Seminar Approximationsalgorithmen

SET COVER

David Münch

Universität Karlsruhe (TH)
Fakultät für Informatik
ITI Wagner

12. Juli 2008

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

IBM findet Computer Viren

- 5000 Viren bekannt. (Elemente)
- 9000 Substrings von 20+ fortlaufenden Bytes von Viren nicht in sauberem Code bekannt. (Mengen)

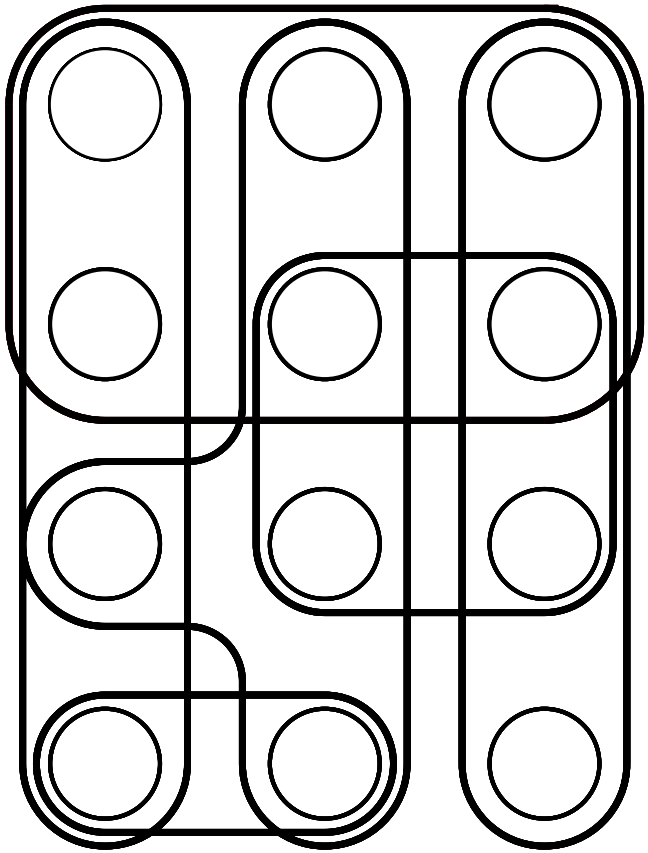
IBM findet Computer Viren

- 5000 Viren bekannt. (Elemente)
- 9000 Substrings von 20+ fortlaufenden Bytes von Viren nicht in sauberem Code bekannt. (Mengen)
- Suche nach 180 von 9000 Substrings reichen aus, um den Virenbefall nachzuweisen. (Set Cover)

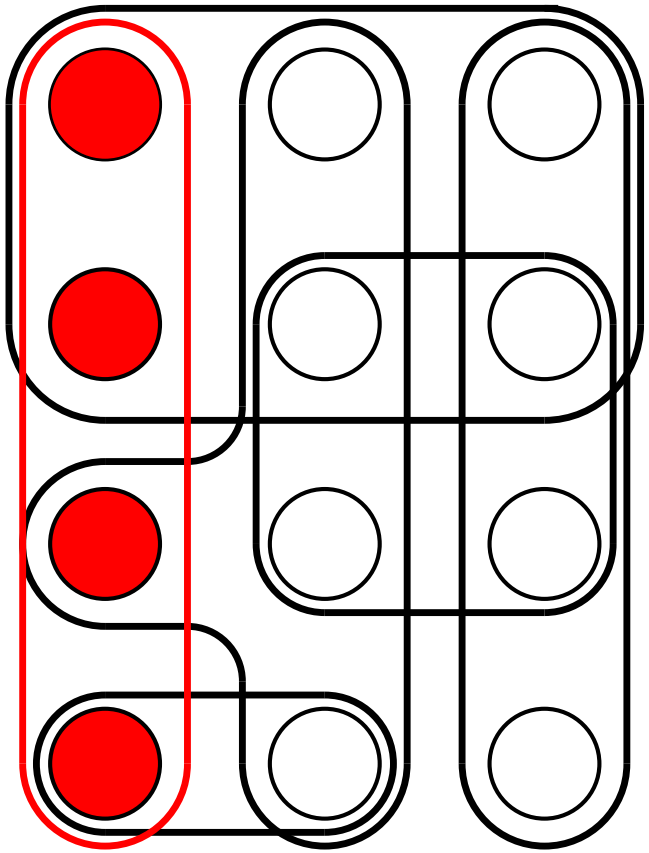
Definition: SET COVER

Gegeben:
Ein Universum U mit n Elementen, eine Menge von Teilmengen aus U: $\mathcal{S} = S_1, ..., S_k$ und eine Kostenfunktion $c : \mathcal{S} \to \mathbb{Q}^+$.
Gesucht:
Eine Teilmenge von $\mathcal{S}$, die alle Elemente von U mit minimalen Kosten überdeckt.

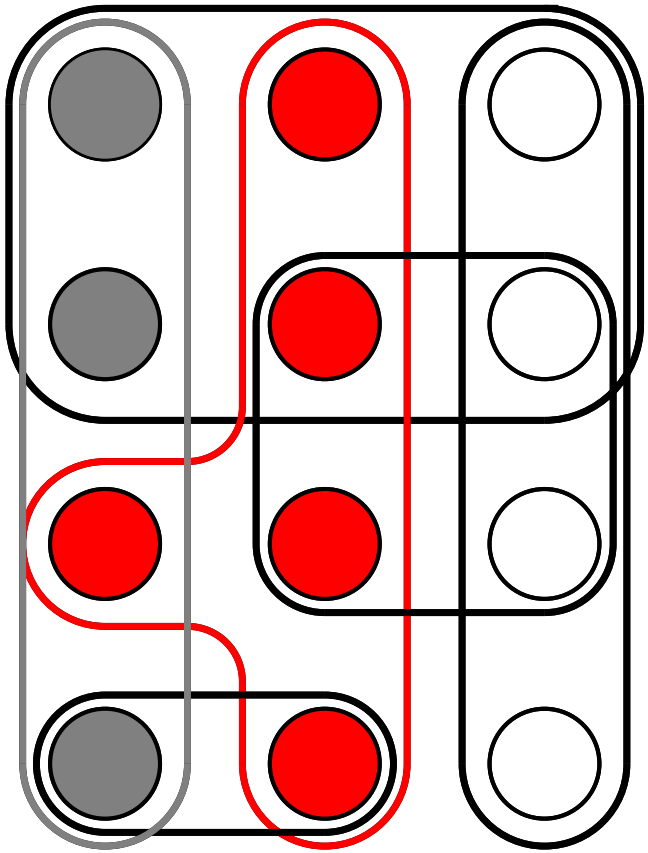
Beispiel



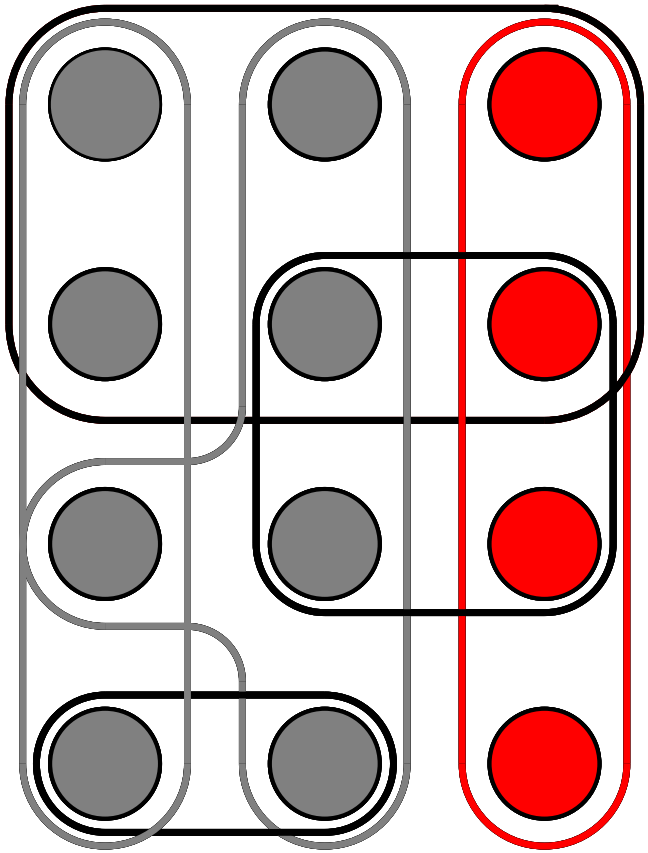
Beispiel



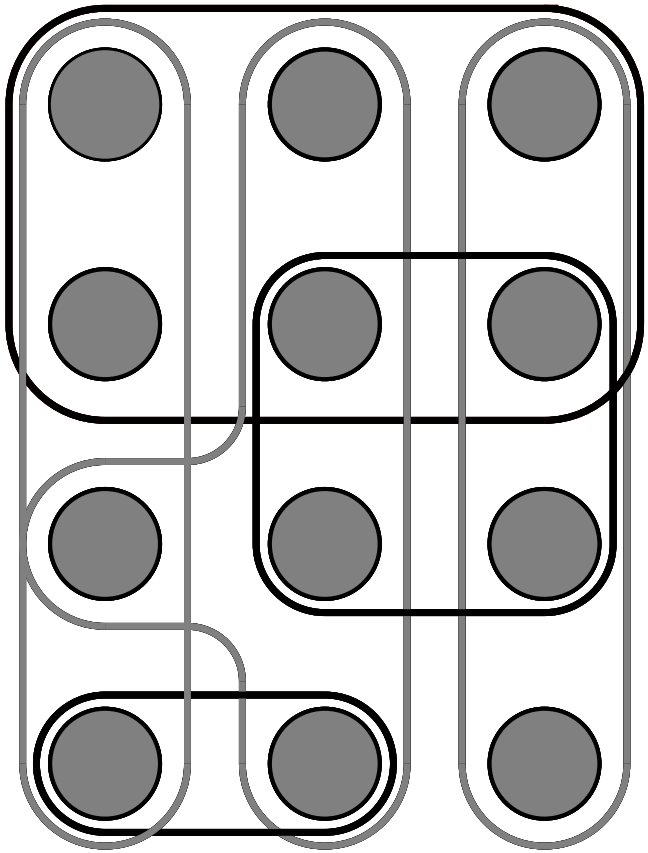
Beispiel



Beispiel



Beispiel



1972 zeigt Richard M. Karp, dass SET COVER NP-vollständig ist.

