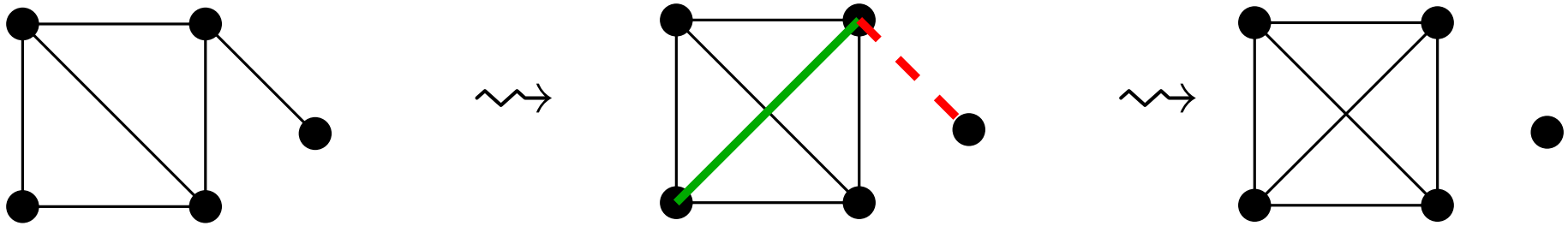
Karlsruher Institut für Technologie

Cluster Editing

Proseminar · 07. Mai 2018

Hamid Doust

INSTITUT FÜR THEORETISCHE INFORMATIK · LEHRSTUHL ALGORITHMIK



Cluster Editing

Gegeben sei ein Graph $G = (V, E)$ und $k \in \mathbb{N}$. Ist es möglich mithilfe von k Editieroperationen G in eine disjunkte Vereinigung von Cliquen zu überführen ?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Cluster Editing

Gegeben sei ein Graph $G = (V, E)$ und $k \in \mathbb{N}$. Ist es möglich mithilfe von k Editieroperationen G in eine disjunkte Vereinigung von Cliquen zu überführen ?

$\rightsquigarrow$ Editieroperation : Kante löschen/einfügen

$\rightsquigarrow$ disjunkte Vereinigung : jeder Knoten gehört genau zur einer Clique

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

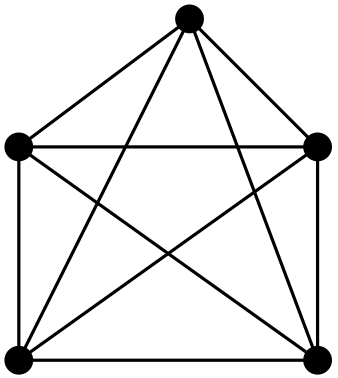
Cluster Editing

Gegeben sei ein Graph G = (V, E) und k $\in \mathbb{N}$. Ist es möglich mithilfe von k Editieroperationen G in eine disjunkte Vereinigung von Cliquen zu überführen ?

$\rightsquigarrow$ Editieroperation : Kante löschen/einfügen

$\rightsquigarrow$ disjunkte Vereinigung : jeder Knoten gehört genau zur einer Clique

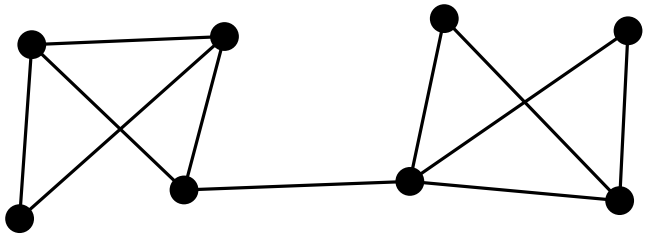
$\rightsquigarrow$ Clique ?



- NP-vollständig

Beispiel

Gegeben sei folgender Graph und k = 4

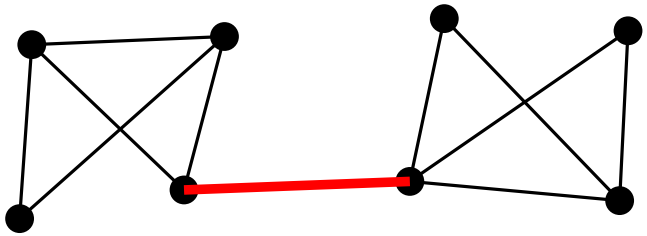


Gibt es eine Lösung für diese Instanz ?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel

Gegeben sei folgender Graph und k = 4

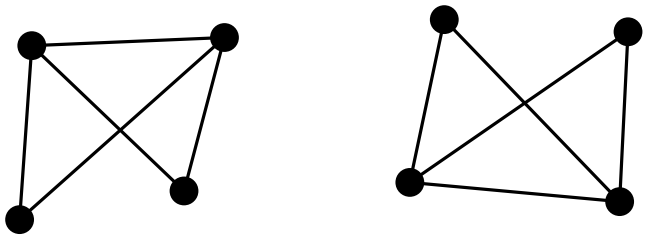


Gibt es eine Lösung für diese Instanz ?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel

Gegeben sei folgender Graph und k = 4

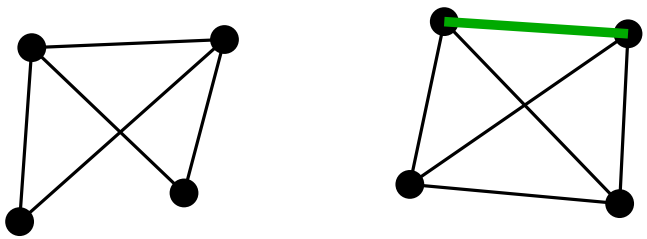


Gibt es eine Lösung für diese Instanz ?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel

Gegeben sei folgender Graph und k = 4

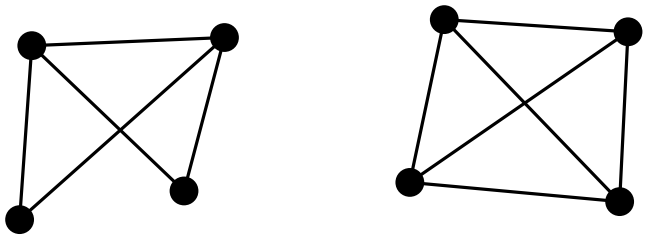


Gibt es eine Lösung für diese Instanz ?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel

Gegeben sei folgender Graph und k = 4

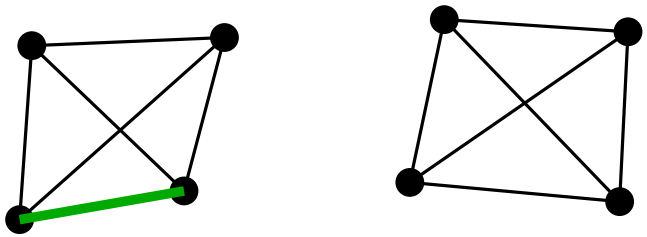


Gibt es eine Lösung für diese Instanz ?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel

Gegeben sei folgender Graph und k = 4

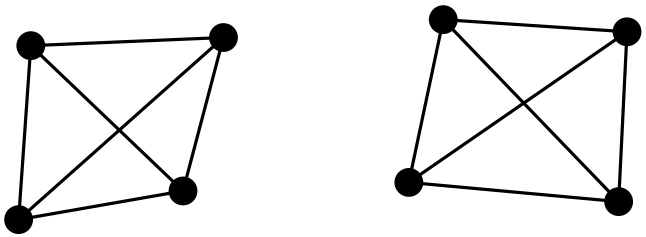


Gibt es eine Lösung für diese Instanz ?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel

Gegeben sei folgender Graph und k = 4

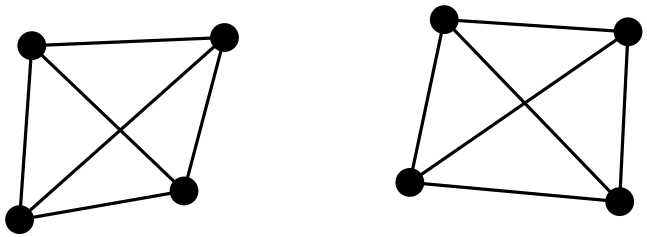


Gibt es eine Lösung für diese Instanz ?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel

Gegeben sei folgender Graph und k = 4



Gibt es eine Lösung für diese Instanz ?

$\rightsquigarrow$ 3 Editieroperation benutzt, d.h. Lösung existiert für k = 4

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kernelization

Sei L ein parametrisiertes Problem der Form (I, k). Dann heißt die reduzierte Instanz (I', k') Problem Kernel falls gilt:

- $|I'| \leq g(k)$ wird Größe des Kernels genannt
- $|k'| \leq f(k)$
- f und g sind berechenbar
- $(I, k) \in L \Leftrightarrow (I', k') \in L$
- Reduktion erfolgt in polynomieller Zeit

Kernelization

Sei L ein parametrisiertes Problem der Form (I, k). Dann heißt die reduzierte Instanz (I', k') Problem Kernel falls gilt:

- $|I'| \leq g(k)$ wird Größe des Kernels genannt
- $|k'| \leq f(k)$
- f und g sind berechenbar
- $(I, k) \in L \Leftrightarrow (I', k') \in L$
- Reduktion erfolgt in polynomieller Zeit

Cluster Editing:

- drei Regeln für Reduktion
- $g \in O(k^c)$, c sei Konstante
- $k' \leq k$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kernelization - Regeln

Regel 1

Für jedes Knotenpaar $u, v \in V$:

1. Falls u und v mehr als k gemeinsame Nachbarn haben, dann muss $\{u, v\}$ in E hinzugefügt werden (falls es nicht schon ist) und als permanent markiert werden.

