Technische Universität Braunschweig

Algorithmen und Datenstrukturen 2 – Übung #4

Approximation: Greedy_k, Vertex Cover

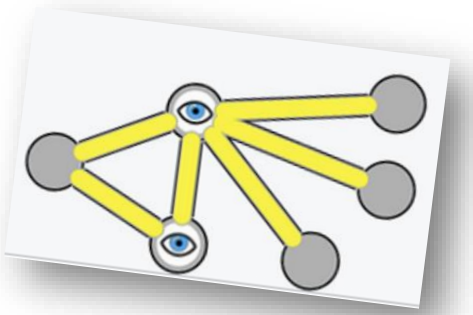
Ramin Kosfeld und Chek-Manh Loi

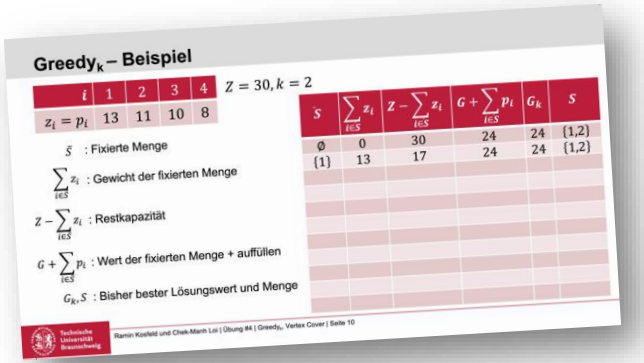
18.06.2025

Heute

Greedy$_k$

- Beispiel
- Worst-Case-Instanz





Vertex Cover

- Definition
- Approximation

Heute

Approximationsalgorithmen

Wir haben ein Problem, das wir nicht immer effizient optimal lösen können

Idee (Approximationsalgorithmen): Verlange keine Optimalität!

- Wollen aber nicht ‘irgendwas’ machen (z.B. irgendeine Greedy-Heuristik)
- Wollen auch im schlimmsten Fall noch ‘relativ gut’ sein → *Garantie!*

Was heißt ‘relativ gut’?

- Konstanter Approximationsfaktor c (nicht von Eingabe abhängig)
- Selbst auf schlimmster Instanz nicht weiter vom Optimum entfernt als Faktor c
- D.h. für Maximierungsprobleme $ALG(I) \geq c \cdot OPT(I)$ für alle Instanzen I $(\mathrm{c} \leq 1)$
- Polynomielle Laufzeit ($O(n^k)$ für nicht von der Eingabe abhängiges k)

Greedy$_k$

Algorithmus GREEDY$_K$

1. Teste jede Teilmenge $\bar{S}$ mit $|\bar{S}| \leq k$
2. Fülle $\bar{S}$ mit Greedy$_0$ auf
3. Aktualisiere Lösung bei Bedarf

Laufzeit

→ $O(n^k)$ zu testende Teilmengen

→ $O(n \log n)$ Zeit pro Teilmenge

Greedy$_k$

Algorithmus GREEDY$_K$

1. Teste jede Teilmenge $\bar{S}$ mit $|\bar{S}| \leq k$
2. Fülle $\bar{S}$ mit Greedy$_0$ auf
3. Aktualisiere Lösung bei Bedarf

Laufzeit

→ $O(n^k)$ zu testende Teilmengen

→ $O(n)$ Zeit pro Teilmenge

→ $O(1)$

→ Insgesamt $O(n^{k+1})$

Greedy_k

Algorithmus GREEDY_K

1. Teste jede Teilmenge $\bar{S}$ mit $|\bar{S}| \leq k$
2. Fülle $\bar{S}$ mit Greedy_0 auf
3. Aktualisiere Lösung bei Bedarf

Laufzeit

→ $O(n^k)$ zu testende Teilmengen

→ $O(n)$ Zeit pro Teilmenge

→ $O(1)$

→ Insgesamt $O(n^{k+1})$

Satz: Greedy_k ist eine $\left(1 - \frac{1}{k+1}\right)$-Approximation

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30$

Achtung: In Übungsaufgaben unterscheiden sich oft die gegebene Reihenfolge und die, die jeweils für Greedy_0 gebraucht wird!

$\bar{S}$: Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$\sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge

$G + \sum_{i \in \bar{S}} p_i$: Fixierter Wert + Greedy auffüllen

G_k, S : Bisher bester Lösungswert und Menge

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

$\bar{S}$: Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$\sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge

$G + \sum_{i \in \bar{S}} p_i$: Fixierter Wert + Greedy auffüllen

G_k, S : Bisher bester Lösungswert und Menge

$\bar{S}$	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

$\bar{S}$: Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$\sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge

$G + \sum_{i \in \bar{S}} p_i$: Fixierter Wert + Greedy auffüllen

G_k, S : Bisher bester Lösungswert und Menge

$\bar{S}$	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
$\emptyset$	0	30	24	24	$\{1,2\}$

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

$\bar{S}$: Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$\sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge

$G + \sum_{i \in \bar{S}} p_i$: Fixierter Wert + Greedy auffüllen

G_k, S : Bisher bester Lösungswert und Menge

$\bar{S}$	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
$\emptyset$	0	30	24	24	$\{1,2\}$
$\{1\}$	13	17	24	24	$\{1,2\}$

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

$\bar{S}$: Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$\sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge

$G + \sum_{i \in \bar{S}} p_i$: Fixierter Wert + Greedy auffüllen

G_k, S : Bisher bester Lösungswert und Menge

$\bar{S}$	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
$\emptyset$	0	30	24	24	$\{1,2\}$
$\{1\}$	13	17	24	24	$\{1,2\}$
$\{2\}$	11	19	24	24	$\{1,2\}$

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

$\bar{S}$: Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$\sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge

$G + \sum_{i \in \bar{S}} p_i$: Fixierter Wert + Greedy auffüllen

G_k, S : Bisher bester Lösungswert und Menge

$\bar{S}$	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
$\emptyset$	0	30	24	24	$\{1,2\}$
$\{1\}$	13	17	24	24	$\{1,2\}$
$\{2\}$	11	19	24	24	$\{1,2\}$
$\{3\}$	10	20	23	24	$\{1,2\}$

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

$\bar{S}$: Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$\sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge

$G + \sum_{i \in \bar{S}} p_i$: Fixierter Wert + Greedy auffüllen

G_k, S : Bisher bester Lösungswert und Menge

$\bar{S}$	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
$\emptyset$	0	30	24	24	$\{1,2\}$
$\{1\}$	13	17	24	24	$\{1,2\}$
$\{2\}$	11	19	24	24	$\{1,2\}$
$\{3\}$	10	20	23	24	$\{1,2\}$
$\{4\}$	8	22	21	24	$\{1,2\}$

Greedy$_k$ – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

$\bar{S}$: Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$\sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge

$G + \sum_{i \in \bar{S}} p_i$: Fixierter Wert + Greedy auffüllen

G_k, S : Bisher bester Lösungswert und Menge

$\bar{S}$	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
$\emptyset$	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}
{3}	10	20	23	24	{1,2}
{4}	8	22	21	24	{1,2}
{1,2}	24	6	24	24	{1,2}

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

$\bar{S}$: Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$\sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge

$G + \sum_{i \in \bar{S}} p_i$: Fixierter Wert + Greedy auffüllen

G_k, S : Bisher bester Lösungswert und Menge

$\bar{S}$	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
$\emptyset$	0	30	24	24	$\{1,2\}$
$\{1\}$	13	17	24	24	$\{1,2\}$
$\{2\}$	11	19	24	24	$\{1,2\}$
$\{3\}$	10	20	23	24	$\{1,2\}$
$\{4\}$	8	22	21	24	$\{1,2\}$
$\{1,2\}$	24	6	24	24	$\{1,2\}$
$\{1,3\}$	23	7	23	24	$\{1,2\}$