1972 zeigt Richard M. Karp, dass SET COVER NP-vollständig ist.

$$\text{SAT} \propto \text{CLIQUE} \propto \text{NODE (VERTEX) COVER} \propto \text{SET COVER}$$

1972 zeigt Richard M. Karp, dass SET COVER NP-vollständig ist.

$$\text{SAT} \propto \text{CLIQUE} \propto \text{NODE (VERTEX) COVER} \propto \text{SET COVER}$$

Falls $\mathcal{P} \neq \mathcal{NP}$, sind NP-vollständige Probleme nicht effizient lösbar.

Greedy Set Cover Algorithmus

$C \leftarrow \emptyset$

while $C \neq U$ **do**

Wähle die Menge S mit den geringsten Kosten.

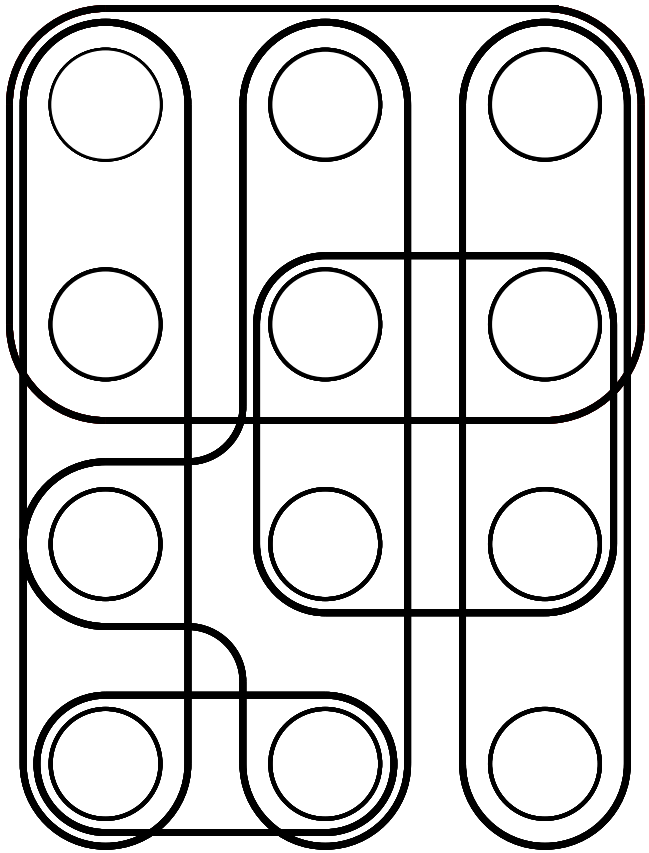
Seien z.B. $\alpha = \frac{c(S)}{|S-C|}$ die Kosten von S.

Nimm S, und für alle $e \in S_C$, setze $price(e) = \alpha$.

$C \leftarrow C \cup S$.

end

Ausgabe der gewählten Mengen.



Greedy Set Cover Algorithmus

$C \leftarrow \emptyset$

while $C \neq U$ **do**

Wähle die Menge S mit den geringsten Kosten.

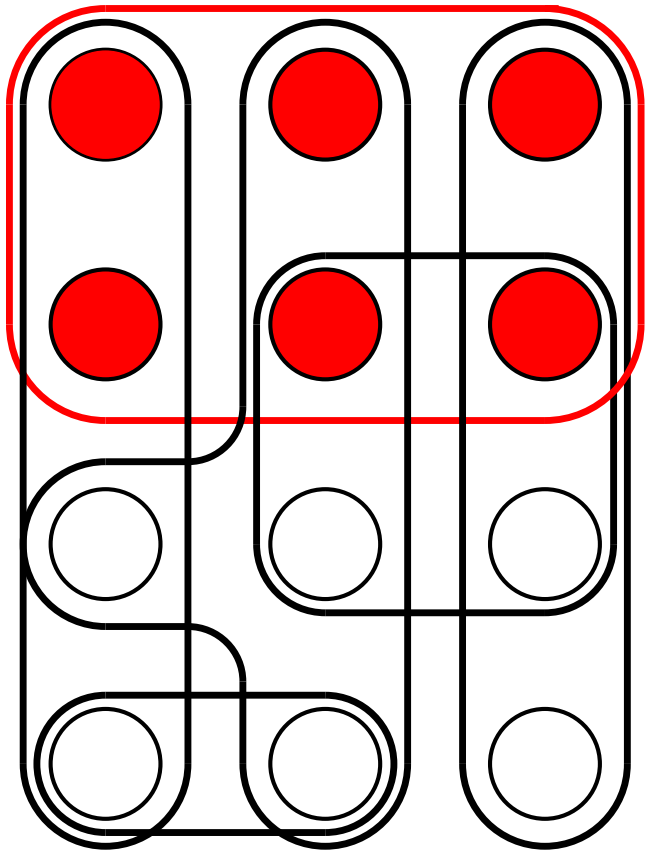
Seien z.B. $\alpha = \frac{c(S)}{|S-C|}$ die Kosten von S.

Nimm S, und für alle $e \in S_C$, setze $price(e) = \alpha$.

$C \leftarrow C \cup S$.

end

Ausgabe der gewählten Mengen.



Greedy Set Cover Algorithmus

$C \leftarrow \emptyset$

while $C \neq U$ **do**

Wähle die Menge S mit den geringsten Kosten.

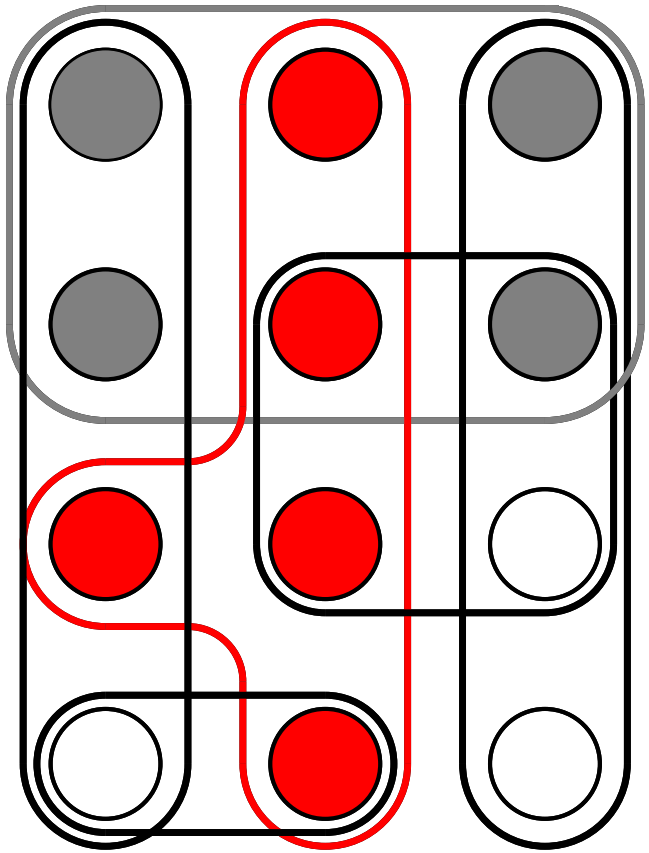
Seien z.B. $\alpha = \frac{c(S)}{|S-C|}$ die Kosten von S.

Nimm S, und für alle $e \in S_C$, setze $price(e) = \alpha$.

$C \leftarrow C \cup S$.

end

Ausgabe der gewählten Mengen.



Greedy Set Cover Algorithmus

$C \leftarrow \emptyset$

while $C \neq U$ **do**

Wähle die Menge S mit den geringsten Kosten.

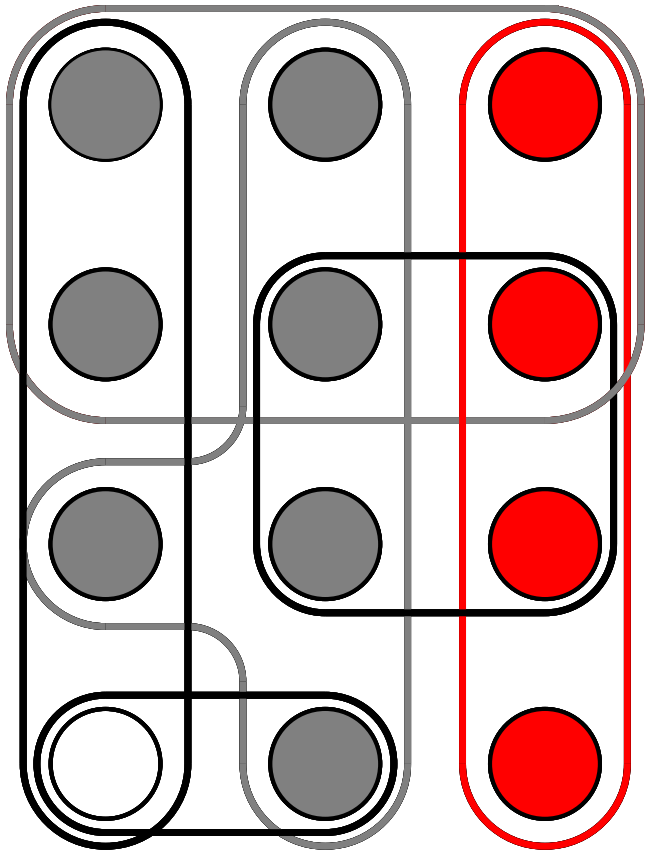
Seien z.B. $\alpha = \frac{c(S)}{|S-C|}$ die Kosten von S.

Nimm S, und für alle $e \in S_C$, setze $price(e) = \alpha$.

$C \leftarrow C \cup S$.

end

Ausgabe der gewählten Mengen.



Greedy Set Cover Algorithmus

$C \leftarrow \emptyset$

while $C \neq U$ **do**

Wähle die Menge S mit den geringsten Kosten.