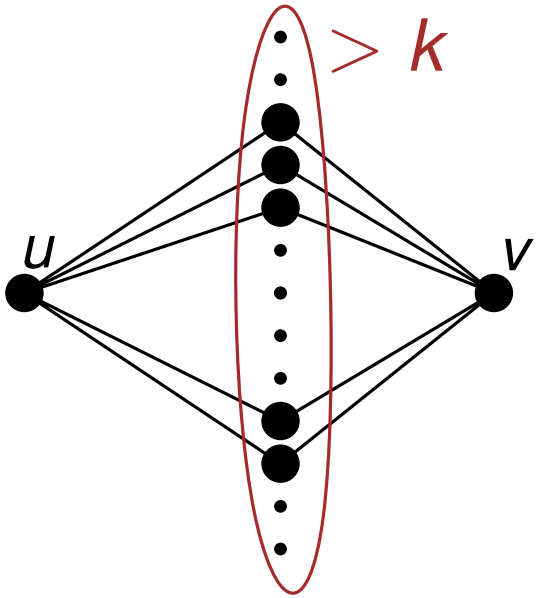
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kernelization - Regeln

Regel 1

Für jedes Knotenpaar $u, v \in V$:

1. Falls u und v mehr als k gemeinsame Nachbarn haben, dann muss $\{u, v\}$ in E hinzugefügt werden (falls es nicht schon ist) und als permanent markiert werden.

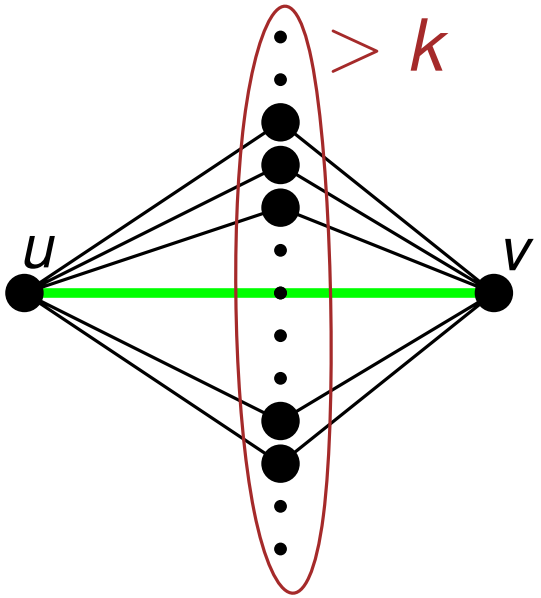


Kernelization - Regeln

Regel 1

Für jedes Knotenpaar $u, v \in V$:

1. Falls u und v mehr als k gemeinsame Nachbarn haben, dann muss $\{u, v\}$ in E hinzugefügt werden (falls es nicht schon ist) und als permanent markiert werden.



Kernelization - Regeln

Regel 1

Für jedes Knotenpaar $u, v \in V$:

2. Falls u und v mehr als k nicht-gemeinsame Nachbarn haben, dann muss $\{u,v\}$ von E gelöscht werden (falls es enthalten ist) und als verboten markiert werden.

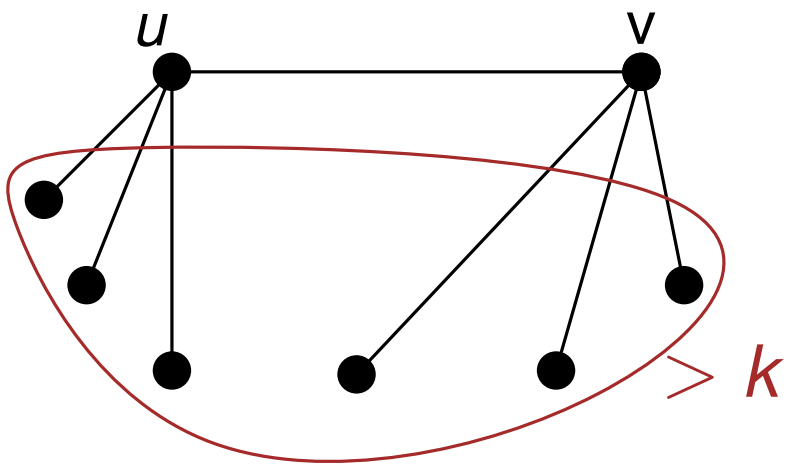
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kernelization - Regeln