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

$\bar{S}$: Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$\sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge

$G + \sum_{i \in \bar{S}} p_i$: Fixierter Wert + Greedy auffüllen

G_k, S : Bisher bester Lösungswert und Menge

$\bar{S}$	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
$\emptyset$	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}
{3}	10	20	23	24	{1,2}
{4}	8	22	21	24	{1,2}
{1,2}	24	6	24	24	{1,2}
{1,3}	23	7	23	24	{1,2}
{1,4}	21	9	21	24	{1,2}

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

$\bar{S}$: Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$\sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge

$G + \sum_{i \in \bar{S}} p_i$: Fixierter Wert + Greedy auffüllen

G_k, S : Bisher bester Lösungswert und Menge

$\bar{S}$	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
$\emptyset$	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}
{3}	10	20	23	24	{1,2}
{4}	8	22	21	24	{1,2}
{1,2}	24	6	24	24	{1,2}
{1,3}	23	7	23	24	{1,2}
{1,4}	21	9	21	24	{1,2}
{2,3}	21	9	29	29	{2,3,4}

Greedy$_k$ – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

$\bar{S}$: Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$\sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge

$G + \sum_{i \in \bar{S}} p_i$: Fixierter Wert + Greedy auffüllen

G_k, S : Bisher bester Lösungswert und Menge

$\bar{S}$	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
$\emptyset$	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}
{3}	10	20	23	24	{1,2}
{4}	8	22	21	24	{1,2}
{1,2}	24	6	24	24	{1,2}
{1,3}	23	7	23	24	{1,2}
{1,4}	21	9	21	24	{1,2}
{2,3}	21	9	29	29	{2,3,4}
{2,4}	19	11	29	29	{2,3,4}

Greedy$_k$ – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

$\bar{S}$: Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$\sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge

$G + \sum_{i \in \bar{S}} p_i$: Fixierter Wert + Greedy auffüllen

G_k, S : Bisher bester Lösungswert und Menge

$\bar{S}$	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
$\emptyset$	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}
{3}	10	20	23	24	{1,2}
{4}	8	22	21	24	{1,2}
{1,2}	24	6	24	24	{1,2}
{1,3}	23	7	23	24	{1,2}
{1,4}	21	9	21	24	{1,2}
{2,3}	21	9	29	29	{2,3,4}
{2,4}	19	11	29	29	{2,3,4}
{3,4}	18	12	29	29	{2,3,4}

Greedy$_k$ – Beispiel

Hausaufgabe 1 (Greedy$_k$): **(7 Punkte)**

In dieser Aufgabe betrachten wir den Algorithmus GREEDY$_k$ aus der Vorlesung. Wende GREEDY$_k$ auf die folgende Instanz an und gib die beste gefundene Lösung an.

i	1	2	3	4
z_i	7	14	14	13
p_i	7	12	13	14

mit $Z = 27$ und $k = 2$

Gib dazu für jede betrachtete fixierte Teilmenge $\overline{S}$ der Objekte die folgenden Mengen bzw. Werte in einer Tabelle an.

- $\overline{S}$: Menge fixierter Objekte
- $\sum_{i \in \overline{S}} z_i$: Gewicht der fixierten Objekte
- $Z - \sum_{i \in \overline{S}} z_i$: Restkapazität
- $G + \sum_{i \in \overline{S}} p_i$: Wert der fixierten Objekte plus Greedy auf nicht fixierten Objekten.
  Falls die in $\overline{S}$ ausgewählten Elemente die Restkapazität überschreiten, trage hier jeweils anstelle des Wertes ein $\times$ ein.
- G_k: Wert der bisher besten gefundenen Lösung
- S: Lösungsmenge der bisher besten Lösung

Betrachte (analog zum Beispiel aus der großen Übung) fixierte Mengen $\overline{S}$ mit weniger Elementen *vor* Mengen mit mehr Elementen. Für zwei fixierte Mengen der gleichen Größe M_1, M_2 betrachte M_1 vor M_2, falls das kleinste Element $x \in M_1 \setminus M_2$ kleiner ist als das kleinste Element $y \in M_2 \setminus M_1$ (lexikografische Sortierung).

(Hinweis: Die Menge $X \setminus Y$ enthält Elemente aus X, die nicht in Y vorkommen. Wir betrachten also $M_1 = \{1,2\}$ vor $M_2 = \{1,3\}$, weil das kleinste Element von $M_1 \setminus M_2 = \{2\}$ kleiner ist als das kleinste Element von $M_2 \setminus M_1 = \{3\}$.)

$\overline{S}$	$\sum_{i \in \overline{S}} z_i$	$Z - \sum_{i \in \overline{S}} z_i$	$G + \sum_{i \in \overline{S}} p_i$	G_k	S
$\emptyset$	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}
{3}	10	20	23	24	{1,2}
{4}	8	22	21	24	{1,2}
{1,2}	24	6	24	24	{1,2}
{1,3}	23	7	23	24	{1,2}
{1,4}	21	9	21	24	{1,2}
{2,3}	21	9	29	29	{2,3,4}
{2,4}	19	11	29	29	{2,3,4}
{3,4}	18	12	29	29	{2,3,4}

Greedy_k – Beispiel

Das ist genau diese Reihenfolge hier :)

Betrachte (analog zum Beispiel aus der großen Übung) fixierte Mengen $\overline{S}$ mit weniger Elementen *vor* Mengen mit mehr Elementen. Für zwei fixierte Mengen der gleichen Größe M_1, M_2 betrachte M_1 vor M_2, falls das kleinste Element $x \in M_1 \setminus M_2$ kleiner ist als das kleinste Element $y \in M_2 \setminus M_1$ (lexikografische Sortierung).

(Hinweis: Die Menge $X \setminus Y$ enthält Elemente aus X, die nicht in Y vorkommen. Wir betrachten also $M_1 = \{1,2\}$ vor $M_2 = \{1,3\}$, weil das kleinste Element von $M_1 \setminus M_2 = \{2\}$ kleiner ist als das kleinste Element von $M_2 \setminus M_1 = \{3\}$.)

$\overline{S}$	$\sum_{i \in \overline{S}} z_i$	$Z - \sum_{i \in \overline{S}} z_i$	$G + \sum_{i \in \overline{S}} p_i$	
$\emptyset$	0	30	24	
$\{1\}$	13	17	24	
$\{2\}$	11	19	24	
$\{3\}$	10	20	23	
$\{4\}$	8	22	21	
$\{1,2\}$	24	6	24	
$\{1,3\}$	23	7	23	
$\{1,4\}$	21	9	21	
$\{2,3\}$	21	9	29	
$\{2,4\}$	19	11	29	
$\{3,4\}$	18	12	29	

Fragen?

Greedy_k

Satz: Greedy_k ist eine $\left(1 - \frac{1}{k+1}\right)$-Approximation

Greedy_k

Satz: Der Approximationsfaktor $\left(1 - \frac{1}{k+1}\right)$ für Greedy_k ist scharf.

