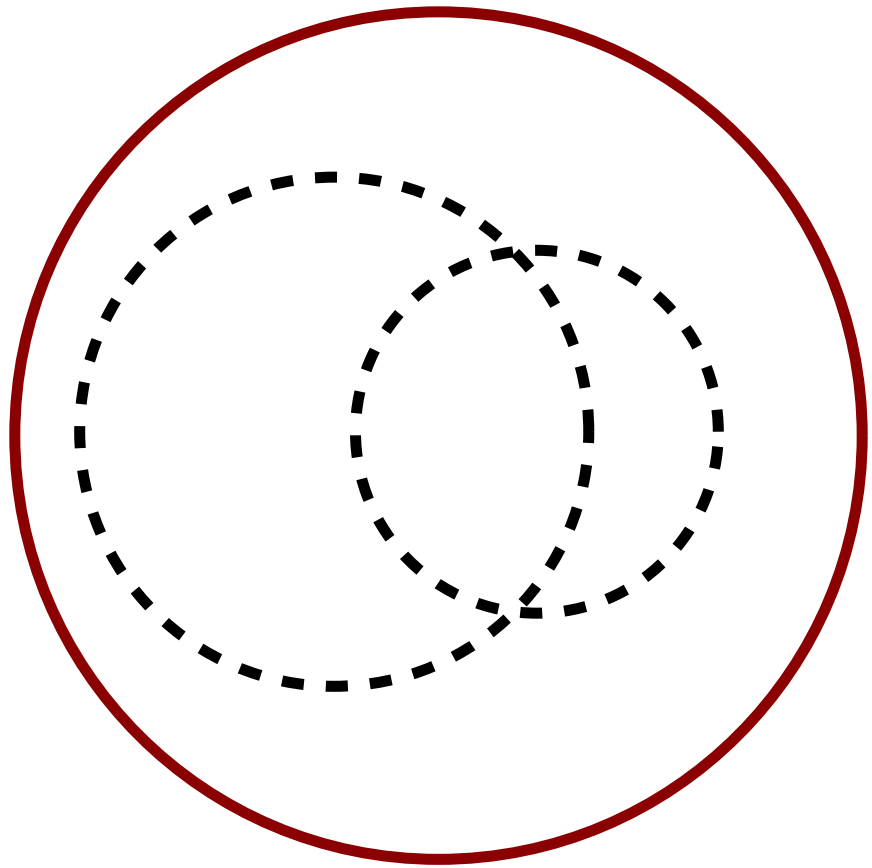
Überblick

	Max-Min	Max-Total
physikalisch realisierbar	$\mathcal{NP}$-schwer	
Stacking	in $\mathcal{P}$	