Regel 1

Für jedes Knotenpaar $u, v \in V$:

2. Falls u und v mehr als k nicht-gemeinsame Nachbarn haben, dann muss $\{u,v\}$ von E gelöscht werden (falls es enthalten ist) und als verboten markiert werden.

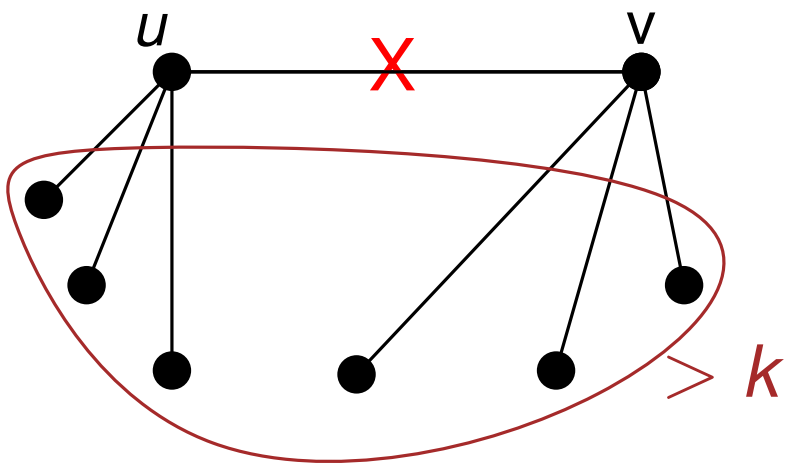


Kernelization - Regeln

Regel 1

Für jedes Knotenpaar $u, v \in V$:

2. Falls u und v mehr als k nicht-gemeinsame Nachbarn haben, dann muss $\{u,v\}$ von E gelöscht werden (falls es enthalten ist) und als verboten markiert werden.



Kernelization - Regeln

Regel 1

Für jedes Knotenpaar $u, v \in V$:

3. Falls u und v mehr als k gemeinsame Nachbarn haben und mehr als k nicht-gemeinsame Nachbarn haben, dann existiert keine Lösung für diese Instanz.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kernelization - Regeln

Regel 2

Für jedes Knotenpaar $u, v \in V$:

1. Falls $\{u, v\}$ und $\{u, w\}$ als permanent markiert worden sind, dann muss $\{v, w\}$ als Kante hinzugefügt werden (falls nicht schon vorhanden) und als permanent markiert werden.

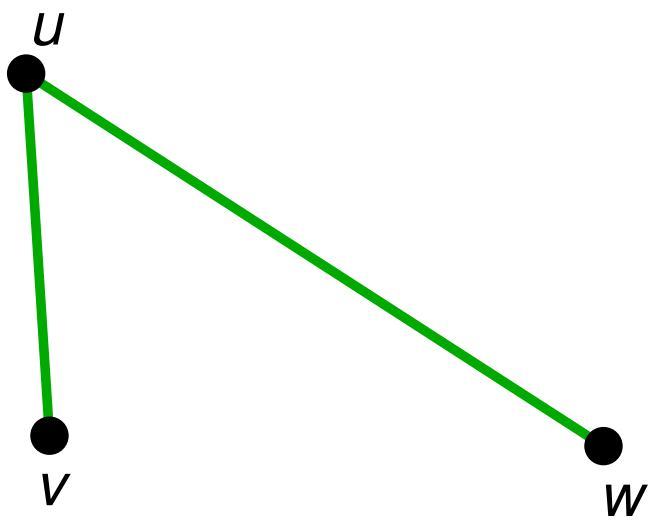
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kernelization - Regeln

Regel 2

Für jedes Knotenpaar $u, v \in V$:

1. Falls $\{u, v\}$ und $\{u, w\}$ als permanent markiert worden sind, dann muss $\{v, w\}$ als Kante hinzugefügt werden (falls nicht schon vorhanden) und als permanent markiert werden.