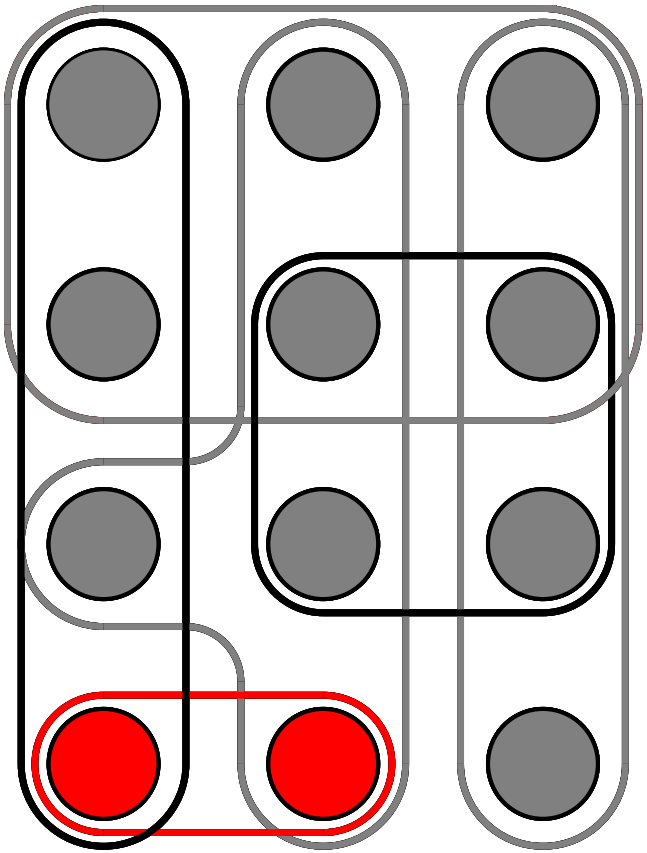
Seien z.B. $\alpha = \frac{c(S)}{|S-C|}$ die Kosten von S.

Nimm S, und für alle $e \in S_C$, setze $price(e) = \alpha$.

$C \leftarrow C \cup S$.

end

Ausgabe der gewählten Mengen.



Greedy Set Cover Algorithmus

$C \leftarrow \emptyset$

while $C \neq U$ **do**

Wähle die Menge S mit den geringsten Kosten.

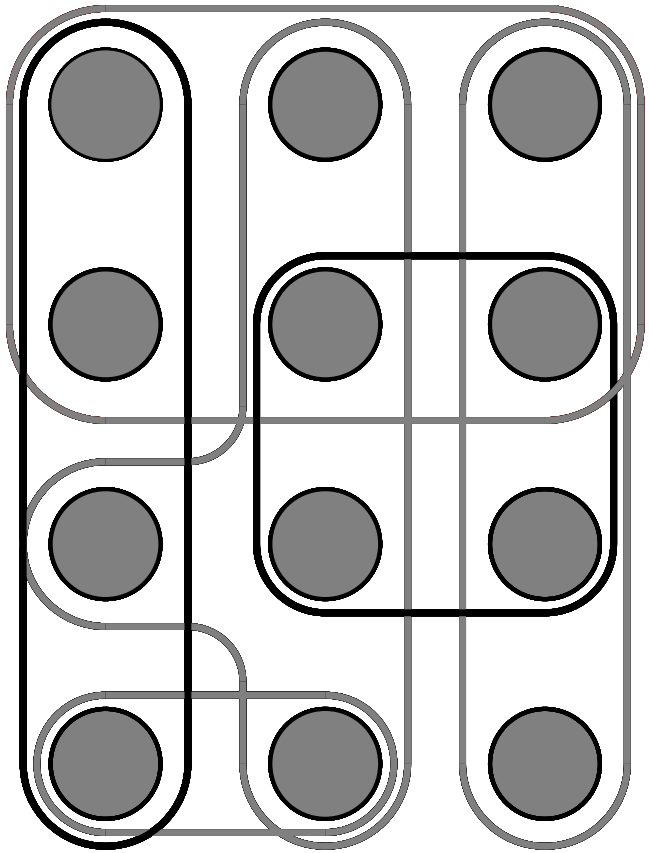
Seien z.B. $\alpha = \frac{c(S)}{|S-C|}$ die Kosten von S.

Nimm S, und für alle $e \in S_C$, setze $price(e) = \alpha$.

$C \leftarrow C \cup S$.

end

Ausgabe der gewählten Mengen.



Lemma 1

Für jedes $k \in \{1, ..., n\}$ ist $price(e_k) \leq \frac{OPT}{n-k+1}$

Bezeichne die Elemente von U nach der Reihenfolge, mit der sie der Greedy Set Cover Algorithmus auswählt, mit e_k, wobei $k \in \{1, ..., n\}$.

Theorem 1

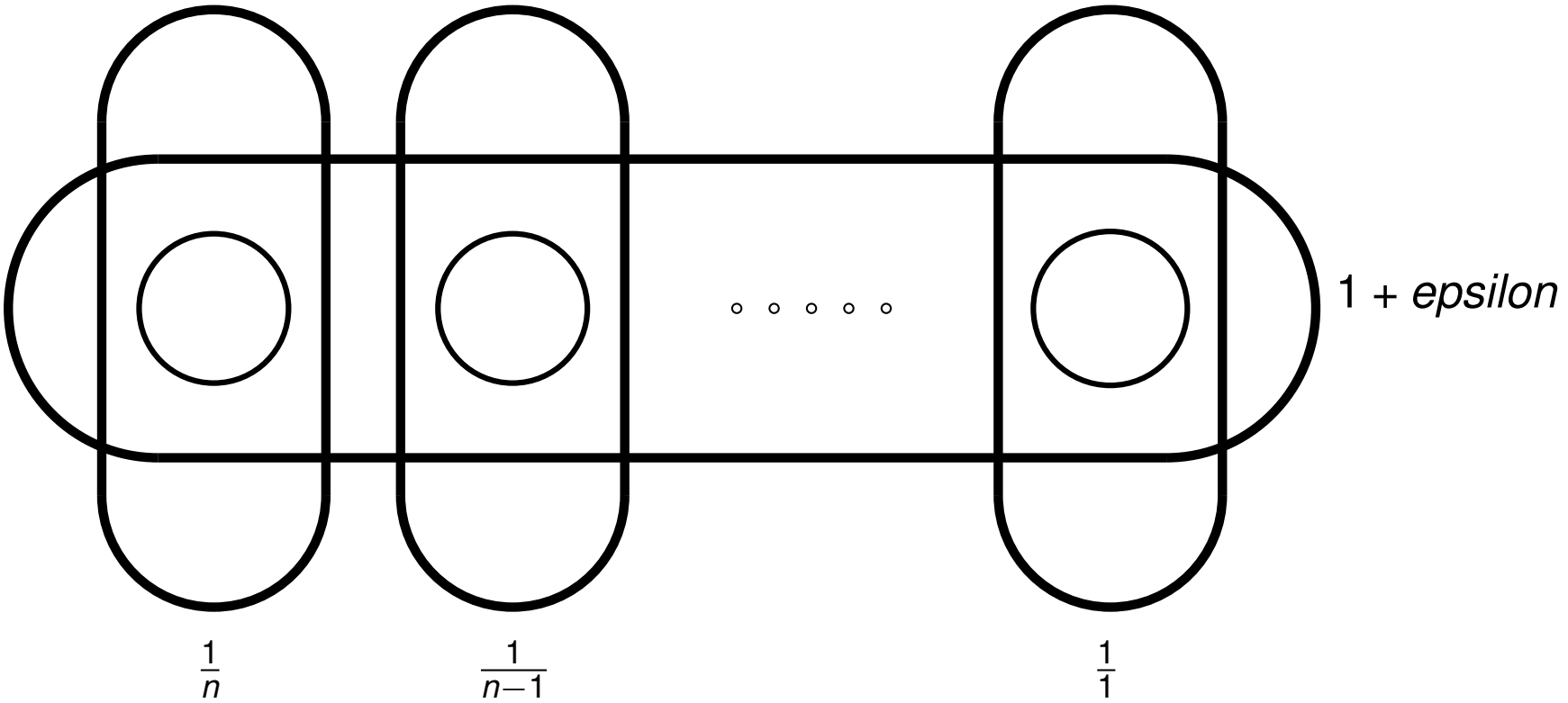
Greedy Set Cover Algorithmus ist ein Faktor-H_n-Approximationsalgorithmus für das zu minimierende SET COVER Problem, wobei $H_n = 1 + \frac{1}{2} + ... + \frac{1}{n} \leq ln(n) + 1$

Theorem 1

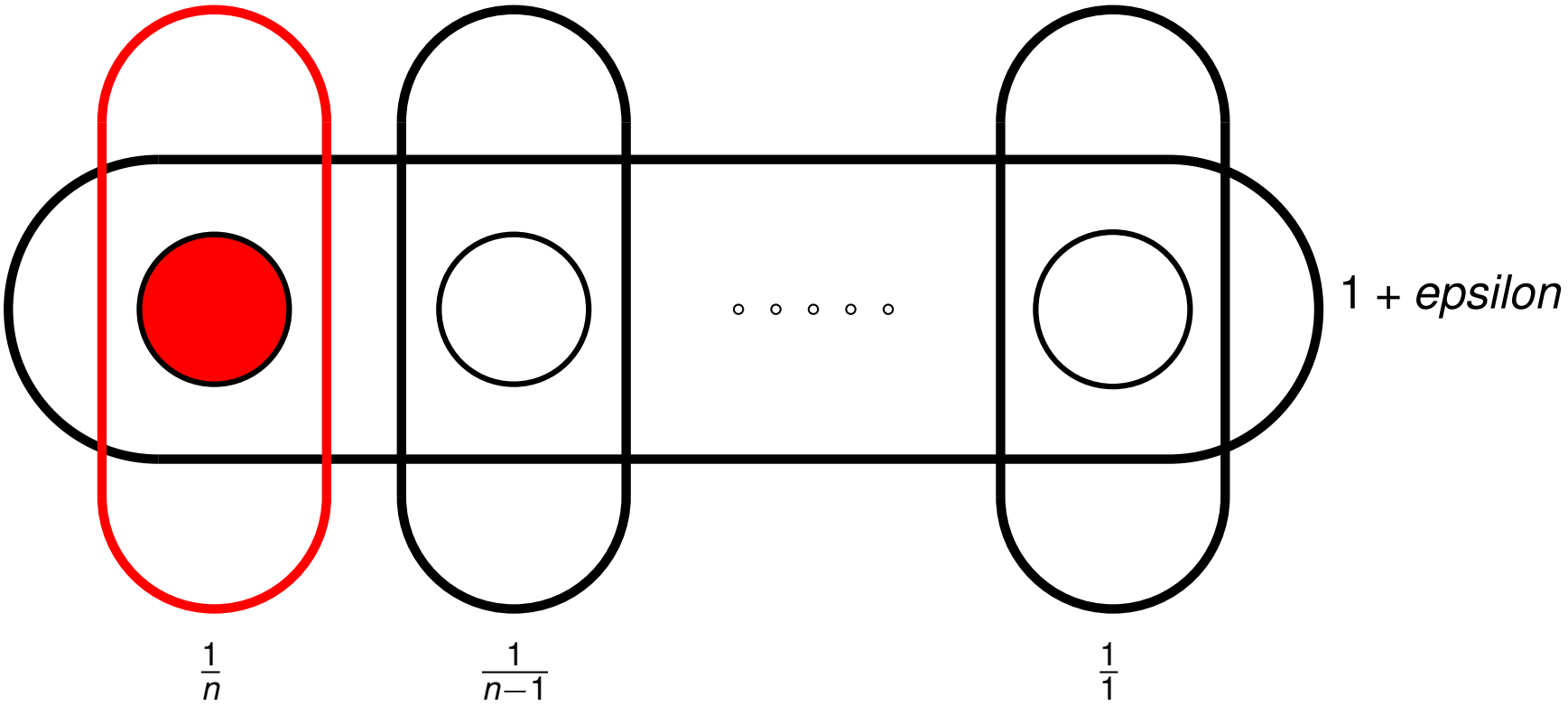
Greedy Set Cover Algorithmus ist ein Faktor-H_n-Approximationsalgorithmus für das zu minimierende SET COVER Problem, wobei $H_n = 1 + \frac{1}{2} + ... + \frac{1}{n} \leq ln(n) + 1$