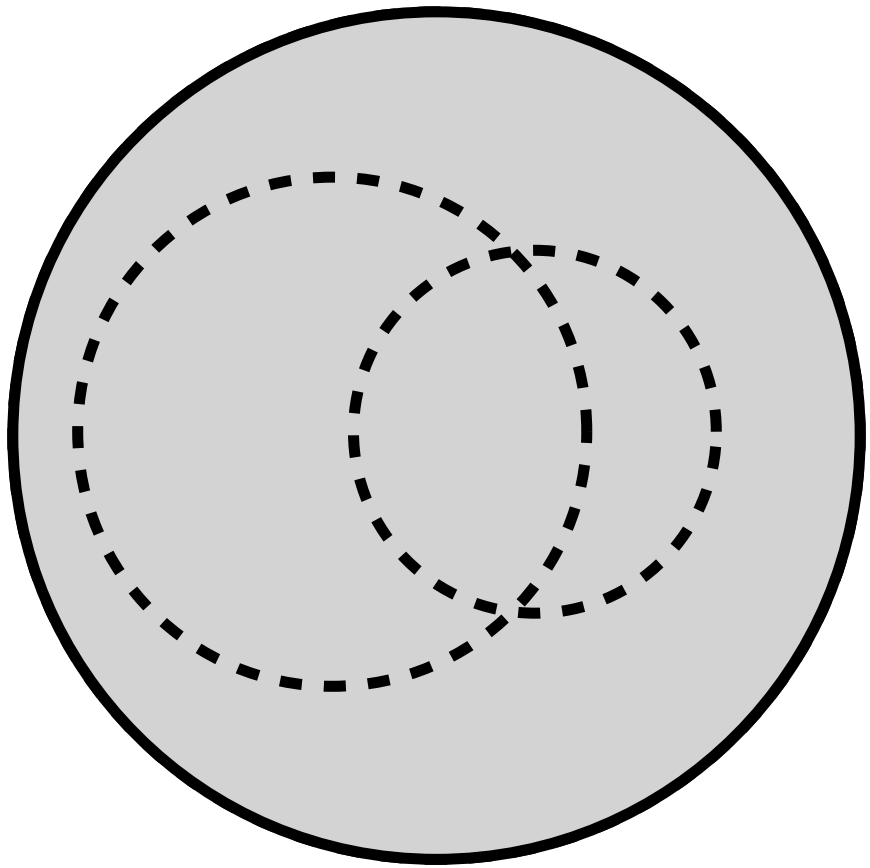
Ein Greedy-Algorithmus



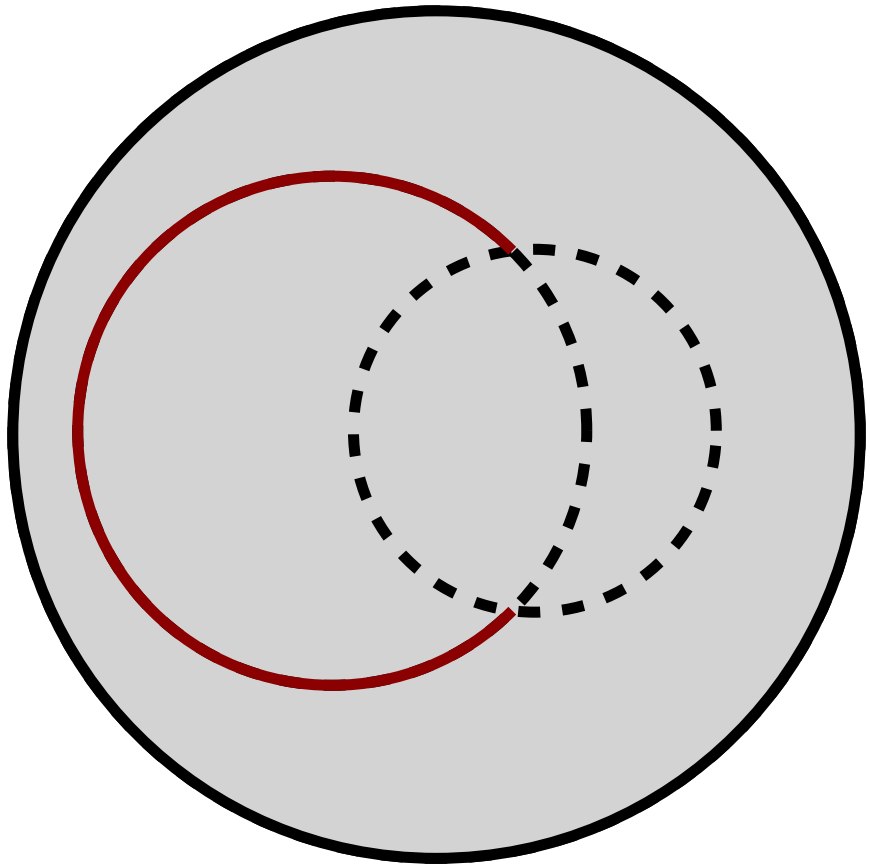
Ein Greedy-Algorithmus



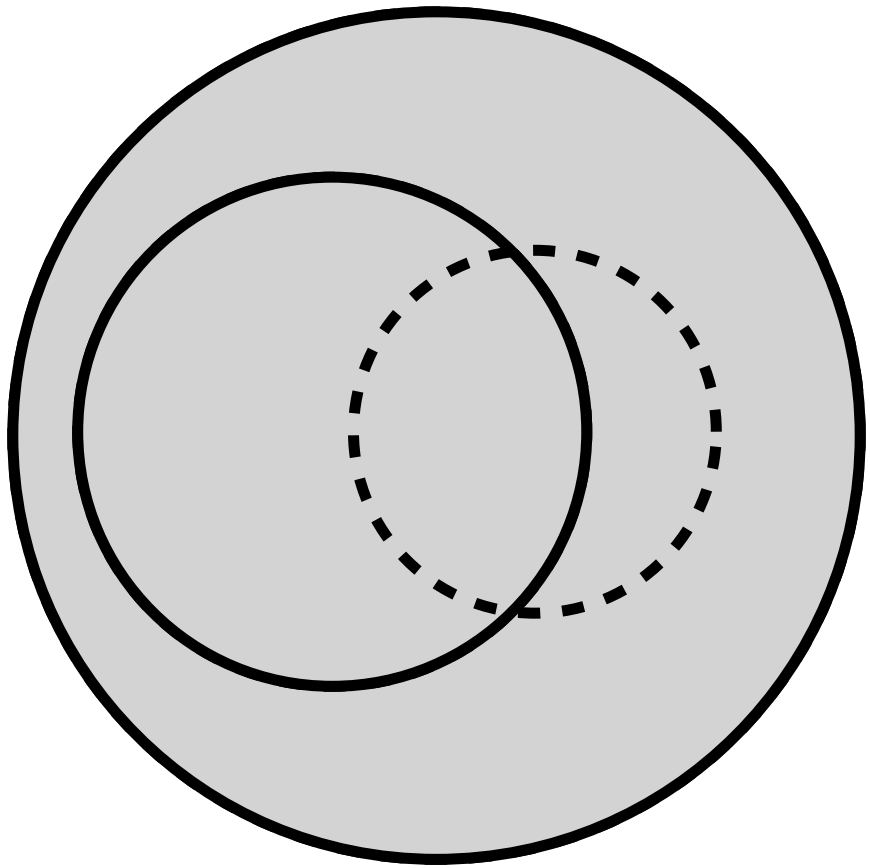
Ein Greedy-Algorithmus



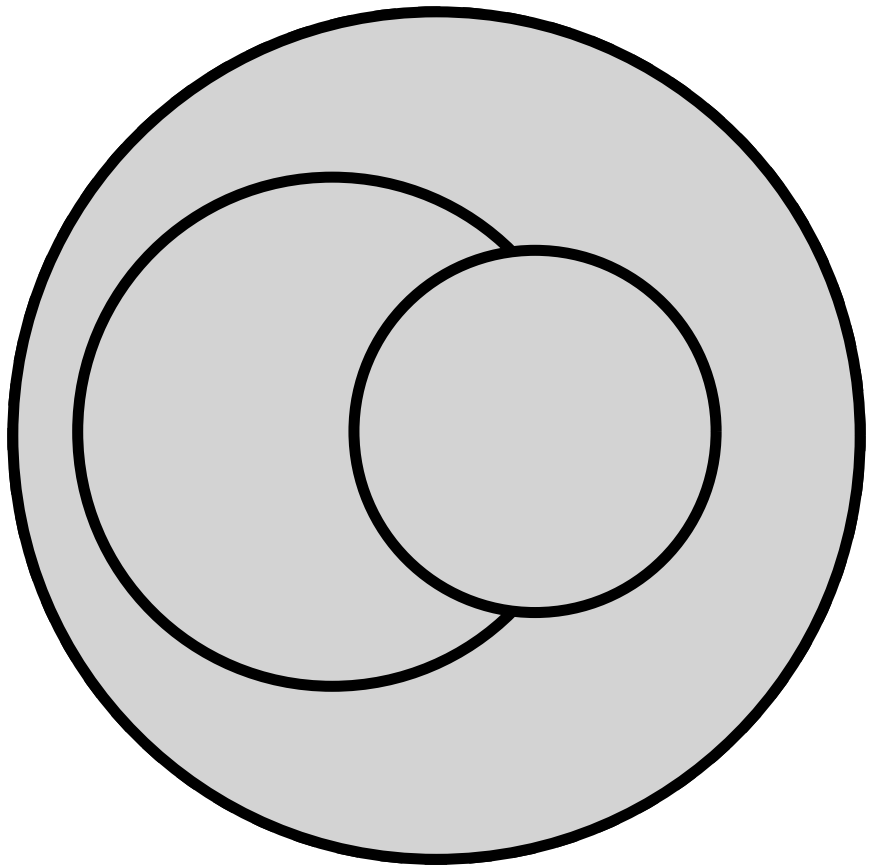
Ein Greedy-Algorithmus



Ein Greedy-Algorithmus



Ein Greedy-Algorithmus



Optimales Max-Min Stacking

Algorithm: `SucheUntersteDisk`(S)

Data: Set of Disks $S = \{D_1, \ldots, D_n\}$

for $i \leftarrow 1$ **to** n **do**

  Bestimme den sichtbaren Rand $vis(D_i)$ von D_i, wenn D_i die unterste Disk wäre

end

Sei $vis(D_k) = \max_{i=1,\ldots,n} vis(D_i)$

Zeichne D_k

`SucheUntersteDisk`$(S \setminus \{D_k\})$

Optimales Max-Min Stacking

- Dieser Algorithmus liefert ein optimales Stacking
- Sei A das Stacking des Algorithmus und $A[i]$ die i-te Disk der Stacking-Reihenfolge
- Sei O^0 ein optimales Stacking und $O^0[i]$ die i-te Disk dieses Stackings
- Vorgehen: Interative Konstruktion von Zwischenstackings $O^1, \ldots, O^n$ mit $O^n = A$ und

$$\min(O^k) \geq \min(O^{k-1})$$

Optimales Max-Min Stacking

A	O^0	O^2
D_1	D_1	
D_2	D_4	
D_3	D_3	
D_4	D_2	
D_5	D_5	

Optimales Max-Min Stacking

A	O^0	O^2
D_1	D_1	
D_2	D_4	
D_3	D_3	
D_4	D_2	
D_5	D_5	

Optimales Max-Min Stacking

A	O^0	O^2
D_1	D_1	
D_2	D_4	
D_3	D_3	
D_4	D_2	
D_5	D_5	

Optimales Max-Min Stacking

A	O^0	O^2
D_1	D_1	
D_2	D_4	
D_3	D_3	
D_4	D_2	
D_5	D_5	

Optimales Max-Min Stacking

A	O^0	O^2
D_1	D_1	D_1
D_2	D_4	D_2
D_3	D_3	D_4
D_4	D_2	D_3
D_5	D_5	D_5

Optimales Max-Min Stacking

A	O^0	O^2
D_1	D_1	D_1
D_2	D_4	D_2
D_3	D_3	D_4
D_4	D_2	D_3
D_5	D_5	D_5

Optimales Max-Min Stacking

A	O^0	O^2
D_1	D_1	D_1
D_2	D_4	D_2
D_3	D_3	D_4
D_4	D_2	D_3
D_5	D_5	D_5

Optimales Max-Min Stacking

A	O^0	O^2
D_1	D_1	D_1
D_2	D_4	D_2
D_3	D_3	D_4
D_4	D_2	D_3
D_5	D_5	D_5

Laufzeit

- Die zugrundeliegende Datenstruktur sind die sogenannten Segment-Trees
- Die Laufzeit wird $\mathcal{O}(n^2 \log n)$ sein
- Also ist das Max-Min Problem für Stackings in $\mathcal{P}$

Segment-Trees

- Gegeben eine Menge von Intervallen (Eingabeintervallen)