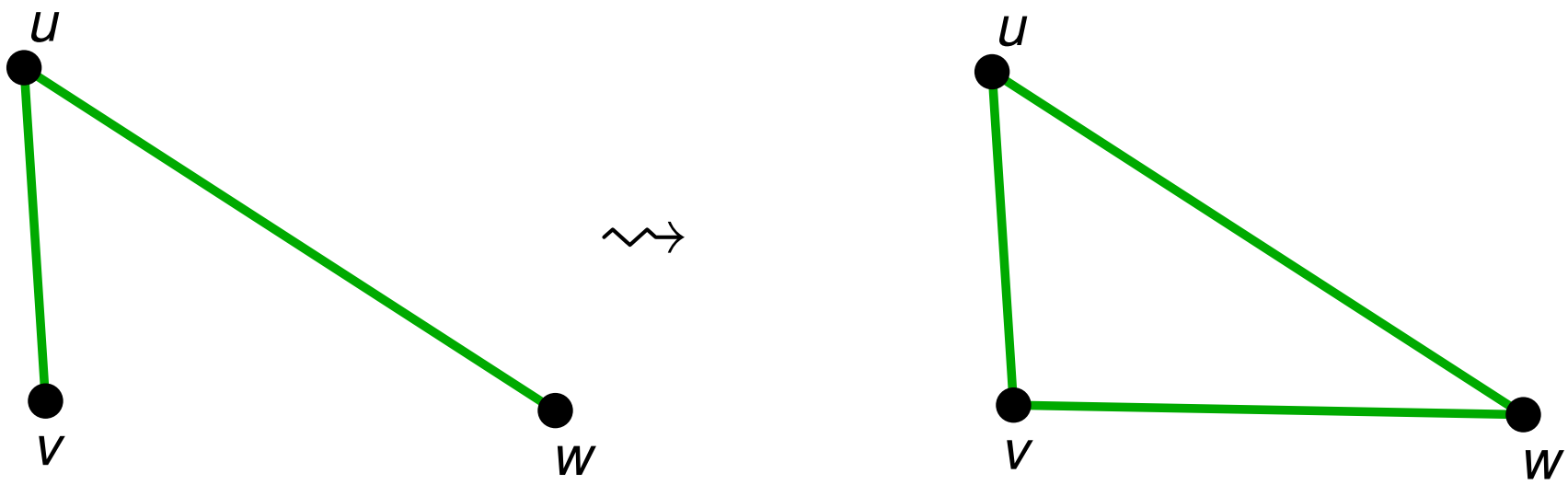


Kernelization - Regeln

Regel 2

Für jedes Knotenpaar $u, v \in V$:

1. Falls $\{u, v\}$ und $\{u, w\}$ als permanent markiert worden sind, dann muss $\{v, w\}$ als Kante hinzugefügt werden (falls nicht schon vorhanden) und als permanent markiert werden.



Kernelization - Regeln

Regel 2

Für jedes Knotenpaar $u, v \in V$:

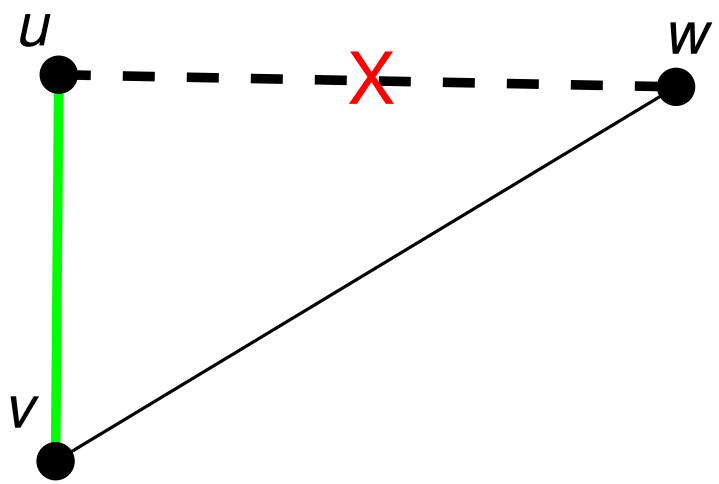
2. Falls $\{u, v\}$ als permanent und $\{u, w\}$ als verboten markiert worden sind, dann muss $\{v, w\}$ als Kante gelöscht werden (falls vorhanden) und als verboten markiert werden.

Kernelization - Regeln

Regel 2

Für jedes Knotenpaar $u, v \in V$:

2. Falls $\{u, v\}$ als permanent und $\{u, w\}$ als verboten markiert worden sind, dann muss $\{v, w\}$ als Kante gelöscht werden (falls vorhanden) und als verboten markiert werden.