Beweis

Mit einem $N \in \mathbb{N}$ konstruieren wir die folgende Instanz:

i	1	2	...	$k+2$
z_i	1	N	...	N
p_i	2	N	...	N

$$Z = N \cdot (k+1)$$

Lösung von Greedy_k

1. Falls $1 \in \bar{S}$: Es kommen k Objekte mit $p_i = z_i = N$ hinzu. ⇒ Wert $kN + 2$
2. Falls $1 \notin \bar{S}$:
   1. Es sind $\leq k$ Objekte mit $p_i = z_i = N$ fixiert.
   2. Greedy_0 packt Objekt 1 hinzu und füllt mit anderen auf. ⇒ Wert $kN + 2$

Greedy_k liefert Wert $kN + 2$

Greedy$_k$

Satz: Der Approximationsfaktor $\left(1 - \frac{1}{k+1}\right)$ für Greedy$_k$ ist scharf.

Beweis

Mit einem $N \in \mathbb{N}$ konstruieren wir die folgende Instanz:

i	1	2	...	$k+2$
z_i	1	N	...	N
p_i	2	N	...	N

$$Z = N \cdot (k+1)$$

Optimale Lösung

Nimm Objekte 2 bis $k+2$ auf. ⇒ Wert $N \cdot (k+1)$

Greedy$_k$ liefert Wert $kN + 2$

Optimum hat Wert $N \cdot (k+1)$

Greedy_k

Satz: Der Approximationsfaktor $\left(1 - \frac{1}{k+1}\right)$ für Greedy_k ist scharf.

Beweis

Mit einem $N \in \mathbb{N}$ konstruieren wir die folgende Instanz:

i	1	2	...	$k+2$
z_i	1	N	...	N
p_i	2	N	...	N

$$Z = N \cdot (k+1)$$

Greedy_k liefert Wert $kN + 2$

Optimum hat Wert $N \cdot (k+1)$

Damit ist:

$$\frac{ALG}{OPT} = \frac{kN+2}{N \cdot (k+1)} = \frac{kN}{N \cdot (k+1)} + \frac{2}{N \cdot (k+1)} = 1 - \frac{1}{k+1} + \underbrace{\frac{2}{N \cdot (k+1)}}_{\to 0 \text{ für } N \to \infty}$$

$$\frac{k}{k+1} = \frac{k+1-1}{k+1}$$

Fragen?

Vertex Cover

& Matchings 

Vertex Cover

Also eigentlich ist der Name voll irritierend weil eigentlich werden ja alle Edges gecoverd und nicht die Vertices…

set of vertices that includes at least one

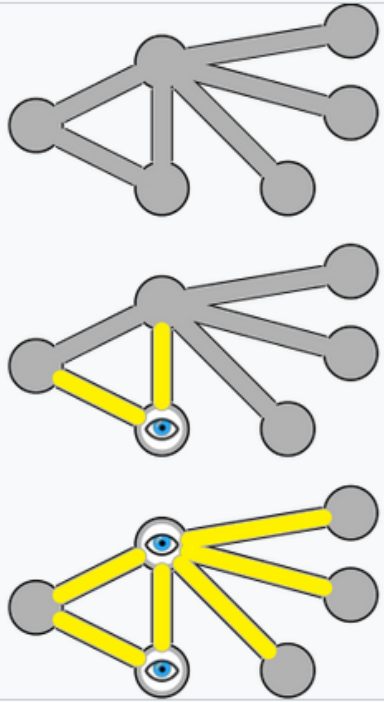
a classical optimization problem. It is
Moreover, it is hard to approximate – it
conjecture is true, the other hand, it
NP-hard optimiz lem that has
em, was one o NP-complete
nal complexi rmore, the
n paramete ory.

linear pr

ergraphs.

set of V such that

ore every edge has at least



Example graph that has a vertex cover comprising 2 vertices (bottom), but none with fewer.

Vertex Cover

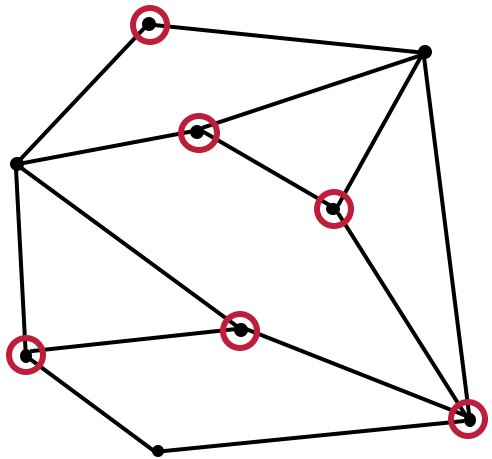
Gegeben

- Graph $G = (V, E)$
- Zahl $k \in \mathbb{N}$

Frage

- Existiert eine Menge $VC \subseteq V$ mit $|VC| \leq k$, sodass für jede Kante $\{u, v\} \in E$ gilt:
  $u \in VC$ oder $v \in VC$?

Das ist unser Vertex-Cover



$|VC| = 6$

Als Optimierungsproblem: Suche die kleinste Zahl k.

(Kleinste) Menge an Knoten, die jede Kante abdeckt!

Vertex Cover

Geht das noch besser?

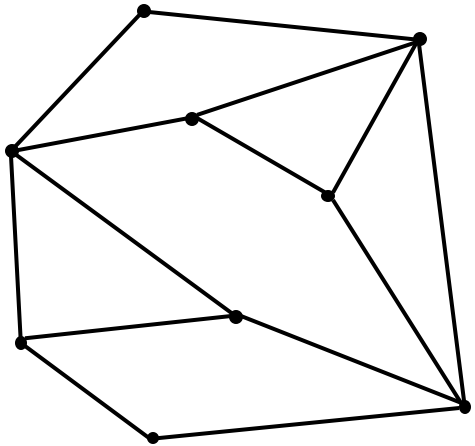
Gegeben

- Graph $G = (V, E)$
- Zahl $k \in \mathbb{N}$

Das ist unser Vertex-Cover

Frage

- Existiert eine Menge $VC \subseteq V$ mit $|VC| \leq k$, sodass für jede Kante $\{u, v\} \in E$ gilt:
  $u \in VC$ oder $v \in VC$?



$|VC| = 6$

Als Optimierungsproblem: Suche die kleinste Zahl k.

(Kleinste) Menge an Knoten, die jede Kante abdeckt!

Vertex Cover

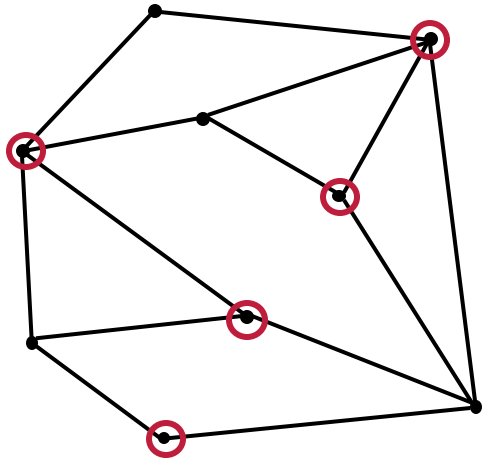
Gegeben

- Graph $G = (V, E)$
- Zahl $k \in \mathbb{N}$

Frage

- Existiert eine Menge $VC \subseteq V$ mit $|VC| \leq k$, sodass für jede Kante $\{u, v\} \in E$ gilt:
  $u \in VC$ oder $v \in VC$?