$$I = \{I_1, I_2, \ldots, I_n\}$$

  mit $I_i = [x_i, x_i'] \subseteq \mathbb{R}$

- Frage: In welchen Intervallen liegt ein Punkt $q \in \mathbb{R}$

Segment-Trees

Die End- und Schnittpunkte der Eingabeintervalle

$$I = \{I_1, I_2, \ldots, I_n\}$$

induzieren eine Zerlegung von I in elementare Intervalle

$$I = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$$

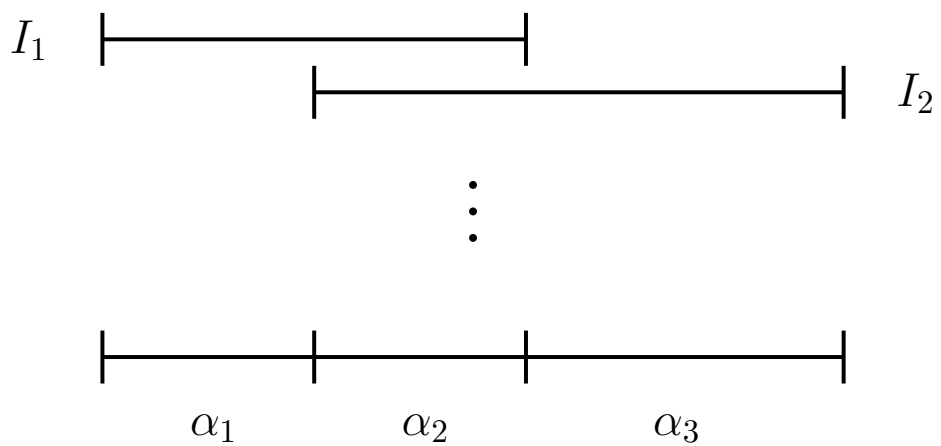
Segment-Trees

Die End- und Schnittpunkte der Eingabeintervalle

$$I = \{I_1, I_2, \ldots, I_n\}$$

induzieren eine Zerlegung von I in elementare Intervalle

$$I = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$$

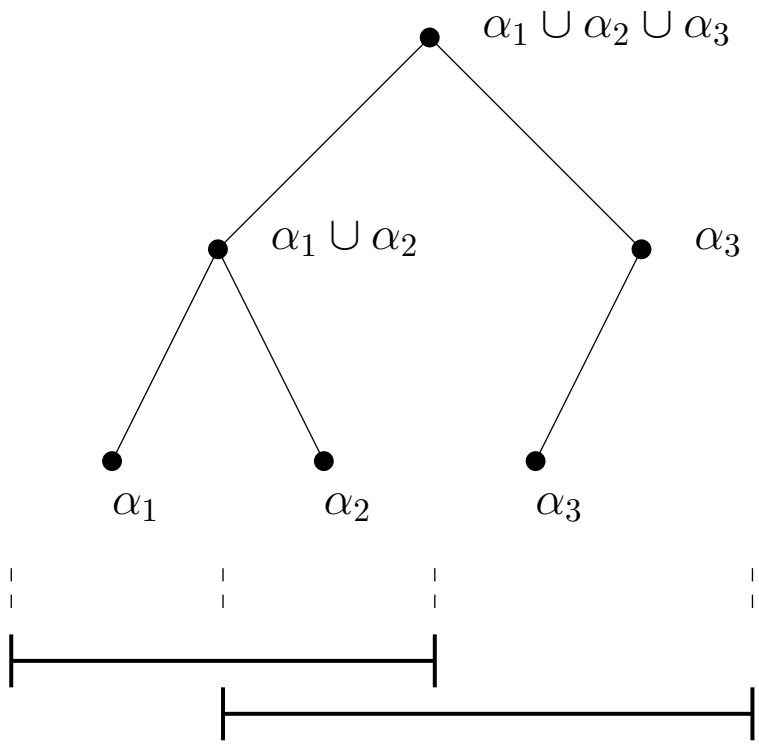


Knotenintervalle

Ein Segment-Tree zu I ist ein binärer Baum an dessen Blättern μ_i in $Int(\mu_i)$ die Intervalle α_i und in dessen inneren Knoten v in $Int(v)$ die Intervalle $Int(lc(v)) \cup Int(rc(v))$ gespeichert werden. Die Intervalle $Int(\cdot)$ heißen Knotenintervalle.

Knotenintervalle

Ein Segment-Tree zu I ist ein binärer Baum an dessen Blättern μ_i in $Int(\mu_i)$ die Intervalle α_i und in dessen inneren Knoten v in $Int(v)$ die Intervalle $Int(lc(v)) \cup Int(rc(v))$ gespeichert werden. Die Intervalle $Int(\cdot)$ heißen Knotenintervalle.

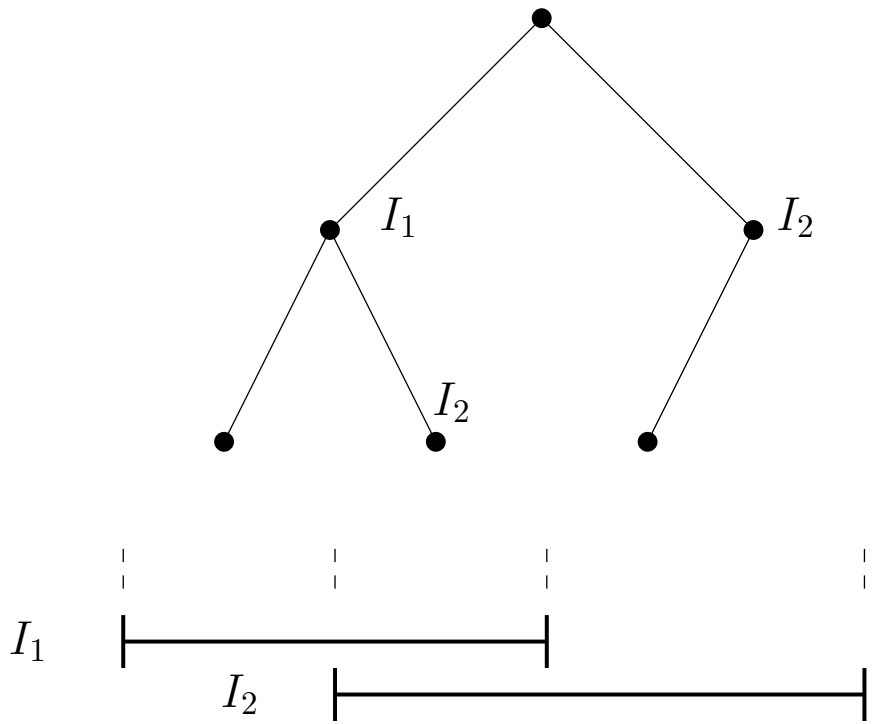


Eingabeintervalle

Jeder Knoten v speichert zusätzlich eine Menge von Eingabeintervallen $I(v) \subseteq I$. Es gilt $I_i \in I(v)$ genau dann, wenn $Int(v) \subseteq I_i$ und $Int(parent(v)) \not\subseteq I_i$

Eingabeintervalle

Jeder Knoten v speichert zusätzlich eine Menge von Eingabeintervallen $I(v) \subseteq I$. Es gilt $I_i \in I(v)$ genau dann, wenn $Int(v) \subseteq I_i$ und $Int(parent(v)) \not\subseteq I_i$



Segment-Trees

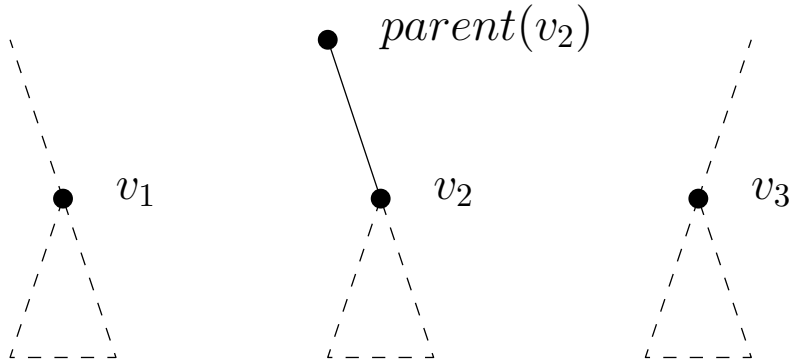
Lemma

Ein Eingabeintervall wird pro Baumlevel höchstens in zwei Knoten gespeichert.

Segment-Trees

Lemma

Ein Eingabeintervall wird pro Baumlevel höchstens in zwei Knoten gespeichert.

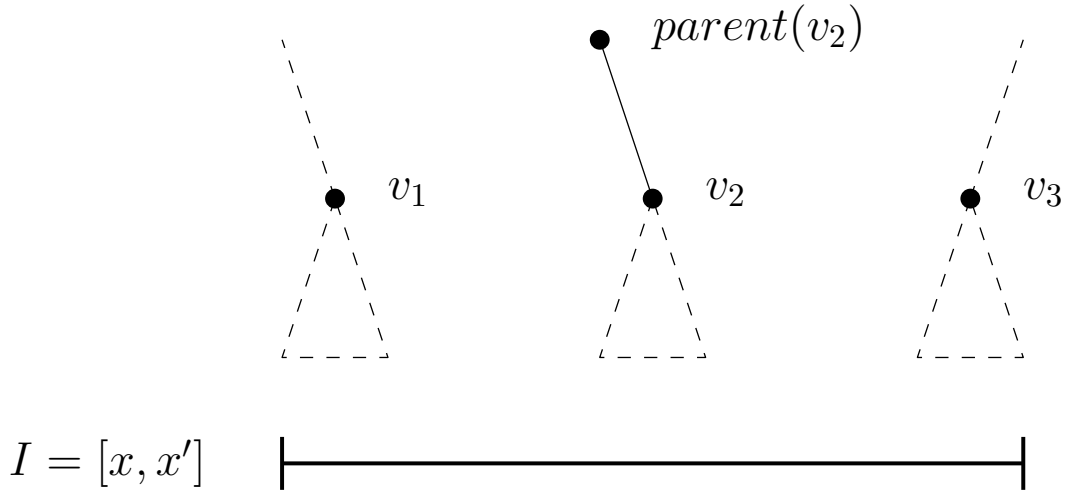


$I = [x, x']$

Segment-Trees

Lemma

Ein Eingabeintervall wird pro Baumlevel höchstens in zwei Knoten gespeichert.