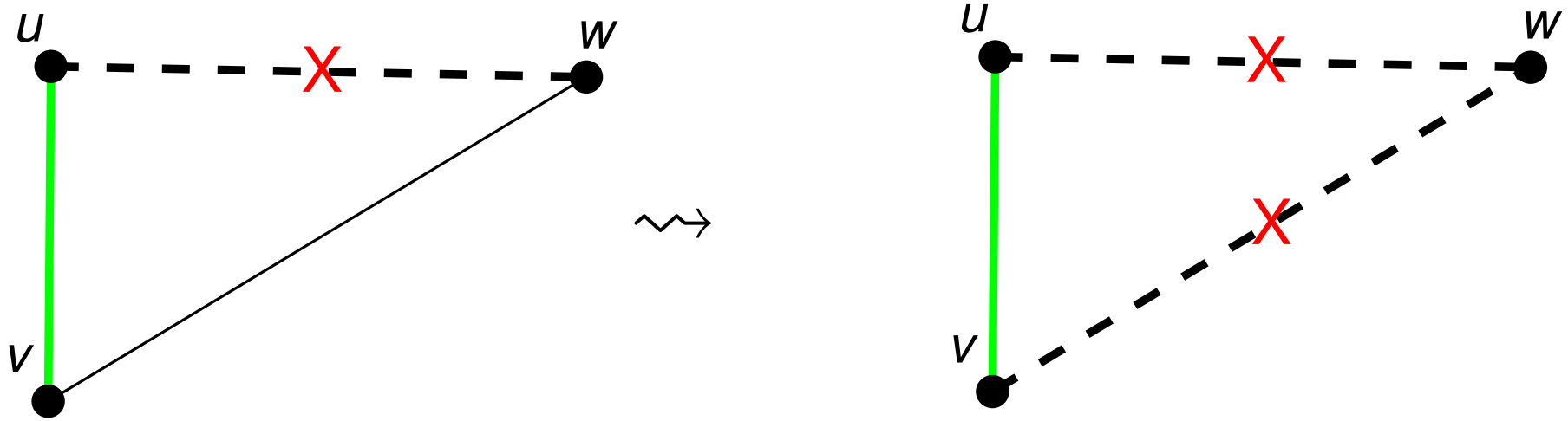


Kernelization - Regeln

Regel 2

Für jedes Knotenpaar $u, v \in V$:

2. Falls $\{u, v\}$ als permanent und $\{u, w\}$ als verboten markiert worden sind, dann muss $\{v, w\}$ als Kante gelöscht werden (falls vorhanden) und als verboten markiert werden.



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kernelization - Regeln

Regel 3

Lösche alle verbundenen Teilgraphen welche eine Clique sind

$\rightsquigarrow$ lineare Laufzeit

Lemma 7.5
Ein n-Knoten Graph lässt sich nach vollständiger Verwendung der Regeln 1 und 2 in $O(n^3)$ in einen vereinfachten Graphen transformieren.

Fallunterscheidungen

Theorem 7.6
Ein zusammenhängender n-Knoten Graph lässt sich zu einem Kernel der Größe $O(k^2)$ Knoten und $O(k^3)$ Kanten in $O(n^3)$ Zeit transformieren.

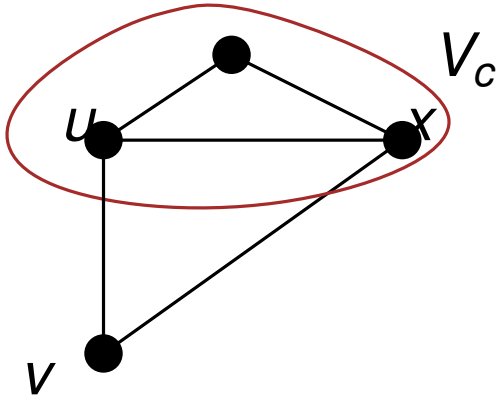
Teilbeweis für $O(k^2)$ Knoten: $|V'| \leq (2 \cdot k + 1) \cdot k$