Beweis:
Die Gesamtkosten des Set Covers entsprechen $\sum_{k=1}^{n} price(e_k)$.
Somit nach Lemma 1 maximal $(1 + \frac{1}{2} + ... + \frac{1}{n}) \cdot OPT$. $\square$

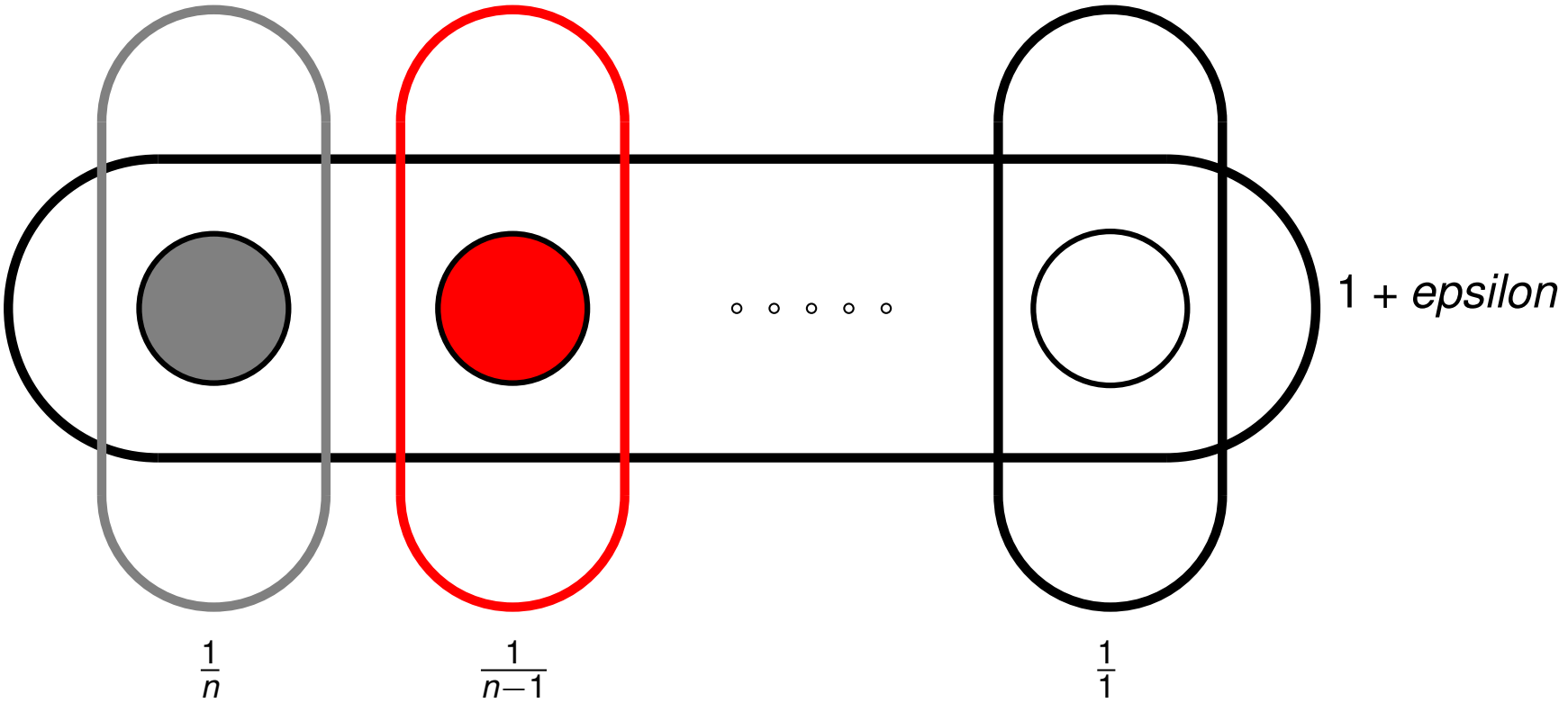
Maximales Gegenbeispiel



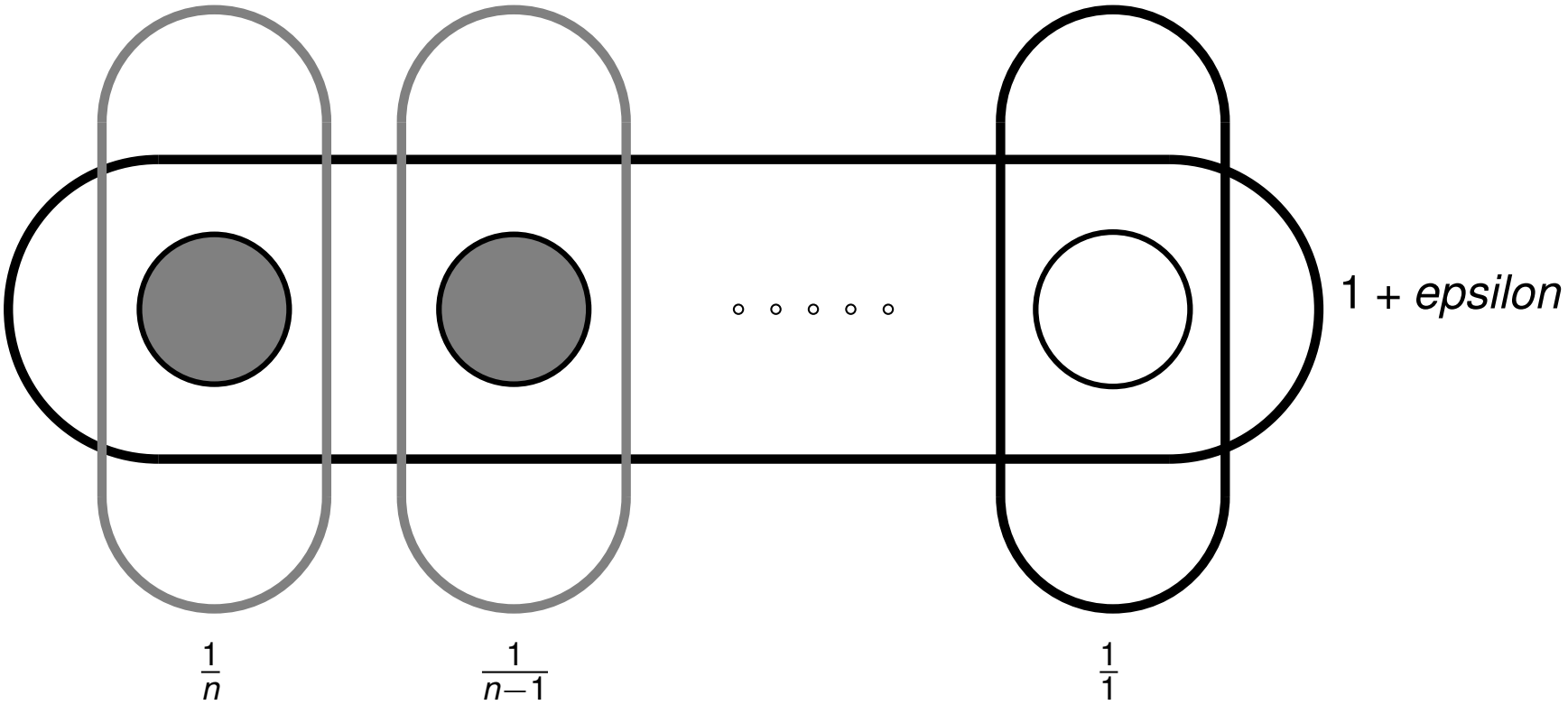
Maximales Gegenbeispiel



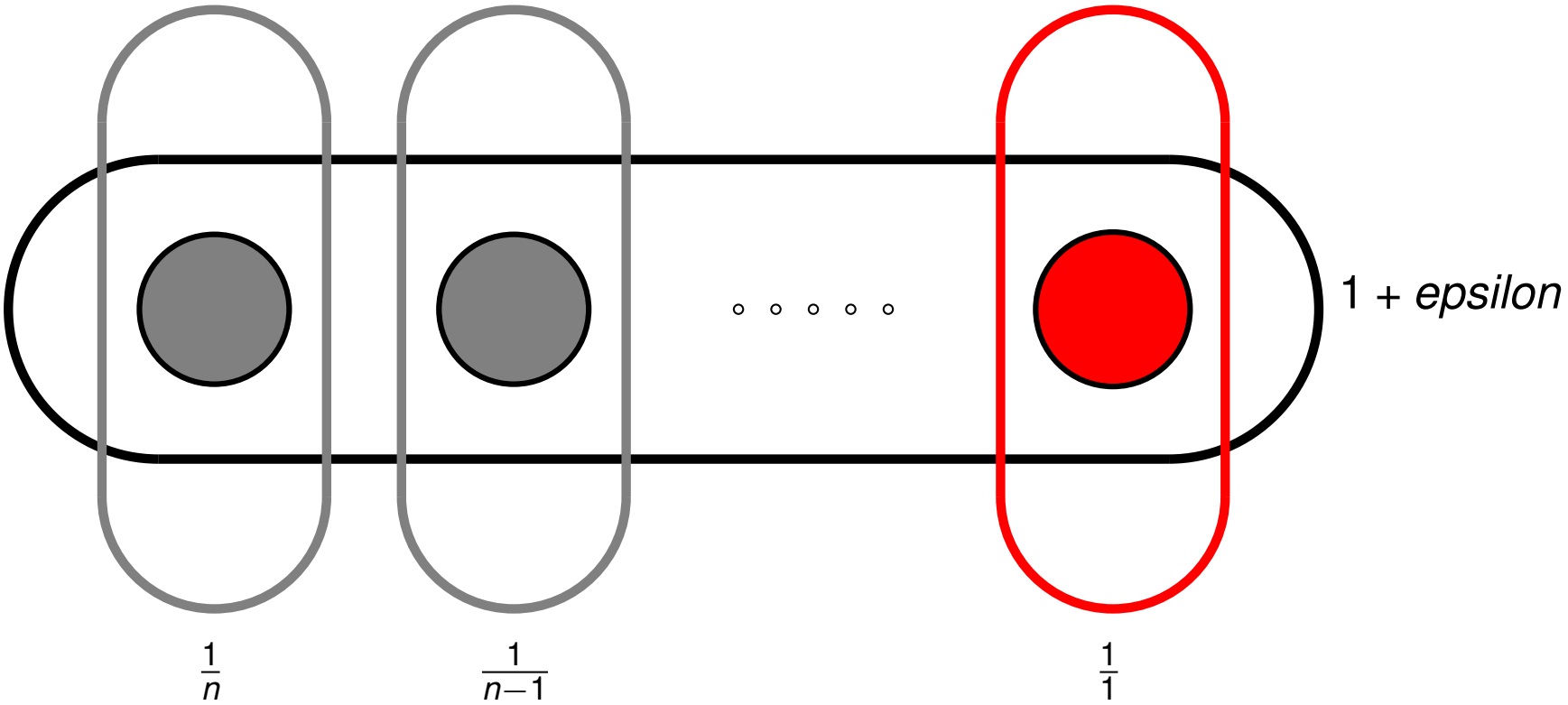
Maximales Gegenbeispiel



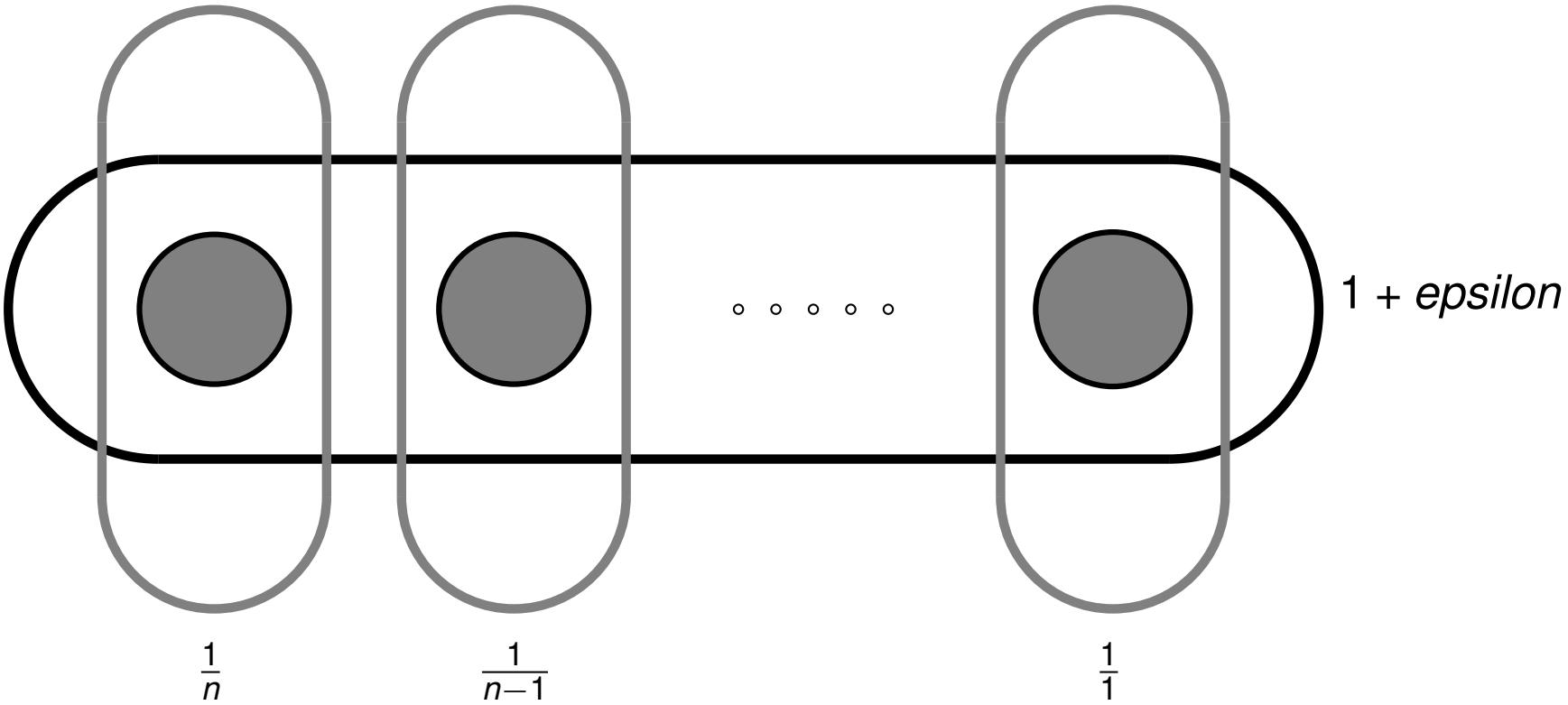
Maximales Gegenbeispiel



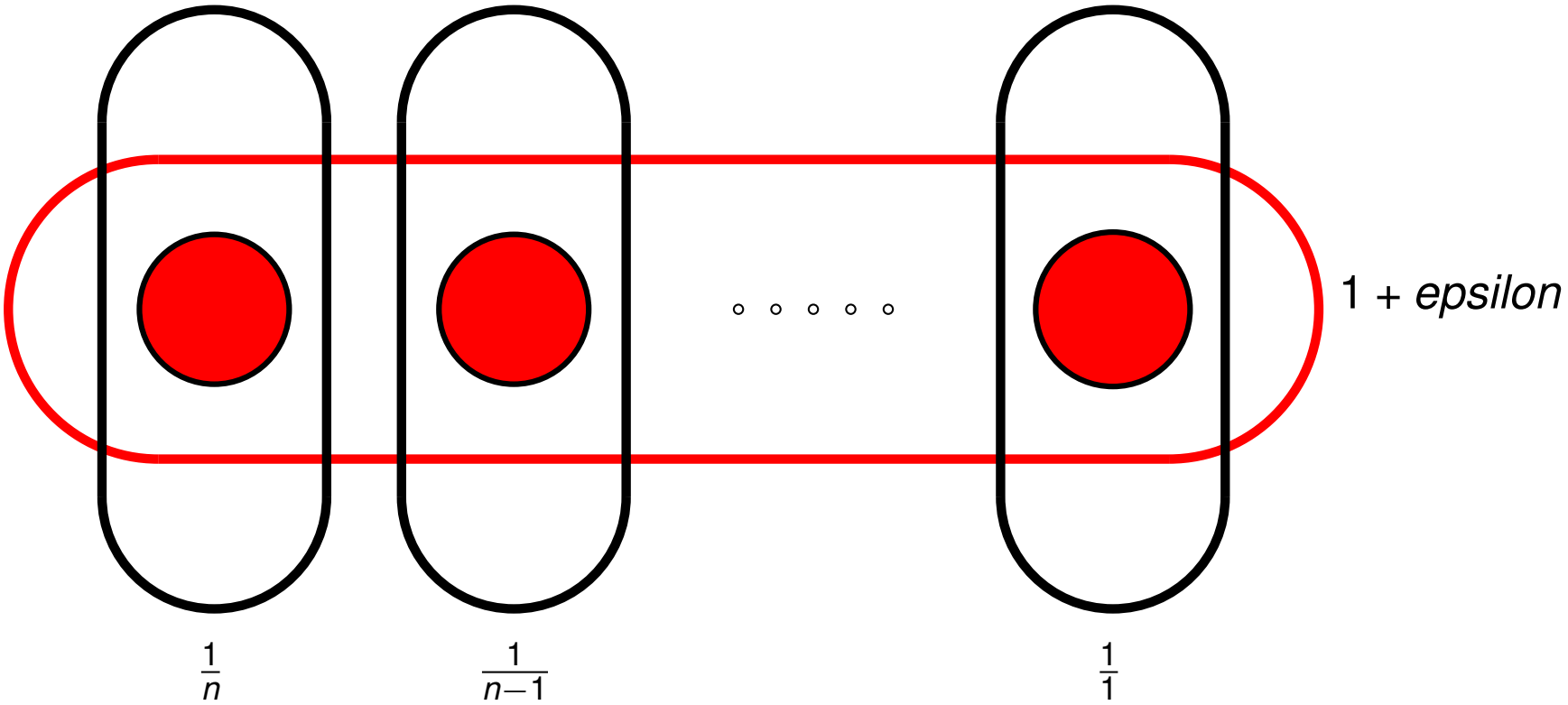
Maximales Gegenbeispiel



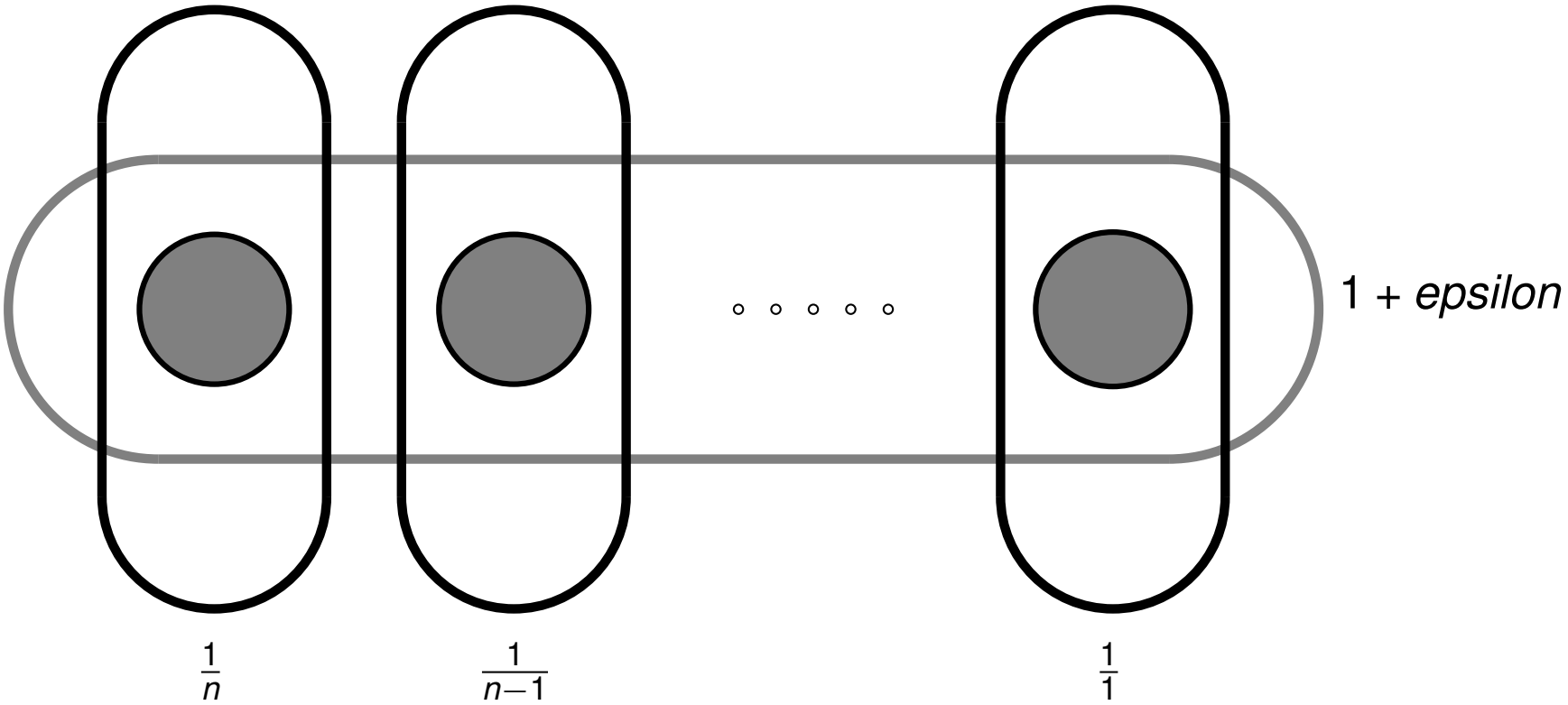
Maximales Gegenbeispiel



Maximales Gegenbeispiel



Maximales Gegenbeispiel



Graph Layering

Unkonventionelles Designkonzept für Algorithmen.

Einführung der Layering-Technik anhand VERTEX COVER.

Transformation auf einen Faktor-f-Approximationsalgorithmus für SET COVER.

Layering Vertex Cover

$D, W \leftarrow \emptyset$, $i = 0$, $G_0 = G$

while $G_i \neq \emptyset$ **do**

Entferne alle Knoten v mit $deg(v) = 0$ aus G_i und füge sie in neue Menge D_i ein.

$c = min\{\frac{w(v)}{deg(v)}\}$

$t(v) = c \cdot deg(v)$

$w'(v) = w(v) - t(v)$.

Alle Knoten mit $w'(v) = 0$ kommen in die Menge W_i

$G_{i+1} = V - (D_i \cup W_i)$, $i++$

end

Ausgabe des Vertex Covers

$C = W_0 \cup ... \cup W_{k-1}$.

G_0