Segment-Trees

Algorithm: `Insert`$(v, \mathtt{I})$

if $Int(v) \subseteq I$ **then**
  Speichere I in v

else
  if $Int(lc(v)) \cap I \neq \emptyset$ **then**
    `Insert`$(lc(v), I)$
  end
  if $Int(rc(v)) \cap I \neq \emptyset$ **then**
    `Insert`$(rc(v), I)$
  end

end

Segment-Trees

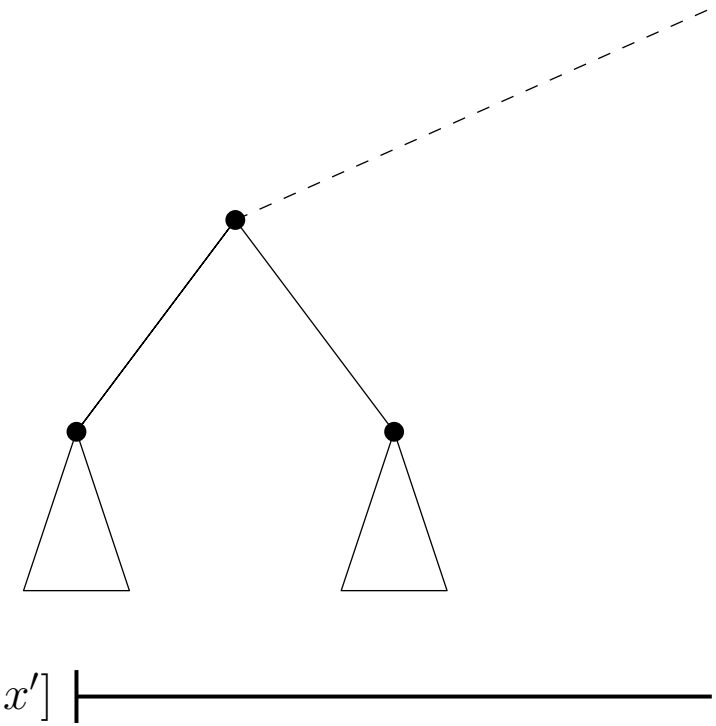
Lemma

Pro Baumlevel werden beim Einfügen eines Eingabeknotens höchstens vier Knoten besucht

Segment-Trees

Lemma

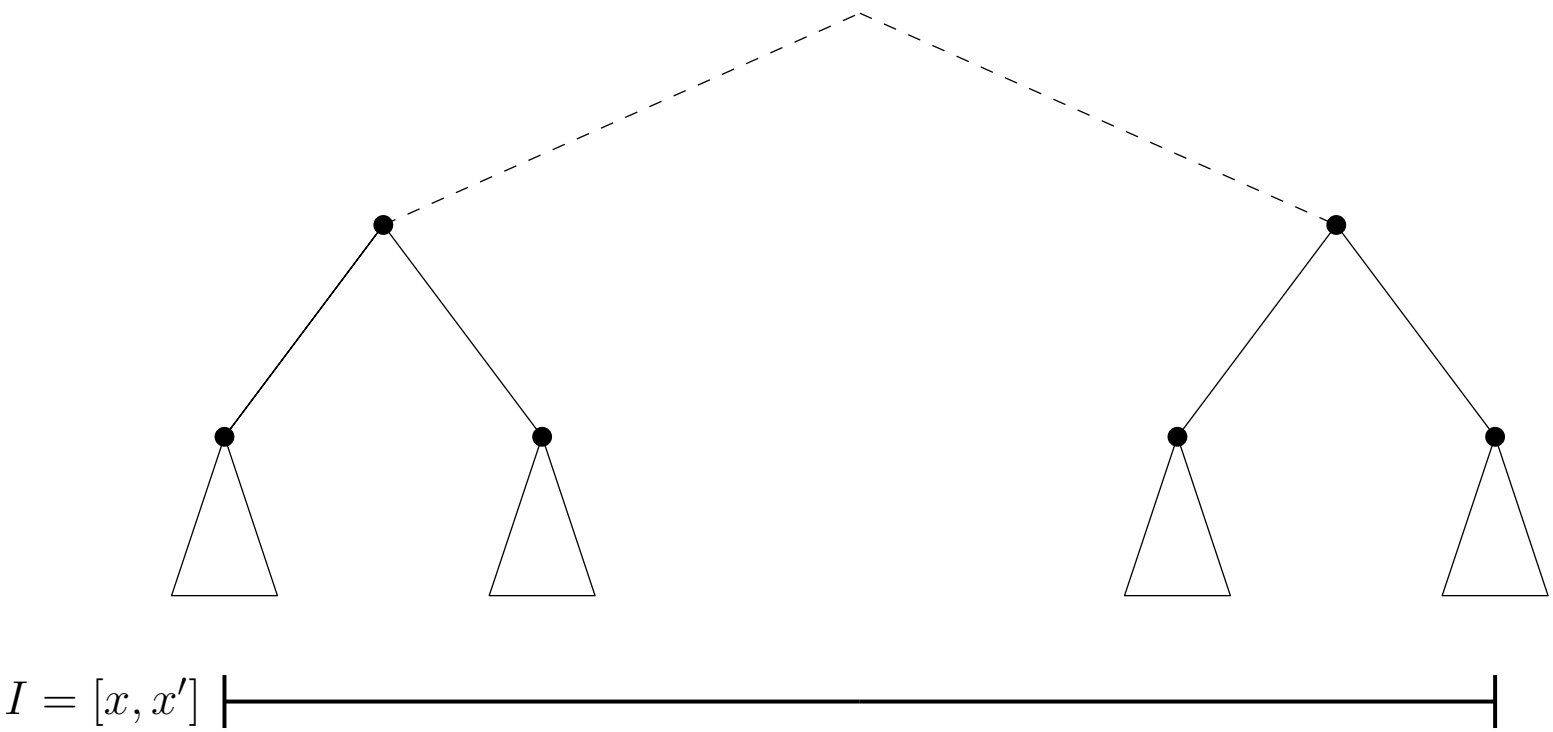
Pro Baumlevel werden beim Einfügen eines Eingabeknotens höchstens vier Knoten besucht



Segment-Trees

Lemma

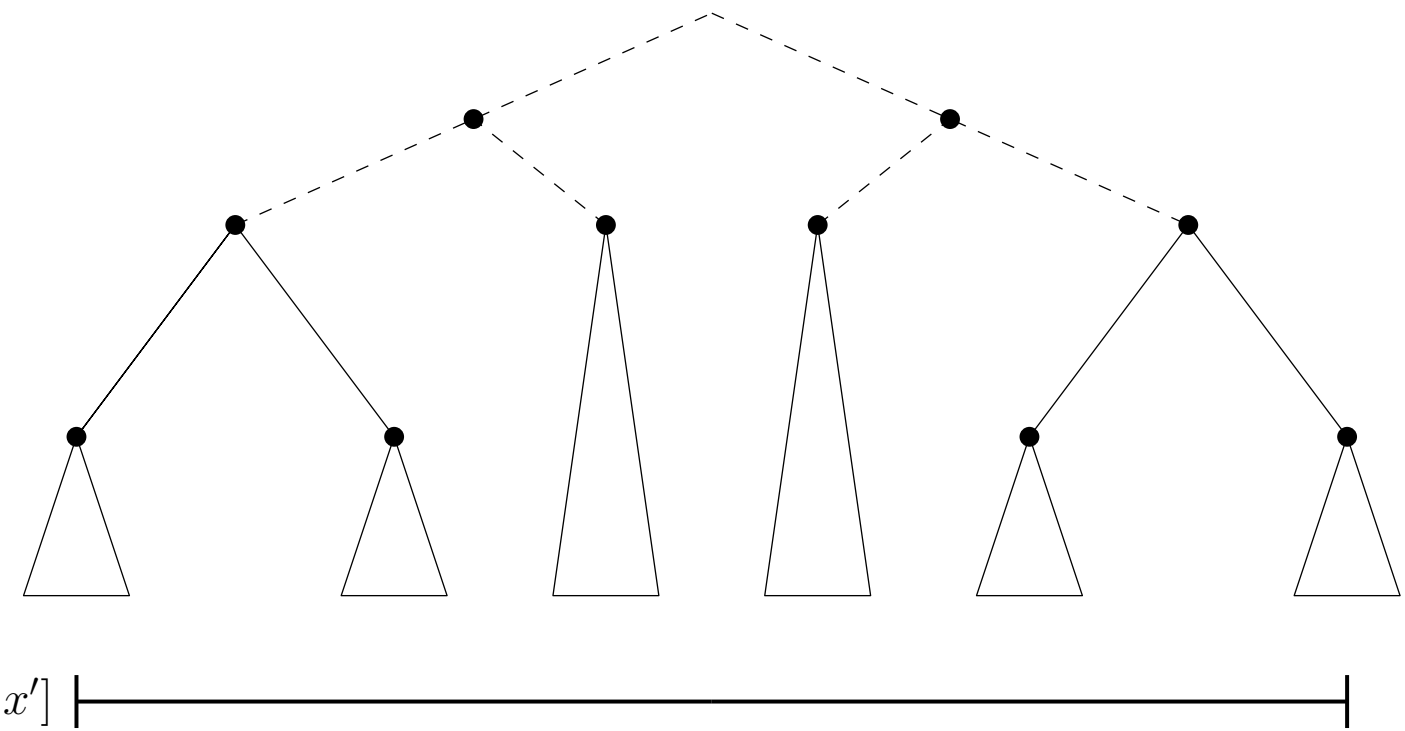
Pro Baumlevel werden beim Einfügen eines Eingabeknotens höchstens vier Knoten besucht