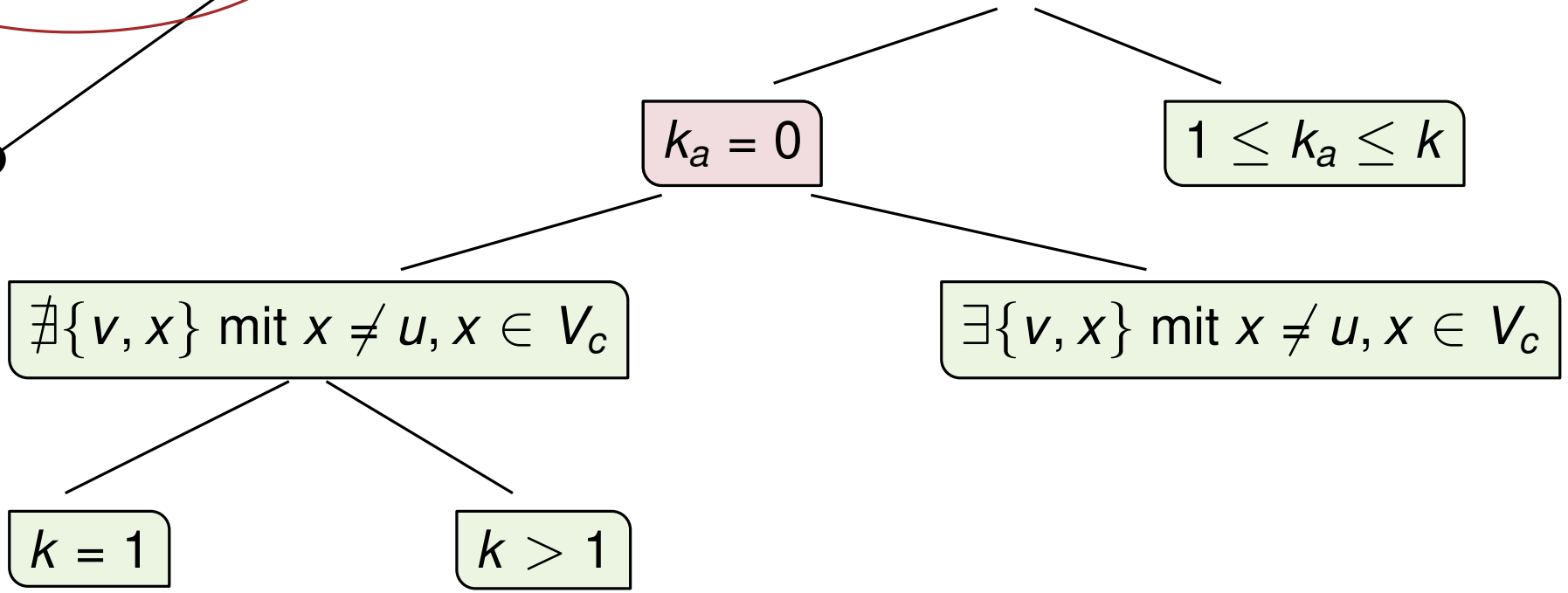
- $G = (V, E)$ ursprünglicher Graph
- $G' = (V', E')$ Kernel, o.B.d.A sei zusammenhängend
- $G'' = (V', E'')$ disjunkte Vereinigung von Cliquen
- $k = k_a + k_d$
- $V_c \subseteq V'$, größte Clique in G''

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Fallunterscheidungen

- $V_c \subseteq V'$, größte Clique in G''
- $\exists \{u, v\} \in E$ mit $u \in V_c$ und $v \notin V_c$