Das ist unser Vertex-Cover

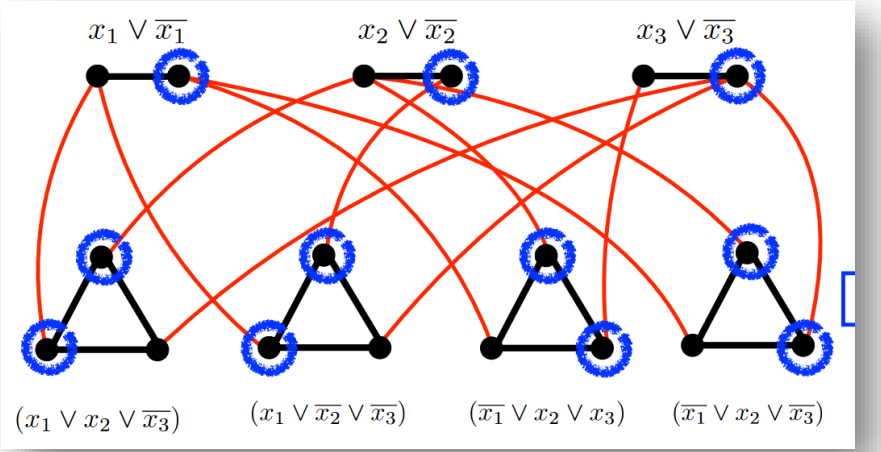


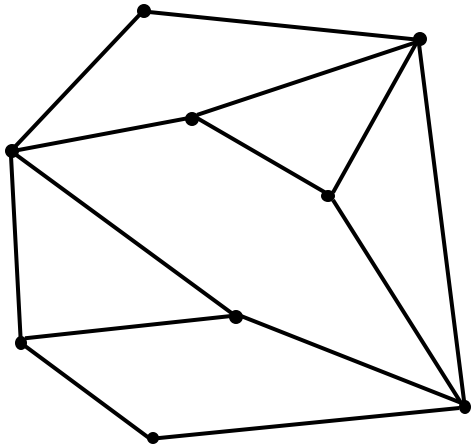
$|VC| = 5$

Als Optimierungsproblem: Suche die kleinste Zahl k.

(Kleinste) Menge an Knoten, die jede Kante abdeckt!

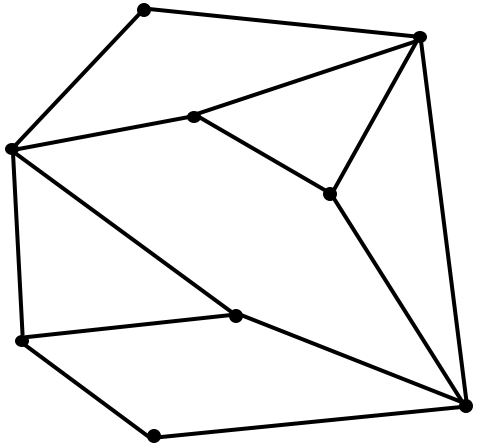
Vertex Cover



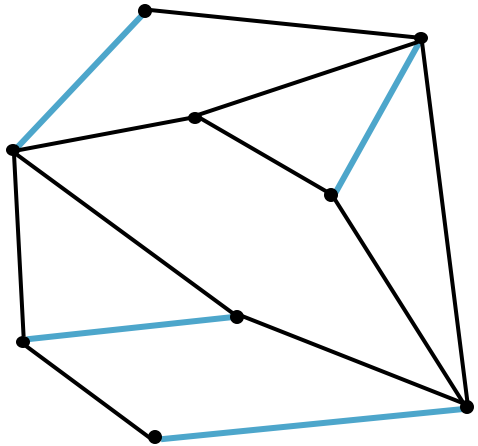


- Vorlesung: Ein minimales Vertex-Cover zu finden ist NP-schwer!
- Unsere Mission also: Wir approximieren das Ding!
- Dafür brauchen wir zuerst … Schranken!

Schranken

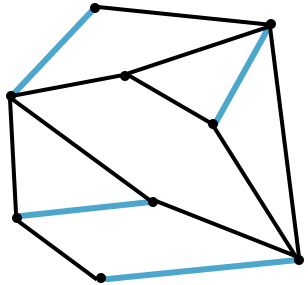


Schranken



Matching

Gegeben	**Gesucht**
Graph $G = (V, E)$	Eine Menge von Kanten $M \subseteq E$, die sich paarweise keinen Knoten teilen.



Ein Matching M ist *inklusionsmaximal*, wenn es keine Kante $e \notin M$ gibt, sodass $M \cup \{e\}$ ein Matching ist.

Das findet man easy!

Engl.:
maximal matching

Ein Matching M ist *kardinalitätsmaximal*, wenn es kein Matching M' gibt, sodass $|M| < |M'|$ gilt.

Hui …*

Engl.:
maximum matching

*Mehr dazu: Netzwerkalgorithmen.
Nicht NP-schwer. Aber auch nicht ganz trivial…

Matching

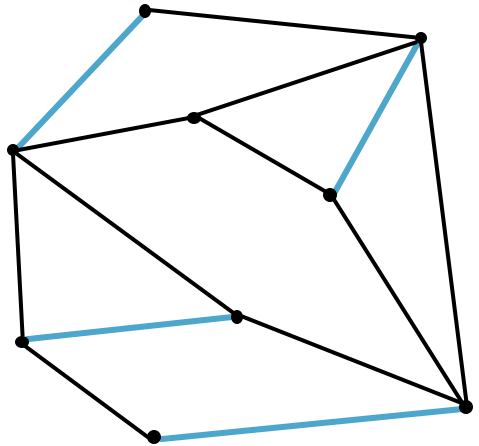
Beobachtungen

Die blauen Kanten bilden ein Matching!

… ein *kardinalitätsmaximales* Matching?
Hmm, das sehen wir nicht so einfach.

… zumindest aber ein
inklusionsmaximales Matching!

(Man kann keine Kante mehr hinzufügen!)



Matching

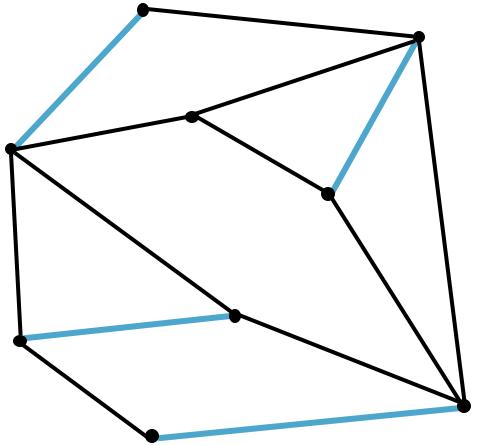
Eigentlich suchen wir doch ein minimales Vertex Cover!
Also: Eine minimale Knotenmenge VC, die jede Kante im Graph abdeckt

Zwei zentrale Beobachtungen

Für jede Kante im Matching benötigen wir mindestens einen Knoten in VC!

Fügen wir jeweils *beide* Knoten zu VC hinzu, erhalten wir ein Vertex Cover!
(bei inklusionsmaximalen Matchings)

Inklusionsmaximales Matching



Matching

Für jede Kante im Matching benötigen wir mindestens einen Knoten in VC!

Untere Schranke für Min. Vertex Cover!

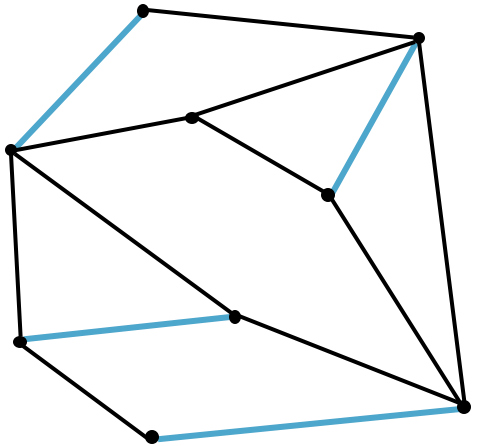
Fügen wir jeweils *beide* Knoten zu VC hinzu, erhalten wir ein Vertex Cover!
(bei inklusionsmaximalen Matchings)

Obere Schranke für Min. Vertex Cover!