Segment-Trees

Lemma

Pro Baumlevel werden beim Einfügen eines Eingabeknotens höchstens vier Knoten besucht

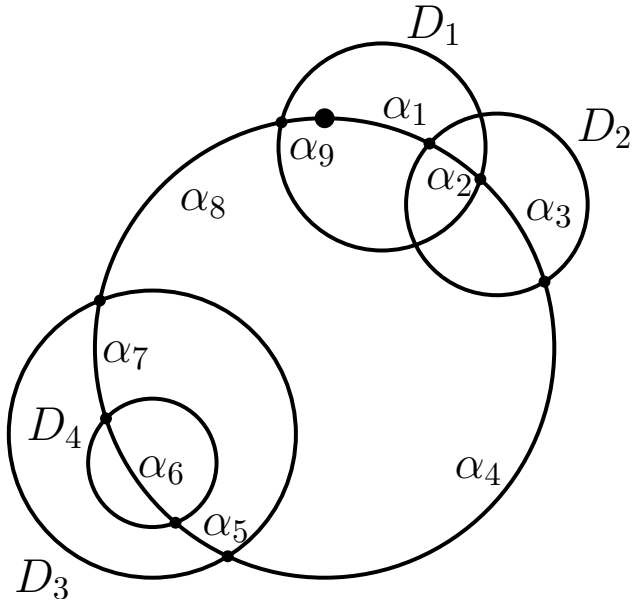


Segment-Trees

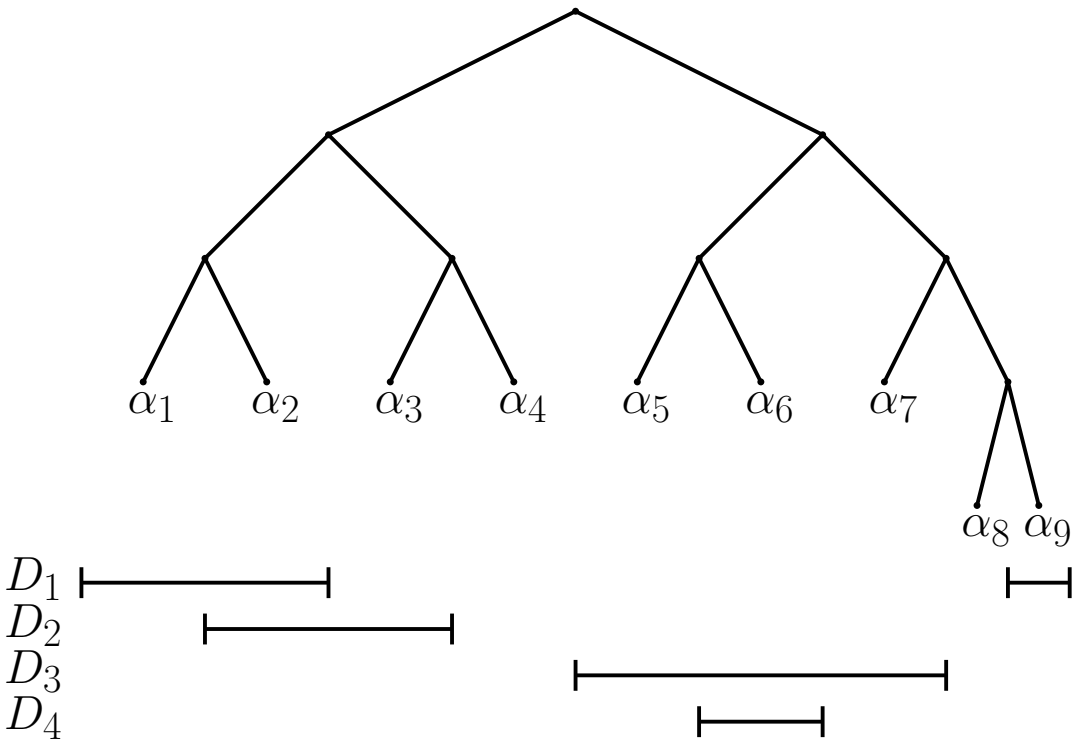
- Das Einfügen und Suchen eines Eingabeknotens in einem Segment-Tree ist in $\mathcal{O}(\log n)$
- Die Konstruktion eines Segment-Trees aus n Eingabeintervallen ist in $\mathcal{O}(n \log n)$

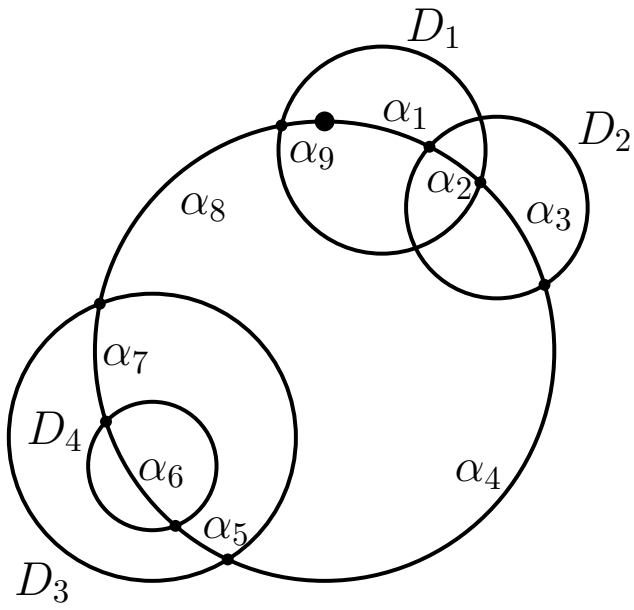
Implementierung

- Rand einer Disk D_j als Intervall vom obersten Punkt im Uhrzeigersinn
- Schnittpunkte mit anderen Disks definieren die Elementarintervalle
- Jede andere Disk D_i überdeckt kein, ein oder zwei Intervalle von D_j



Implementierung





Implementierung

- Für jede Disk D_i ein Segment-Tree mit Eingabeintervallen D_j $(i \neq j)$
- Speichere an jedem Knoten v von D_i die Werte $Int(v)$, $I(v)$ und $vis(v)$

Implementierung

- Für jede Disk D_i ein Segment-Tree mit Eingabeintervallen D_j $(i \neq j)$
- Speichere an jedem Knoten v von D_i die Werte $Int(v)$, $I(v)$ und $vis(v)$
- $vis(v)$ ist definert durch

$$vis(v) = \begin{cases} 0, & \text{falls } |I(v)| \geq 1 \\ vis(lc(v)) + vis(rc(v)), & \text{falls } v \text{ innerer Knoten} \\ |Int(v)|, & \text{falls } v \text{ Blatt} \end{cases}$$