Layering Vertex Cover

$D, W \leftarrow \emptyset$, $i = 0$, $G_0 = G$

while $G_i \neq \emptyset$ **do**

Entferne alle Knoten v mit $deg(v) = 0$ aus G_i und füge sie in neue Menge D_i ein.

$c = min\{\frac{w(v)}{deg(v)}\}$

$t(v) = c \cdot deg(v)$

$w'(v) = w(v) - t(v)$.

Alle Knoten mit $w'(v) = 0$ kommen in die Menge W_i

$G_{i+1} = V - (D_i \cup W_i)$, $i++$

end

Ausgabe des Vertex Covers

$C = W_0 \cup ... \cup W_{k-1}$.

Layering Vertex Cover

$D, W \leftarrow \emptyset$, $i = 0$, $G_0 = G$

while $G_i \neq \emptyset$ **do**

Entferne alle Knoten v mit $deg(v) = 0$ aus G_i und füge sie in neue Menge D_i ein.

$c = min\{\frac{w(v)}{deg(v)}\}$

$t(v) = c \cdot deg(v)$

$w'(v) = w(v) - t(v)$.

Alle Knoten mit $w'(v) = 0$ kommen in die Menge W_i

$G_{i+1} = V - (D_i \cup W_i)$, $i++$

end

Ausgabe des Vertex Covers

$C = W_0 \cup ... \cup W_{k-1}$.

Layering Vertex Cover

$D, W \leftarrow \emptyset$, $i = 0$, $G_0 = G$

while $G_i \neq \emptyset$ **do**

Entferne alle Knoten v mit $deg(v) = 0$ aus G_i und füge sie in neue Menge D_i ein.

$c = min\{\frac{w(v)}{deg(v)}\}$

$t(v) = c \cdot deg(v)$

$w'(v) = w(v) - t(v)$.

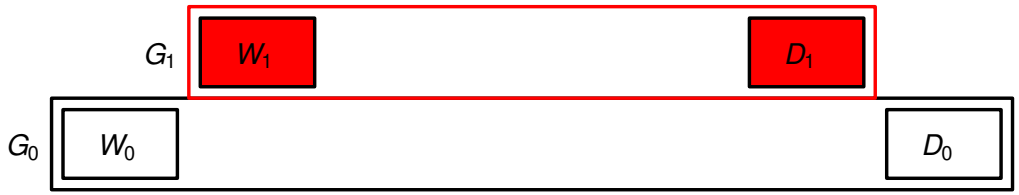
Alle Knoten mit $w'(v) = 0$ kommen in die Menge W_i

$G_{i+1} = V - (D_i \cup W_i)$, $i++$