Damit haben wir eine 2-Approximation für Vertex Cover gefunden!

… wir schreiben das mal sauber auf.

Inklusionsmaximales Matching



Vertex Cover und Matchings

Satz: Sei VC_{opt} ein kleinstes Vertex Cover und M ein inklusionsmaximales Matching. Dann gilt:

$$|M| \leq |VC_{opt}| \leq 2|M|$$

Für jede Kante $\{u, v\} \in M$ muss u oder v in VC_{opt} enthalten sein.

$$\Rightarrow |M| \leq \left|VC_{opt}\right|$$

Betrachte nun $VC' = \bigcup_{\{u,v\} \in M} \{u, v\}$.

Annahme: VC' ist kein Vertex Cover

Dann existiert eine Kante $\{u', v'\}$ mit $u', v' \notin VC'$.

Damit kann M mit $\{u', v'\}$ erweitert werden. Widerspruch, dass M inklusionsmaximal ist.

$$\Rightarrow 2|M| \geq |VC'| \geq \left|VC_{opt}\right|$$

Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

$$|M| \leq |VC_{opt}| \leq 2|M| \leq 2|VC_{opt}|$$

Aus welchen Schritten besteht diese Approximation?

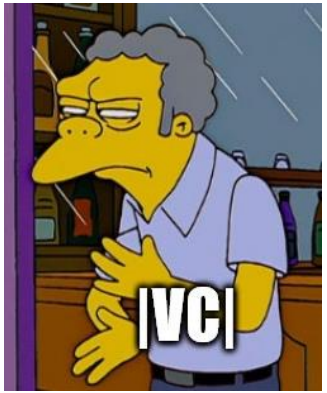
Wir haben ja gerade gesehen, wie man dieses Vertex Cover findet.

- Finde für den gegebenen Graph ein inklusionsmaximales Matching M
- Wähle beide Knoten in allen Kanten aus M, um alle Kanten zu covern

Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

Beweis (grafisch)

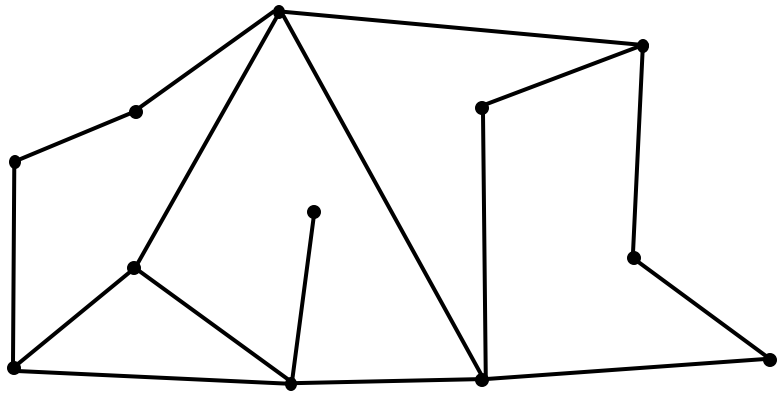


Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

Beide Schritte zusammen:

$VC := \emptyset$
for each $\{u, v\} \in E$ **do**
  if $u \notin VC$ und $v \notin VC$ **then**
    $VC := VC \cup \{u, v\}$
return VC

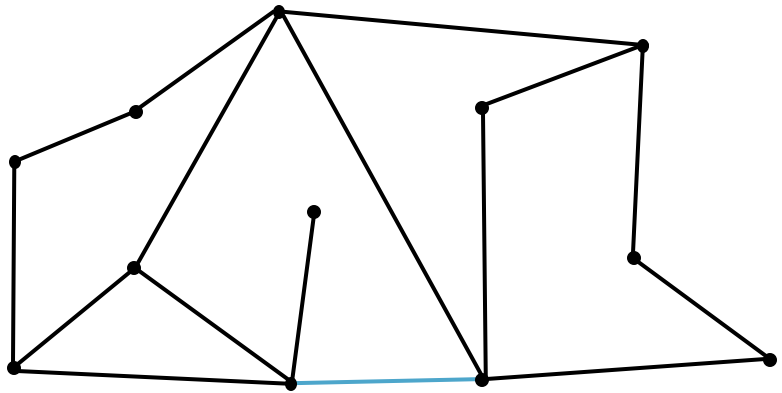


Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

Beide Schritte zusammen:

$VC \coloneqq \emptyset$
for each $\{u, v\} \in E$ **do**
  if $u \notin VC$ und $v \notin VC$ **then**
    $VC \coloneqq VC \cup \{u, v\}$
return VC

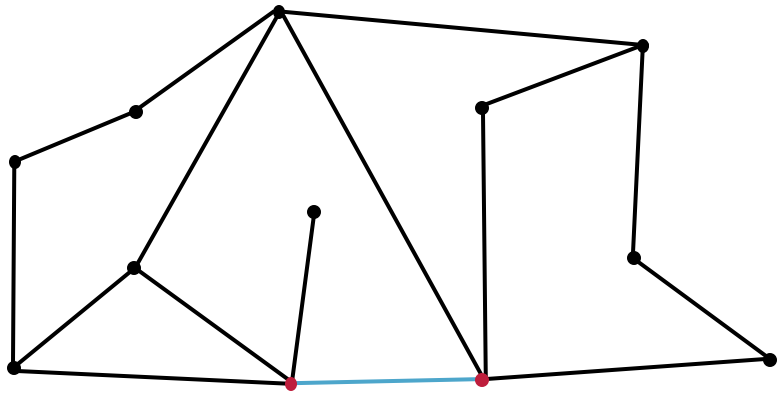


Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

Beide Schritte zusammen:

$VC \coloneqq \emptyset$
for each $\{u, v\} \in E$ **do**
  if $u \notin VC$ und $v \notin VC$ **then**
    $VC \coloneqq VC \cup \{u, v\}$
return VC

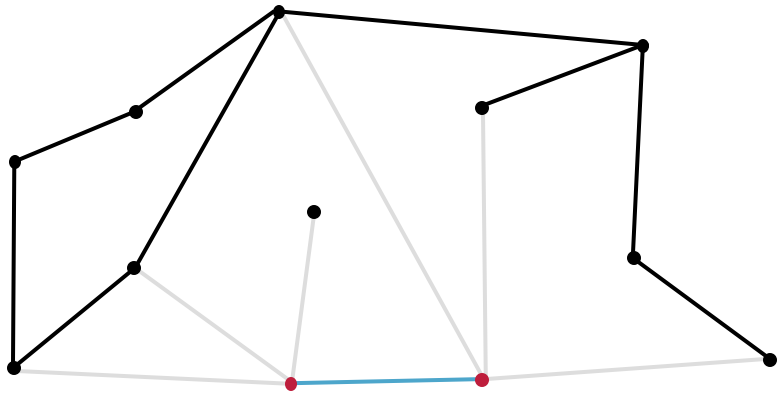


Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

Beide Schritte zusammen:

$VC \coloneqq \emptyset$
for each $\{u, v\} \in E$ **do**
  if $u \notin VC$ und $v \notin VC$ **then**
    $VC \coloneqq VC \cup \{u, v\}$
return VC