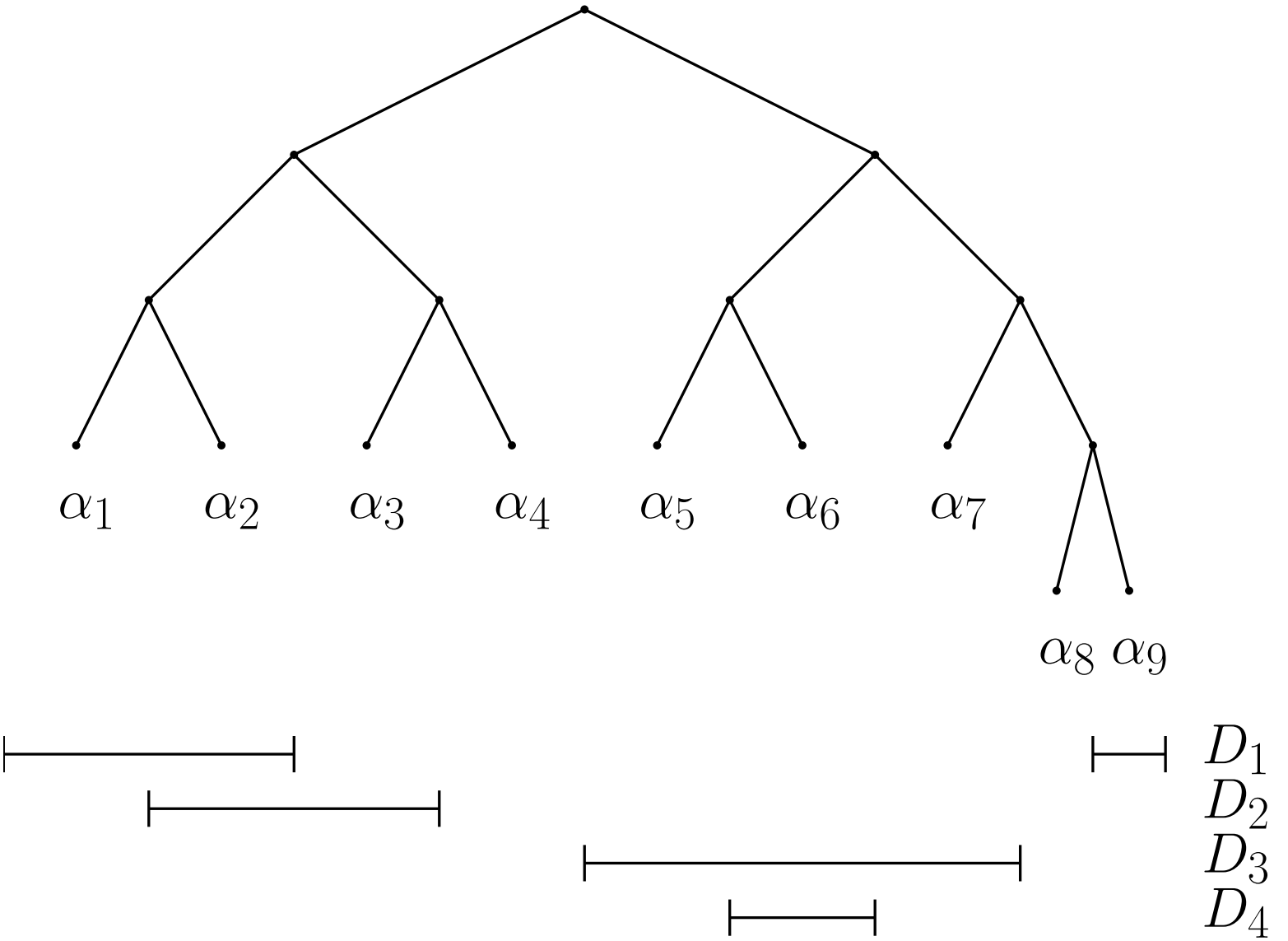
Implementierung

- Für jede Disk D_i ein Segment-Tree mit Eingabeintervallen D_j $(i \neq j)$
- Speichere an jedem Knoten v von D_i die Werte $Int(v)$, $I(v)$ und $vis(v)$
- $vis(v)$ ist definert durch

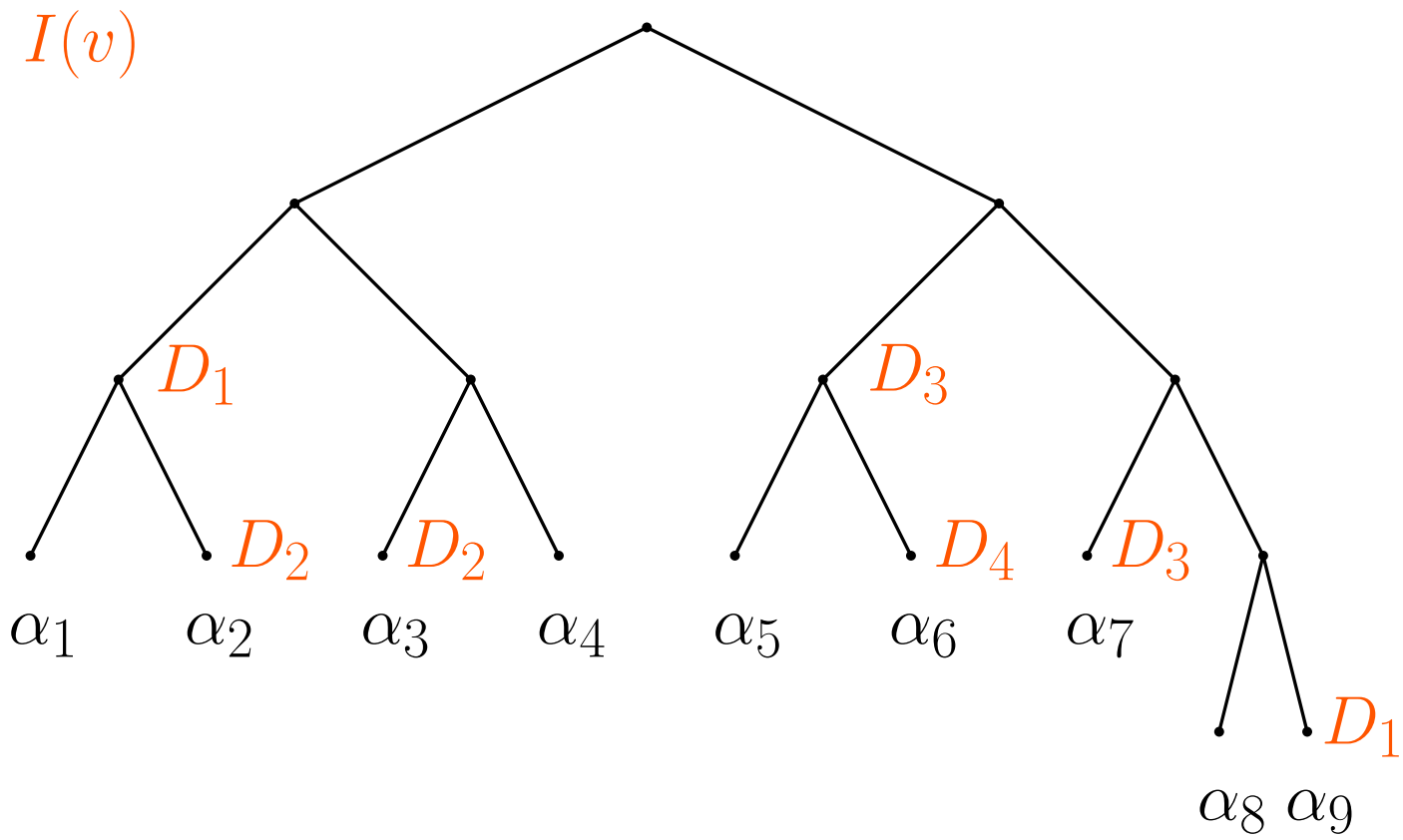
$$vis(v) = \begin{cases} 0, & \text{falls } |I(v)| \geq 1 \\ vis(lc(v)) + vis(rc(v)), & \text{falls } v \text{ innerer Knoten} \\ |Int(v)|, & \text{falls } v \text{ Blatt} \end{cases}$$

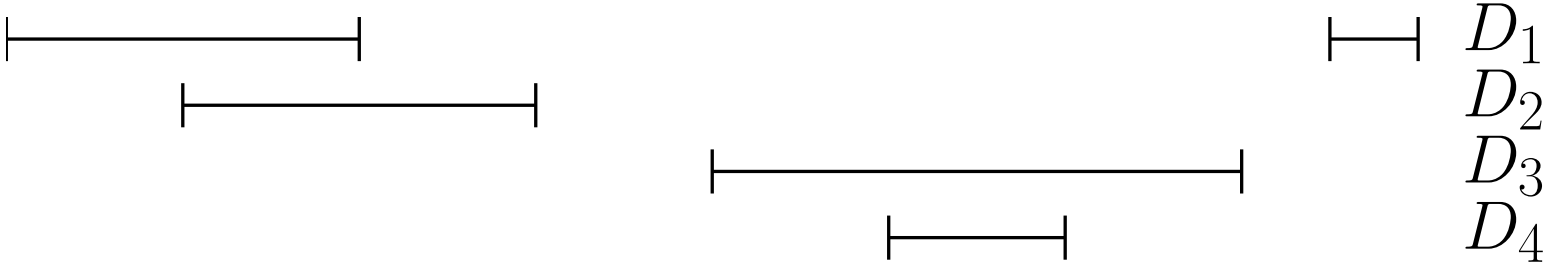
- $vis(root(T_i)$ ist der sichtbare Rand von D_i falls alle Disks D_j auf D_i gezeichnet werden.