end

Ausgabe des Vertex Covers

$C = W_0 \cup ... \cup W_{k-1}$.



Layering Vertex Cover

$D, W \leftarrow \emptyset$, $i = 0$, $G_0 = G$

while $G_i \neq \emptyset$ **do**

Entferne alle Knoten v mit $deg(v) = 0$ aus G_i und füge sie in neue Menge D_i ein.

$c = min\{\frac{w(v)}{deg(v)}\}$

$t(v) = c \cdot deg(v)$

$w'(v) = w(v) - t(v)$.

Alle Knoten mit $w'(v) = 0$ kommen in die Menge W_i

$G_{i+1} = V - (D_i \cup W_i)$, $i++$

end

Ausgabe des Vertex Covers

$C = W_0 \cup ... \cup W_{k-1}$.

Layering Vertex Cover

$D, W \leftarrow \emptyset$, $i = 0$, $G_0 = G$

while $G_i \neq \emptyset$ **do**

Entferne alle Knoten v mit $deg(v) = 0$ aus G_i und füge sie in neue Menge D_i ein.

$c = min\{\frac{w(v)}{deg(v)}\}$

$t(v) = c \cdot deg(v)$

$w'(v) = w(v) - t(v)$.

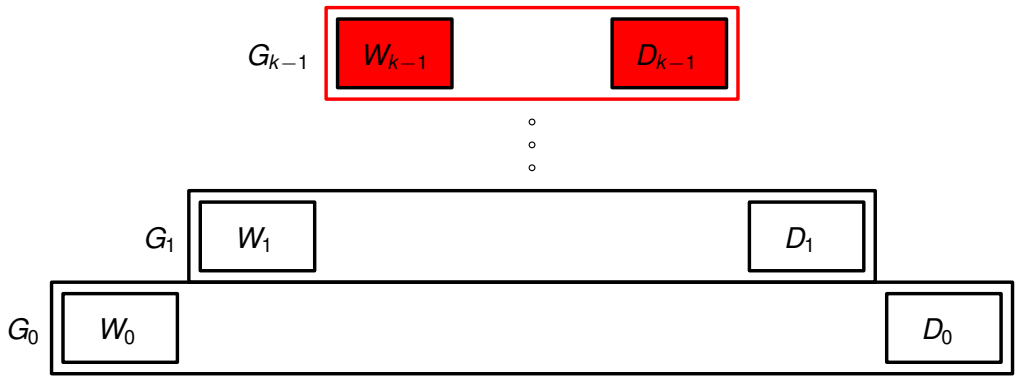
Alle Knoten mit $w'(v) = 0$ kommen in die Menge W_i

$G_{i+1} = V - (D_i \cup W_i)$, $i++$

end

Ausgabe des Vertex Covers

$C = W_0 \cup ... \cup W_{k-1}$.



Layering Vertex Cover

$D, W \leftarrow \emptyset$, $i = 0$, $G_0 = G$

while $G_i \neq \emptyset$ **do**

Entferne alle Knoten v mit $deg(v) = 0$ aus G_i und füge sie in neue Menge D_i ein.

$c = min\{\frac{w(v)}{deg(v)}\}$

$t(v) = c \cdot deg(v)$

$w'(v) = w(v) - t(v)$.