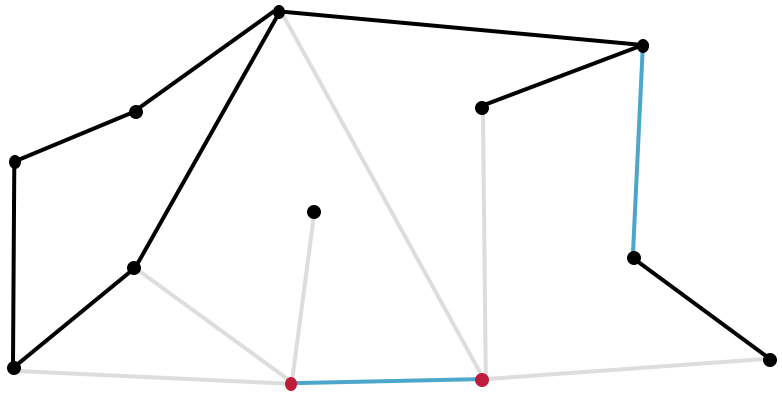


Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

Beide Schritte zusammen:

$VC \coloneqq \emptyset$
for each $\{u, v\} \in E$ **do**
  if $u \notin VC$ und $v \notin VC$ **then**
    $VC \coloneqq VC \cup \{u, v\}$
return VC

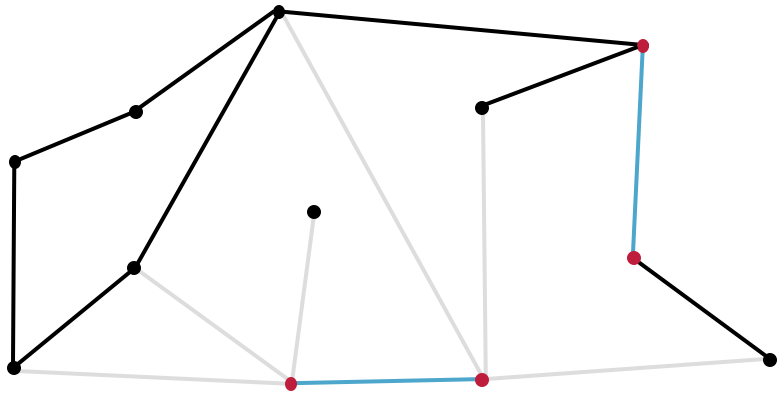


Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

Beide Schritte zusammen:

$VC \coloneqq \emptyset$
for each $\{u, v\} \in E$ **do**
  if $u \notin VC$ und $v \notin VC$ **then**
    $VC \coloneqq VC \cup \{u, v\}$
return VC

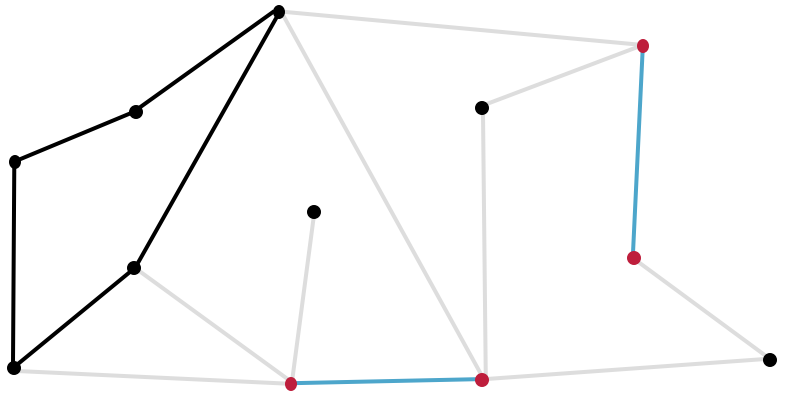


Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

Beide Schritte zusammen:

$VC \coloneqq \emptyset$
for each $\{u, v\} \in E$ **do**
  if $u \notin VC$ und $v \notin VC$ **then**
    $VC \coloneqq VC \cup \{u, v\}$
return VC

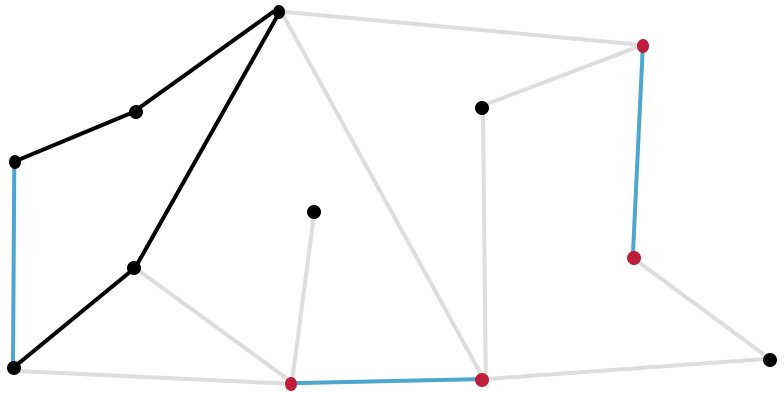


Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

Beide Schritte zusammen:

$VC \coloneqq \emptyset$
for each $\{u, v\} \in E$ **do**
  if $u \notin VC$ und $v \notin VC$ **then**
    $VC \coloneqq VC \cup \{u, v\}$
return VC

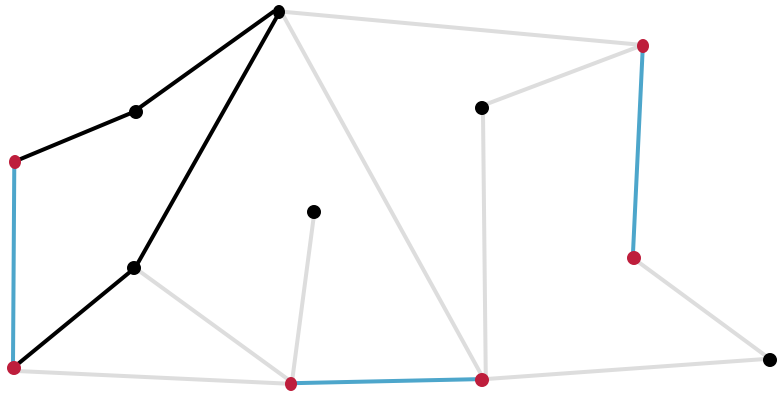


Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

Beide Schritte zusammen:

$VC \coloneqq \emptyset$
for each $\{u, v\} \in E$ **do**
  if $u \notin VC$ und $v \notin VC$ **then**
    $VC \coloneqq VC \cup \{u, v\}$
return VC

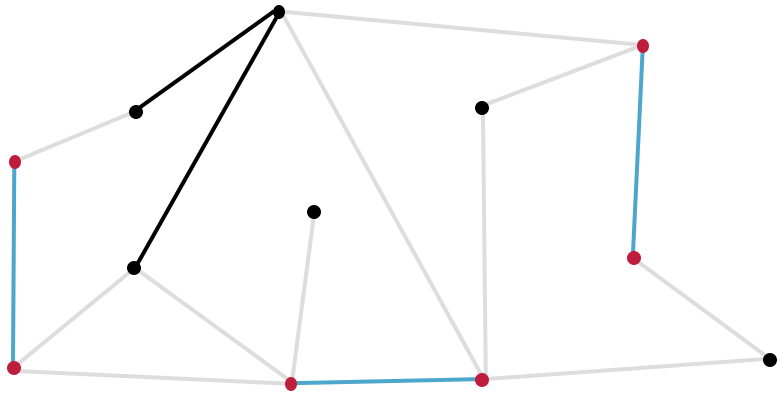


Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

Beide Schritte zusammen:

$VC \coloneqq \emptyset$
for each $\{u, v\} \in E$ **do**
  if $u \notin VC$ und $v \notin VC$ **then**
    $VC \coloneqq VC \cup \{u, v\}$
return VC

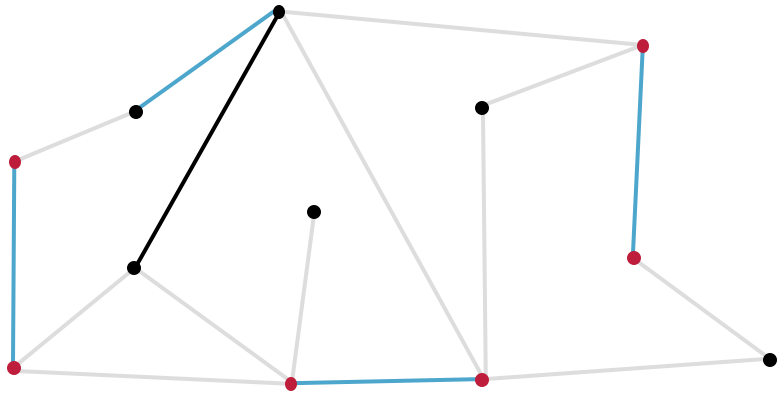


Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

Beide Schritte zusammen:

$VC \coloneqq \emptyset$
for each $\{u, v\} \in E$ **do**
  if $u \notin VC$ und $v \notin VC$ **then**
    $VC \coloneqq VC \cup \{u, v\}$
return VC

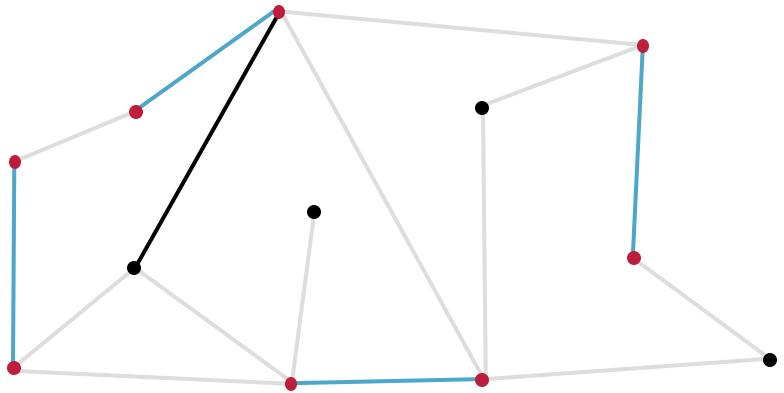


Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

Beide Schritte zusammen:

$VC \coloneqq \emptyset$
for each $\{u, v\} \in E$ **do**
  if $u \notin VC$ und $v \notin VC$ **then**
    $VC \coloneqq VC \cup \{u, v\}$
return VC

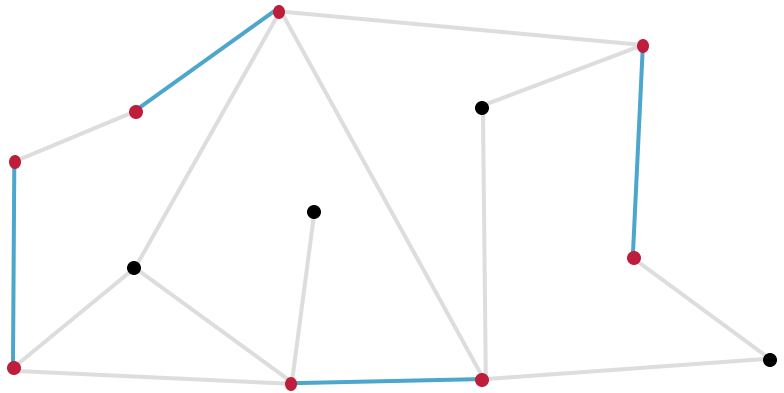


Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

Beide Schritte zusammen:

$VC \coloneqq \emptyset$
for each $\{u, v\} \in E$ **do**
  if $u \notin VC$ und $v \notin VC$ **then**
    $VC \coloneqq VC \cup \{u, v\}$
return VC



Fragen?

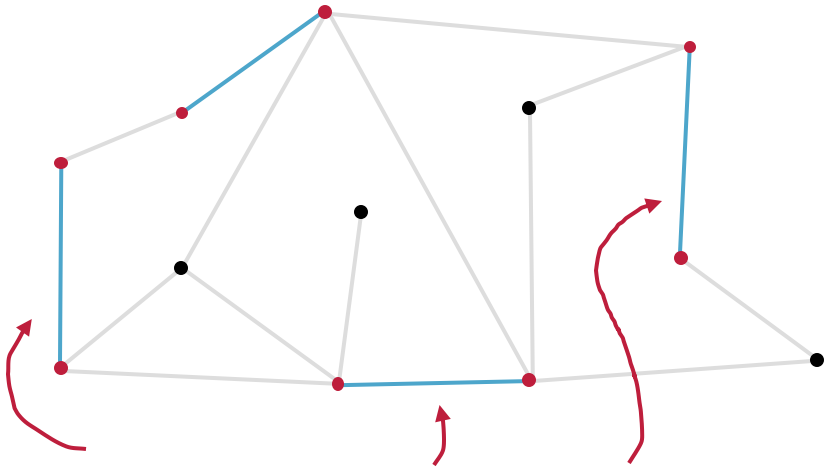
Vertex Cover und Matchings

In diesem Graphen gibt es ein inklusionsmaximales Matching der Größe 4.

Also gilt für ein minimales Vertex-Cover VC:

$$4 \leq |VC| \leq 8$$

Basierend hierauf:
Wie können wir andere (und vielleicht bessere) untere Schranken finden?



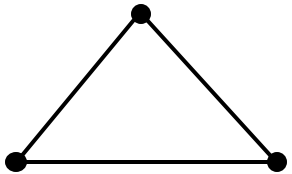
Vielleicht können wir den Graphen statt in einzelne unabhängige Kanten in größere Teilgraphen unterteilen und die jeweils lösen …

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

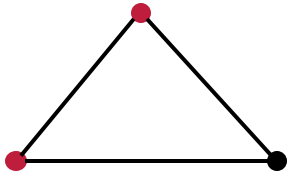


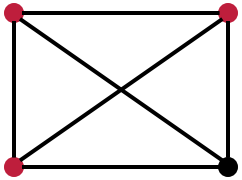
Optimale Vertex Cover

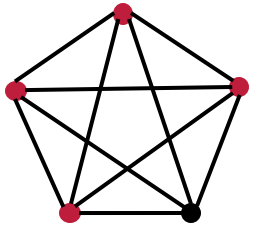
Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen







$$|VC| = n - 1$$

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

$|VC| = n - 1$

Pfad Graphen

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

$|VC| = n - 1$

Pfad Graphen

$|VC| = \left\lfloor \frac{n}{2} \right\rfloor$

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

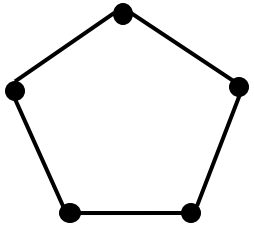
Vollständige Graphen

$|VC| = n - 1$

Pfad Graphen

$|VC| = \left\lfloor \frac{n}{2} \right\rfloor$

Kreis Graphen



Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

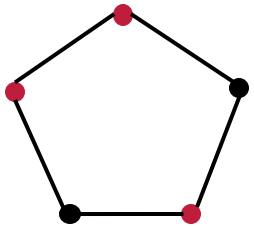
Vollständige Graphen

$$|VC| = n - 1$$

Pfad Graphen

$$|VC| = \left\lfloor \frac{n}{2} \right\rfloor$$

Kreis Graphen



$$|VC| = \left\lceil \frac{n}{2} \right\rceil$$

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

$|VC| = n - 1$

Pfad Graphen

$|VC| = \left\lfloor \frac{n}{2} \right\rfloor$

Kreis Graphen

$|VC| = \left\lceil \frac{n}{2} \right\rceil$

Bäume

$O(n)$ Algorithmus

Intervall-Graphen

$O(n \log n)$ Algorithmus

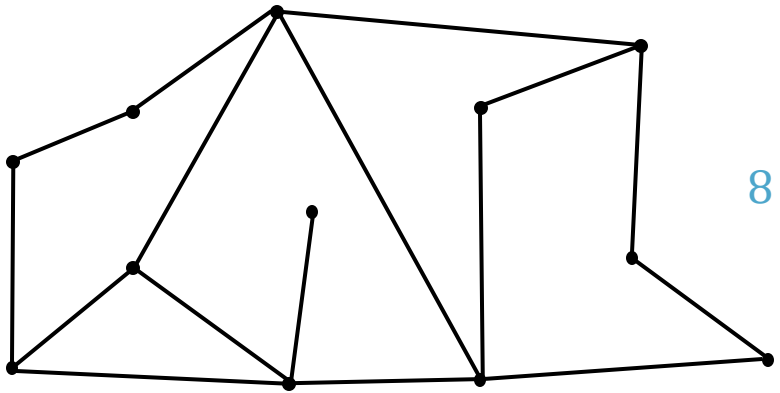
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

$8 \geq |VC| \geq 2 + 1 + 3 = 6$

Wie groß ist ein kleinstes Vertex Cover? 6, 7 oder 8?

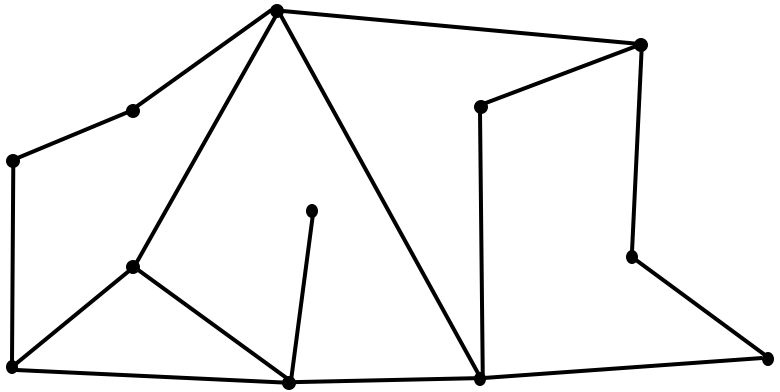
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$$8 \geq |VC| \geq 1 + 3 + 3 = 7$$

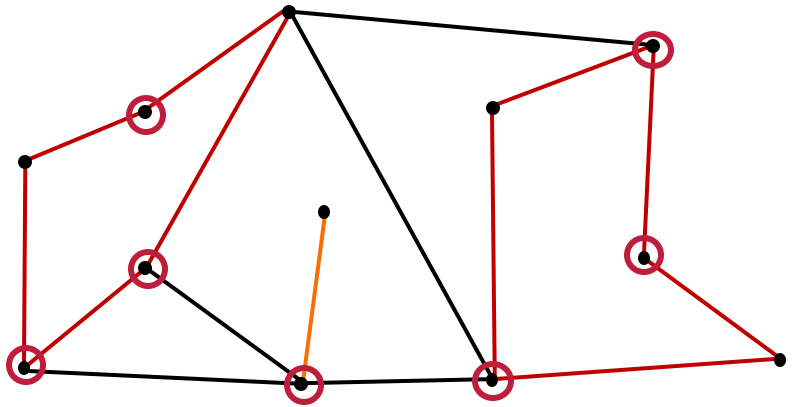
Wie groß ist ein kleinstes Vertex Cover? ~~6~~, 7 oder 8?

Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

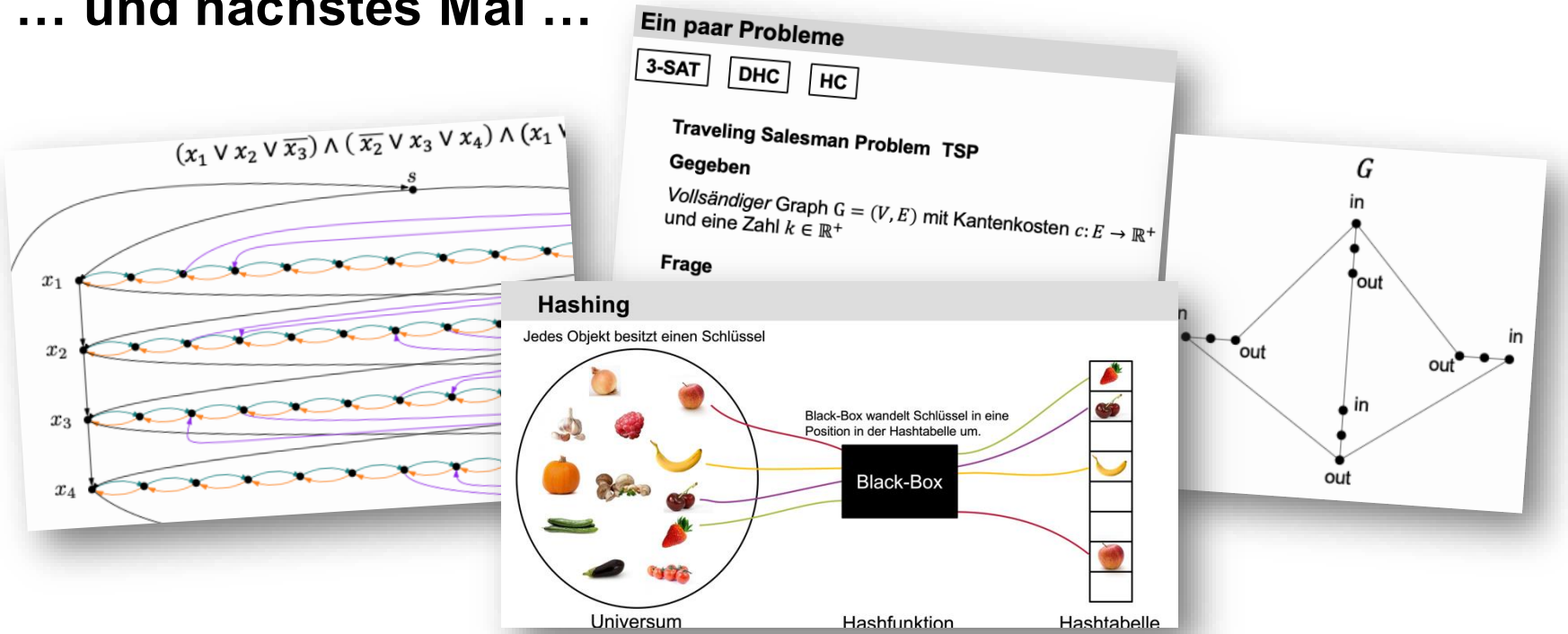


$$7 \geq |VC| \geq 1 + 3 + 3 = 7$$

Wie groß ist ein kleinstes Vertex Cover? ~~6~~, 7 oder ~~8~~?

Fragen?

… und nächstes Mal …



… gibt es Komplexität, Reduktionen (und Hashing)? We'll see!

am 02. Juli
(in zwei Wochen)