Implementierung

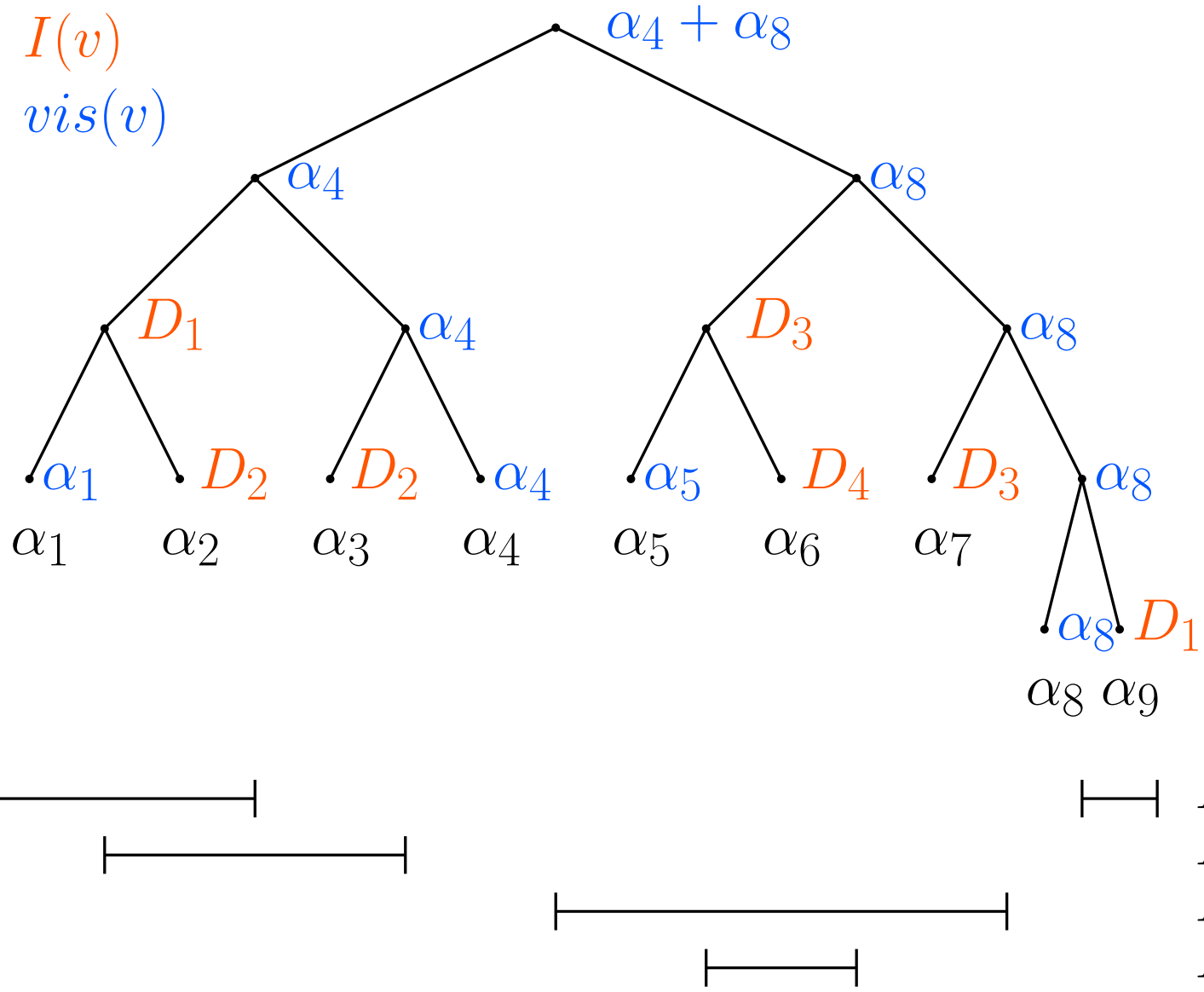


Implementierung





Implementierung



Implementierung

- Ein Schritt des Algorithmus:
- Suche D_i so, dass

$$vis(root(T_i)) = max_{j=1,\ldots,n} vis(root(T_j))$$

Implementierung

- Ein Schritt des Algorithmus:
- Suche D_i so, dass

$$vis(root(T_i)) = max_{j=1,\ldots,n} vis(root(T_j))$$

- Lösche T_i und zeichne D_i

Implementierung

- Ein Schritt des Algorithmus:
- Suche D_i so, dass

$$vis(root(T_i)) = max_{j=1,\ldots,n} vis(root(T_j))$$

- Lösche T_i und zeichne D_i
- Für $i \neq j$: Lösche das Eingabeintervall D_i aus allen Baumknoten $v \in T_j$ und aktualisiere $vis(\mu)$ für alle Knoten von v bis zur Wurzel

Implementierung

- Ein Schritt des Algorithmus:
- Suche D_i so, dass

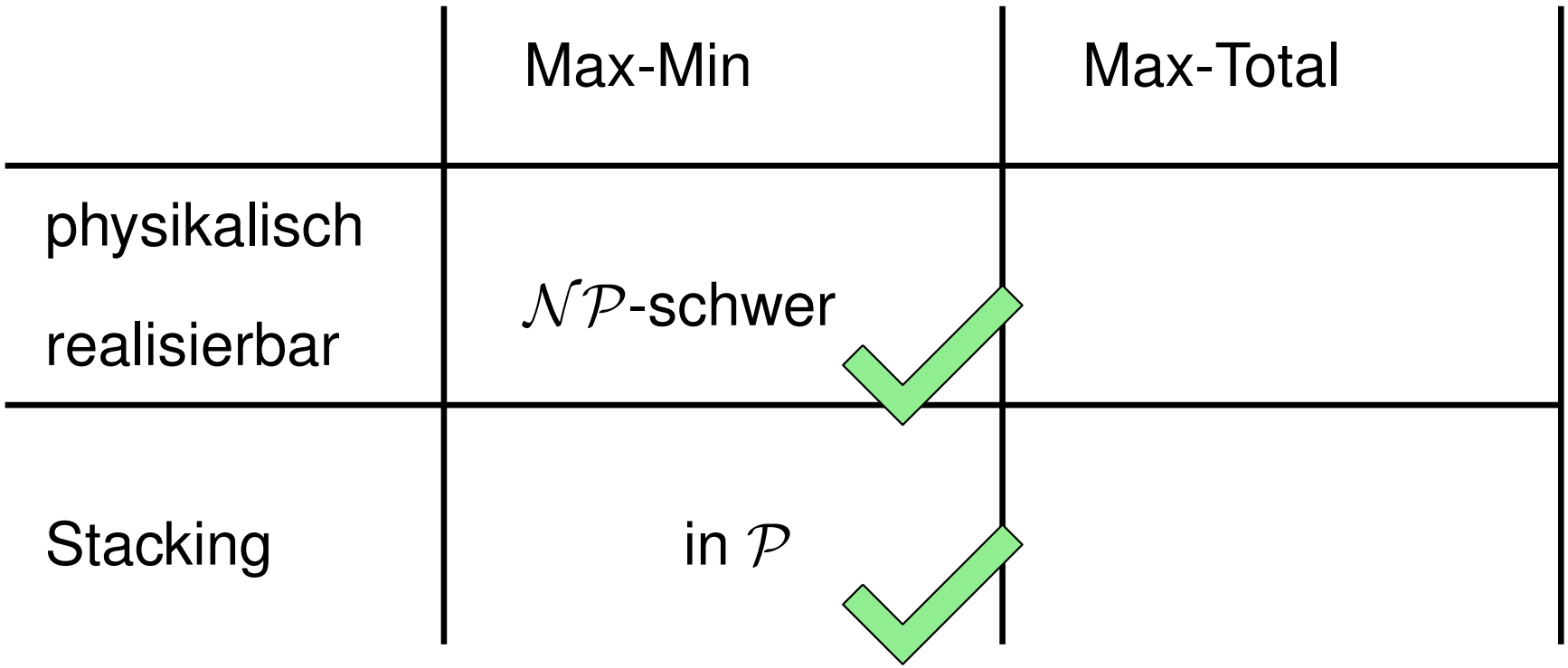
$$vis(root(T_i)) = max_{j=1,\ldots,n} vis(root(T_j))$$

- Lösche T_i und zeichne D_i
- Für $i \neq j$: Lösche das Eingabeintervall D_i aus allen Baumknoten $v \in T_j$ und aktualisiere $vis(\mu)$ für alle Knoten von v bis zur Wurzel
- Ein Schritt des Algorithmus ist damit in $\mathcal{O}(n \log n)$