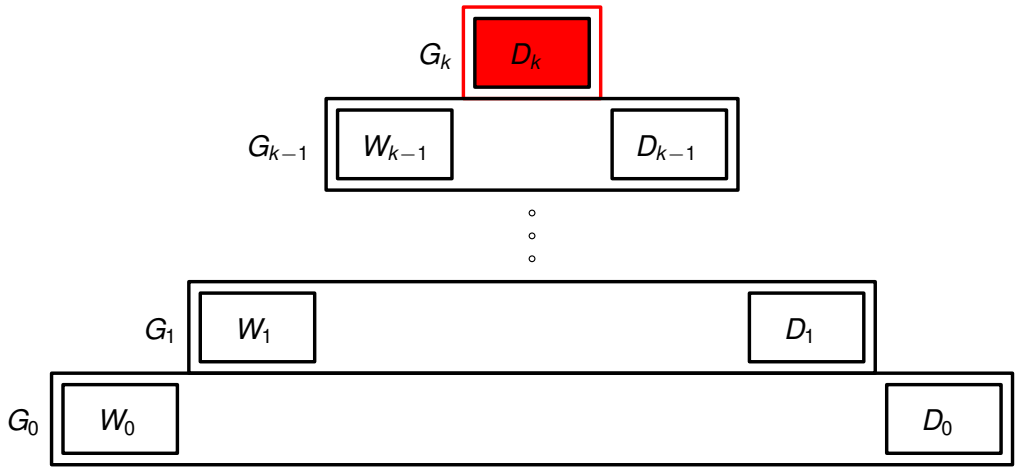
Alle Knoten mit $w'(v) = 0$ kommen in die Menge W_i

$G_{i+1} = V - (D_i \cup W_i)$, $i++$

end

Ausgabe des Vertex Covers

$C = W_0 \cup ... \cup W_{k-1}$.



Layering Vertex Cover

$D, W \leftarrow \emptyset$, $i = 0$, $G_0 = G$

while $G_i \neq \emptyset$ **do**

Entferne alle Knoten v mit $deg(v) = 0$ aus G_i und füge sie in neue Menge D_i ein.

$c = min\{\frac{w(v)}{deg(v)}\}$

$t(v) = c \cdot deg(v)$

$w'(v) = w(v) - t(v)$.

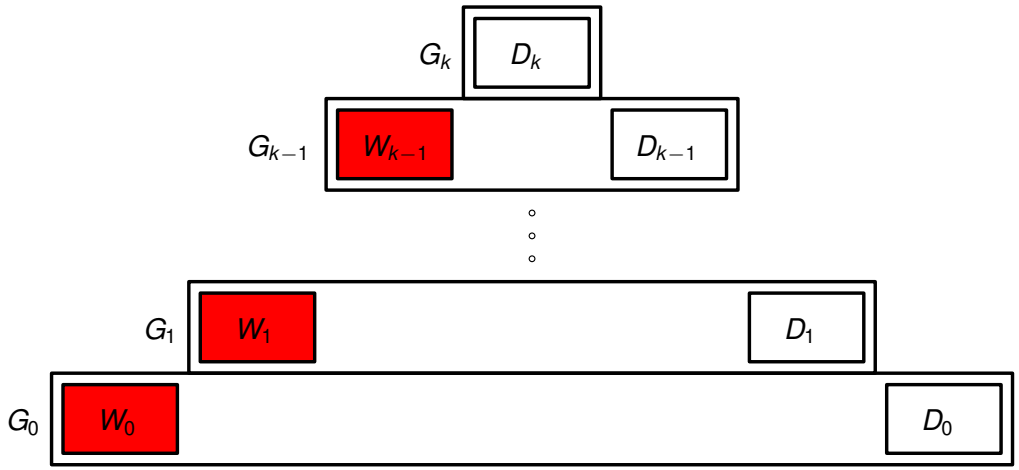
Alle Knoten mit $w'(v) = 0$ kommen in die Menge W_i

$G_{i+1} = V - (D_i \cup W_i)$, $i++$

end

Ausgabe des Vertex Covers

$C = W_0 \cup ... \cup W_{k-1}$.



Theorem 2

Layering Vertex Cover Algorithmus ist ein Faktor-2-Approximationsalgorithmus für das zu minimierende VERTEX COVER Problem bei beliebigen Knotengewichten $w(v)$.

SET COVER als ILP

$$\text{minimiere} \sum_{S \in \mathcal{S}} c(S) x_S$$

$$\text{in Abhängigkeit von} \sum_{S : e \in S} x_S \geq 1, e \in U$$

$$x_S \in \{0, 1\}, S \in \mathcal{S}$$

Relaxation mit $0 \leq x_S \leq 1$.
Gebrochene SET COVER Lösungen?

SET COVER als LP, Primales LP

$$\text{minimiere} \sum_{S \in \mathcal{S}} c(S) x_S$$

$$\text{unter den Nebenbedingungen} \sum_{S : e \in S} x_S \geq 1, e \in U$$

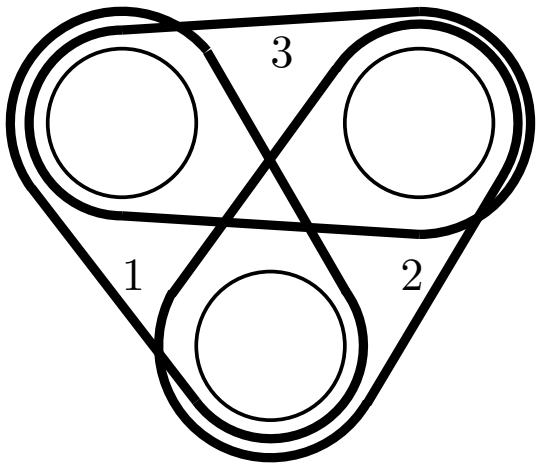
$$x_S \geq 0, S \in \mathcal{S}$$

Beispiel

Zu diesem Beispiel erhalten wir mit z.B. $c(S) = |S|$ folgendes LP:

minimiere $2x_1 + 2x_2 + 2x_3$
unter Nebenbedingungen:

$$x_1 + x_2 \geq 1$$
$$x_2 + x_3 \geq 1$$
$$x_1 + x_3 \geq 1$$
$$x_1, x_2, x_3 \geq 0$$



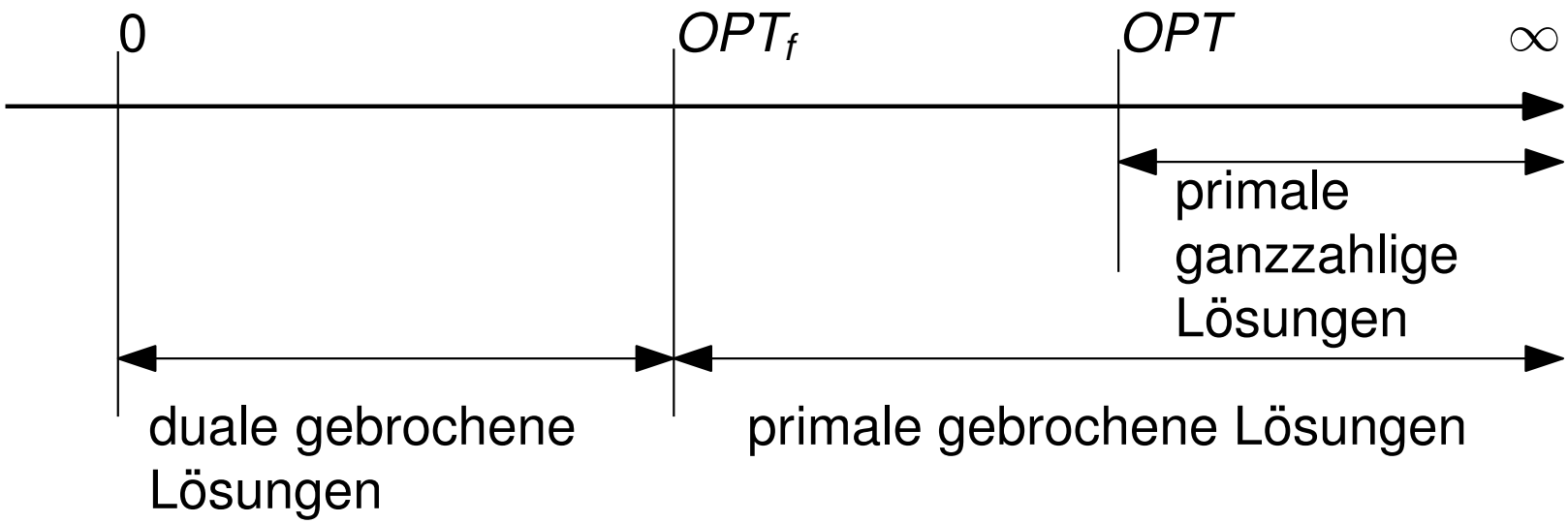
Optimale Lösung:
$x_1 = 0.5$, $x_2 = 0.5$ und $x_3 = 0.5$.

SET COVER als LP, Duales LP

$$\text{maximiere} \sum_{e \in U} y_e$$

$$\text{unter den Nebenbedingungen} \sum_{e:e \in S} y_e \leq c(S), S \in \mathcal{S}$$

$$y_e \geq 0, e \in U$$


0
OPT_f
OPT
∞
primale ganzzahlige Lösungen
duale gebrochene Lösungen
primale gebrochene Lösungen

Alternativer Beweis des Approximationsfaktors für den Greedy-Algorithmus

- Duale Lösung meistens keine gültige Lösung.
- Idee: Vermindere duale Lösung um Faktor H_n.
- Definiere $y_e = \frac{price(e)}{H_n}$
- y jetzt gültige duale Lösung.

Theorem 3

Greedy Set Cover Algorithmus ist ein Faktor-H_n-Approximationsalgorithmus für das zu minimierende SET COVER Problem.

SET COVER als LP

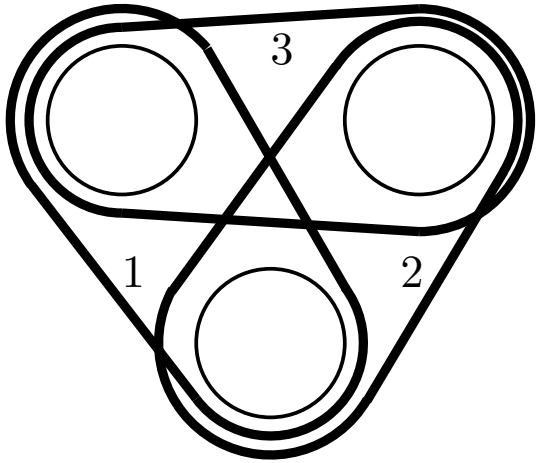
$$\text{minimiere} \sum_{S \in \mathcal{S}} c(S) x_S$$

$$\text{unter den Nebenbedingungen} \sum_{S: e \in S} x_S \geq 1, e \in U$$

$$x_S \geq 0, S \in \mathcal{S}$$

Problem:
Eine Menge S_i gehört $(x_{S_i} = 1)$ oder gehört nicht $(x_{S_i} = 0)$ zu einer möglichen Lösung.
Reelle Zahlen als Lösung des LP möglich.

Beispiel



Optimale Lösung mit Simplexverfahren:
$x_1 = 0.5$, $x_2 = 0.5$ und $x_3 = 0.5$.

Möglicher Lösungsansatz:

Runde alle Werte des Lösungsvektors $x_S > 0$ auf Eins.

Möglicher Lösungsansatz:
Runde alle Werte des Lösungsvektors $x_S > 0$ auf Eins.

Noch besser:

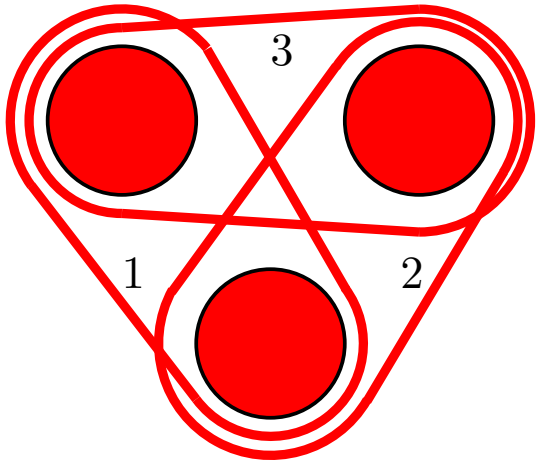
SET COVER MIT LP-RUNDEN

Finde eine optimale Lösung für die LP-Relaxation.
Nimm alle Mengen S mit $x_S \geq 1/f$ in die Lösungsmenge.

Definition: f

f ist die maximale Anzahl in wie vielen verschienden Mengen ein Element $e \in U$ gleichzeitig ist.

Beispiel



Theorem 4

Set cover mit LP-Runden Algorithmus ist ein Faktor-f-Approximationsalgorithmus für das SET COVER Problem.

Fazit:

LP mit Ellipsoidmethode oder Innere-Punkte-Verfahren in polynomieller Zeit lösbar.
Das Runden benötigt lineare Zeit.

Neue Idee:

Optimale gebrochene Lösung als Wahrscheinlichkeiten auffassen.

Neue Idee:
Optimale gebrochene Lösung als Wahrscheinlichkeiten auffassen.

Wir wollen es versuchen:

RANDOMISIERTES RUNDEN BEI SET COVER

Finde eine optimale Lösung für die LP-Relaxation.

repeat

> Werfe Münze für jede Menge S mit der optimalen gebrochenen Lösung als Wahrscheinlichkeit.
> Nimm alle Mengen S die ausgewählt wurden in die Lösungsmenge.

until *Gültige Lösung gefunden.*

Theorem 5

Randomisiertes Runden bei Set Cover Algorithmus ist ein $\mathcal{O}(\log n)$-Faktor-Apprixiomationsalgorithmus.

Beweis:
Nicht hier. Siehe VAZIRANI Kap. 14.2.
Mit dem Erwartungswert der ausgewählten Mengen S und der Wahrscheinlichkeitsverteilung, ob ein Element von einer Menge überdeckt wird, kann man Theorem 5 zeigen.□

Das Primal-Duale Schema ist die Methode der Wahl um einen schnellen und guten Approximationsalgorithmus zu konstruieren.

Zuerst Vorstellen des allgemeinen Primal-Dualen Schemas

Dann SET COVER mit dem Primal-Dualen-Schema.

Primales und Duales LP in Standardform

Primales LP

minimiere $\sum_{j=1}^{n} c_j x_j$

$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \;\; i = 1, .., m$

$x_j \geq 0, \;\; j = 1, ..., n$

Duales LP

maximiere $\sum_{j=1}^{n} b_i y_i$

unter den Nebenbedingungen

$\sum_{i=1}^{m} a_{ij} y_i \leq c_j, \;\; j = 1, .., n$

$y_i \geq 0, \;\; i = 1, ..., m$

Definition: komplementäre Schlupfbeingungen

Primale komplementäre Schlupfbedingung

Sei $\alpha \geq 1$.

$$\forall j : 1 \leq j \leq n : \text{ entweder } x_j = 0 \text{ oder } c_j/\alpha \leq \sum_{i=1}^{m} a_{ij} y_i \leq c_j$$

Duale komplementäre Schlupfbedingung

Sei $\beta \geq 1$.

$$\forall i : 1 \leq i \leq m : \text{ entweder } y_i = 0 \text{ oder } b_i \leq \sum_{j=1}^{n} a_{ij} x_j \leq \beta b_i$$

Satz 1

Wenn x und y gültige Lösungen für das primale und duale LP sind, dann gilt, ausgehend von den komplementären Schlupfbedingungen:

$$\sum_{j=1}^{n} c_j x_j \leq \alpha \cdot \beta \cdot \sum_{i=1}^{m} b_i y_i$$

Heisst:
Die gültige Lösung ist maximal um $\alpha \cdot \beta$ schlechter als die optimale Lösung.
Folglich hier Aproximationsfaktor von $\alpha\beta$.

Primal-Dualer Algorithmus

Beginne mit einer ungültigen primalen Lösung und einer gültigen dualen Lösung. Trivialerweise ist das $x = 0, y = 0$.

while *Satz 1 nicht erfüllt* **do**

> Verbessere die Gültigkeit der primalen und Optimalität der dualen Lösung abwechselnd.
>
> ```
> /* Lokale Verbesserungen gegenseitig um globales
>    Optimum zu erreichen.                          */
> ```

end

Ausgabe: optimale ganzzahlige Lösung

Besonderer Fall: Wenn beide komplementären Schlupfbedingungen erfüllt sind, dann ist die gültige Lösung optimal.

Konkrete Anwendung auf Set Cover

Wähle $\alpha = 1, \beta = f \Rightarrow$ Approximationsfaktor f.

Primale komplementäre Schlupfbedingung

$$\forall S \in \mathcal{S} : x_S \neq 0 \Rightarrow \sum_{e:e \in S} y_e = c(S)$$

Duale komplementäre Schlupfbedingung

$$\forall e : y_e \neq 0 \Rightarrow \sum_{S:e \in S} x_S \leq f$$

Ein Element kann maximal f Mal überdeckt werden, trivialerweise erfüllt.

Primal-Dualer Algorithmus für Set Cover

Beginne mit einer ungültigen primalen Lösung und einer gültigen dualen Lösung. Trivialerweise ist das $x = 0, y = 0$.

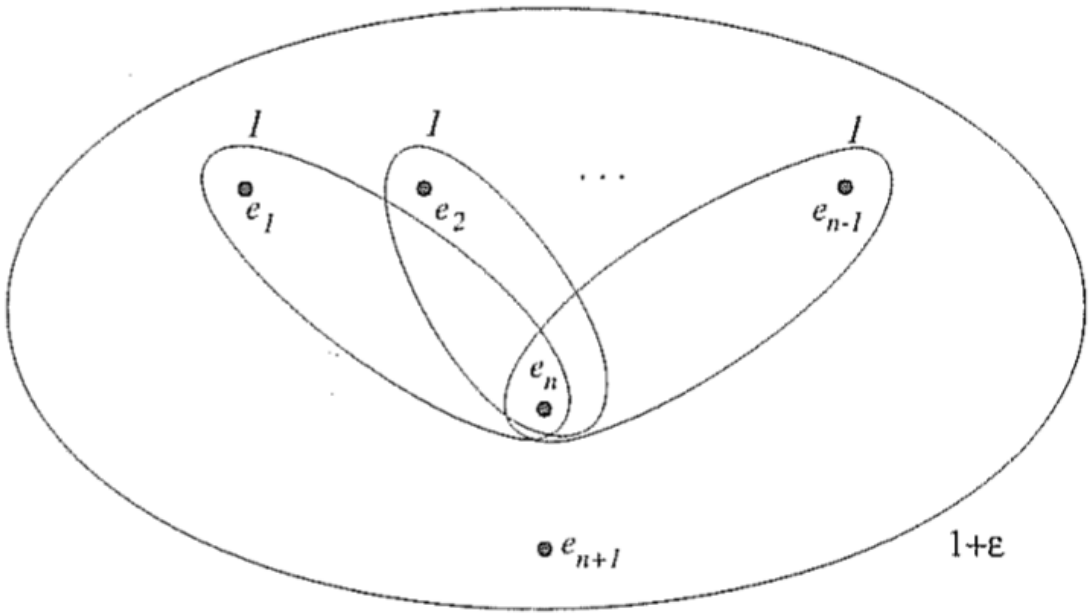
while *Noch nicht alle Elemente überdeckt* **do**

> Wähle ein noch nicht überdecktes Element e, erhöhe y_e, bis einige Mengen gesättigt.
>
> Nimm alle gesättigte Mengen in das Cover und aktualisiere x.
>
> Alle Elemente in diesen Mengen sind nun überdeckt.

end

Ausgabe: optimale ganzzahlige Lösung: Set Cover x.

Maximales Gegenbeispiel



Fazit

- Set Cover NP-Vollständig $\Rightarrow$ nicht optimal, effizient lösbar
- Greedy-Algorithmus in $\mathcal{O}(|C| \cdot n)$ und Approximationsfaktor H_n
- Formulierung von Set Cover als Lineares Programm, effizient lösbar. $\mathcal{O}(n^{3,5} L^2 \ln L \ln \ln L)$ mit n Variablen und L Eingabebits.
- einfacher LP-Runden Algorithmus Approximationsfaktor f
- randomisierter LP-Runden Algorithmus Approximationsfaktor $\mathcal{O}(log(n))$
- Primal-Dualer Algorithmus in $\mathcal{O}(n \cdot f)$, weil das LP nicht gelöst werden muss und Approximationsfaktor f

Quellen

Vijay V. Vazirani. Approximation Algorithms. Springer-Verlag, 2001. Kapitel: 2.1, 2.2, 13.1, 14, 15

http://www.cs.berkeley.edu/~luca/cs172/karp.pdf

http://people.orie.cornell.edu/~dpw/cornell.ps