Laufzeit

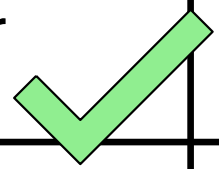
- Das Erstellen der n Segment-Trees ist in $\mathcal{O}(n^2 \log n)$
- Die Gesamtlaufzeit ist also in $\mathcal{O}(n^2 \log n)$

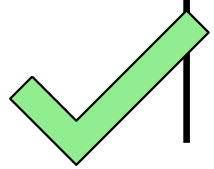
Überblick


Max-Min
Max-Total
physikalisch
realisierbar
$\mathcal{NP}$-schwer
Stacking
in $\mathcal{P}$

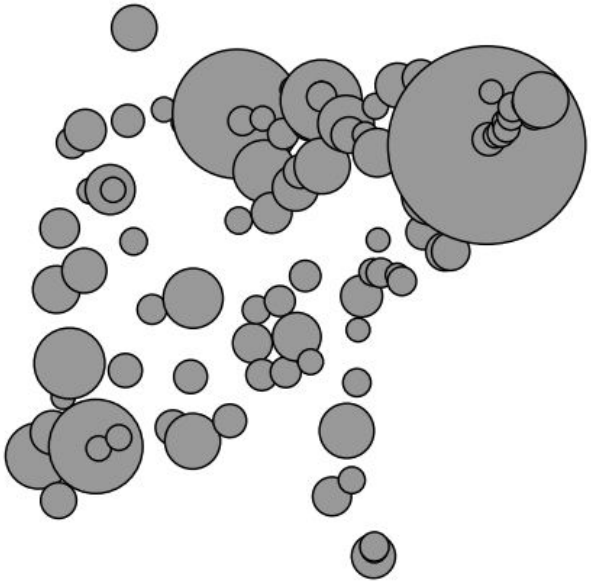
Überblick

	Max-Min	Max-Total
physikalisch realisierbar	$\mathcal{NP}$-schwer	$\mathcal{NP}$-schwer
Stacking	in $\mathcal{P}$	?

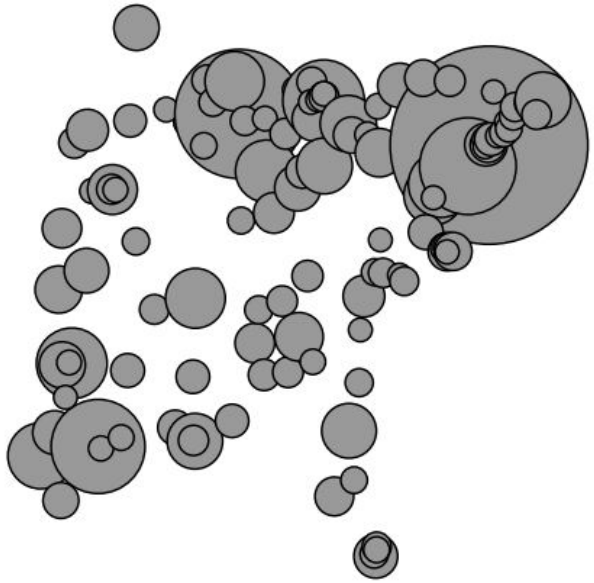




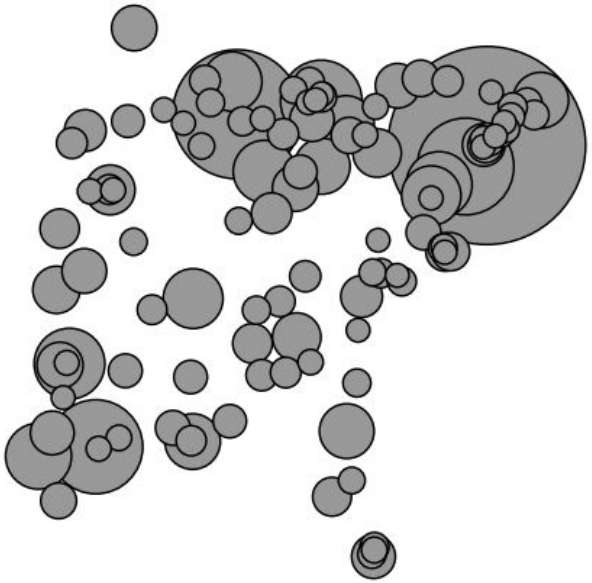
Evaluation



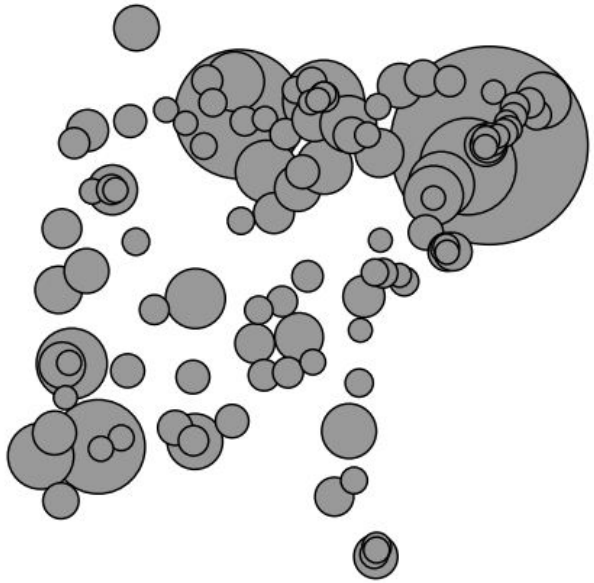
(a) Left-to-right by center.



(b) Left-to-right by leftmost.

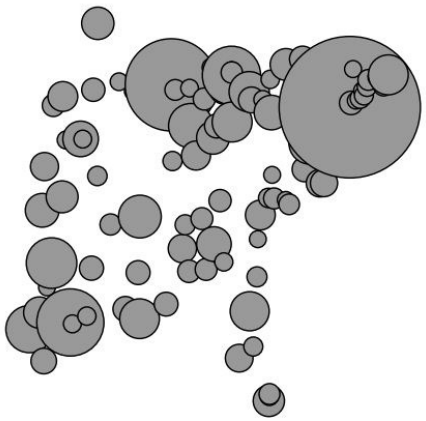


(c) Large-to-small.

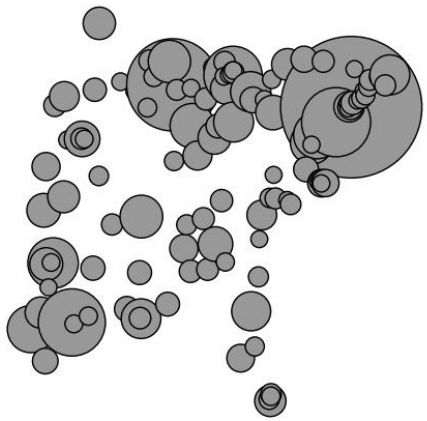


(d) Max-Min.

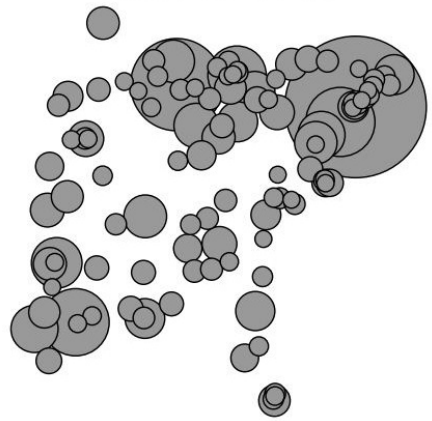
Evaluation



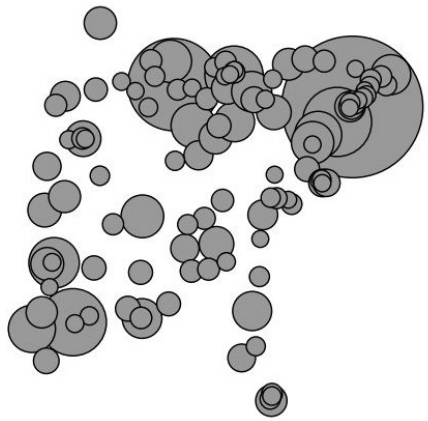
(a) Left-to-right by center.



(b) Left-to-right by leftmost.



(c) Large-to-small.



(d) Max-Min.

	Top-10 average		Total boundary	
	Absolute	(Relative)	Absolute	(Relative)
Left-to-right by center	0.00	(0.00%)	1405	(57.76%)
Left-to-right by leftmost	2.14	(21.48%)	1711	(74.84%)
Large-to-small	2.72	(25.48%)	1730	(78.21%)
Max-Min	4.42	(43.03%)	1759	(79.33%)


THE
END