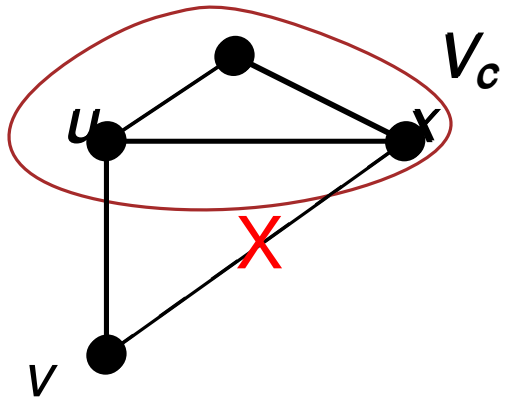


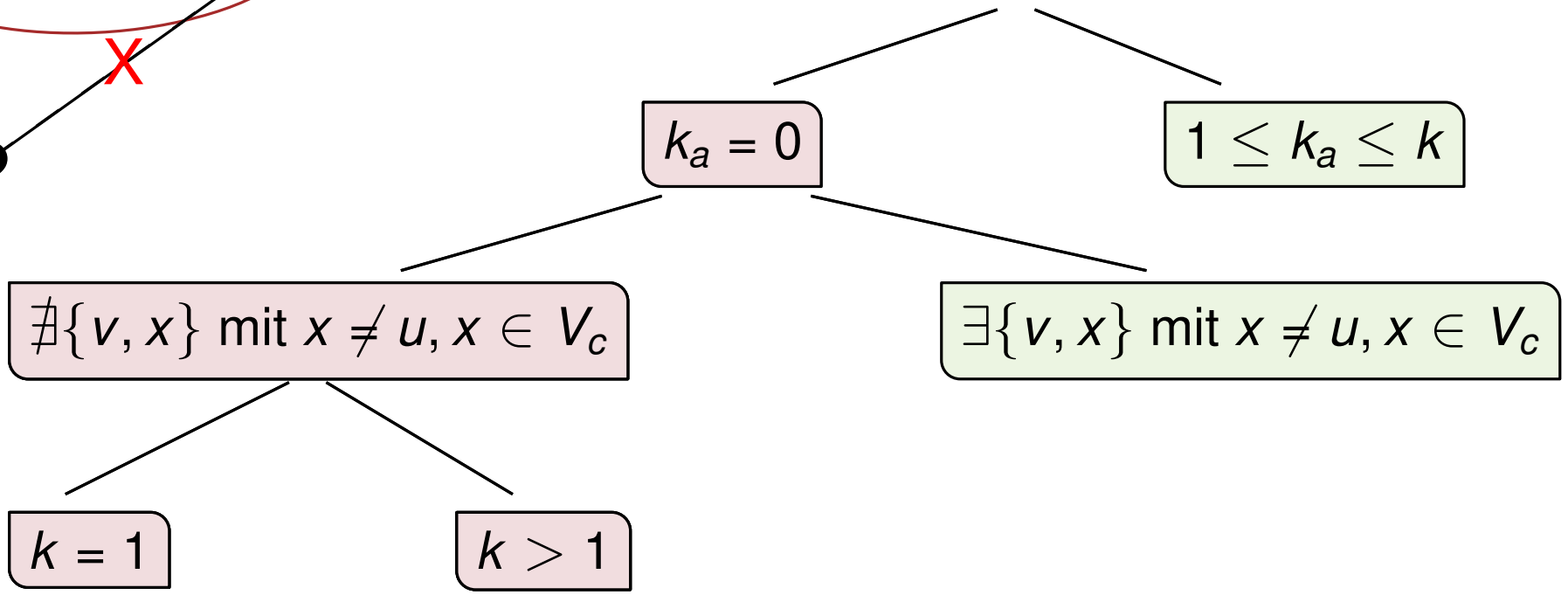


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Fallunterscheidungen

- $V_c \subseteq V'$, größte Clique in G''
- $\exists \{u, v\} \in E$ mit $u \in V_c$ und $v \notin V_c$

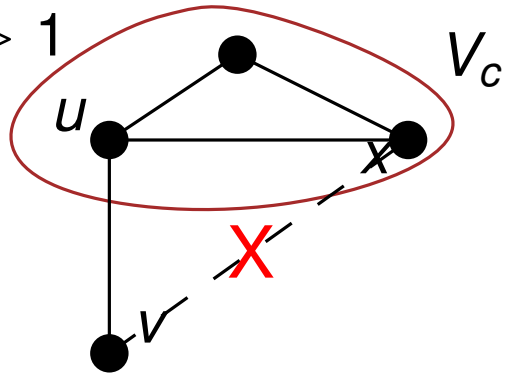




Institut für Theoretische Informatik
Lehrstuhl Algorithmik

$\nexists \{v, x\}$ mit $x \neq u, x \in V_c \wedge k_d = k > 1$

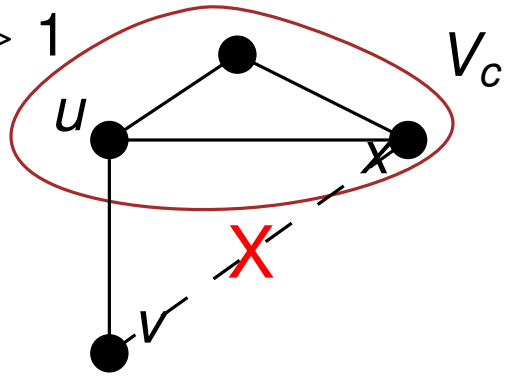
Z.z: $|V'| \leq (2 \cdot k + 1) \cdot k$



- $n = |V'| > (2 \cdot k + 1) \cdot k$ (1)
- k_d Löschoperationen $\rightsquigarrow$ höchstens $k_d + 1$ Cliquen
- $|V_c| \geq \frac{|V'|}{(k_d + 1)}$ (2)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

$\nexists \{v, x\}$ mit $x \neq u, x \in V_c \wedge k_d = k > 1$



Z.z: $|V'| \leq (2 \cdot k + 1) \cdot k$

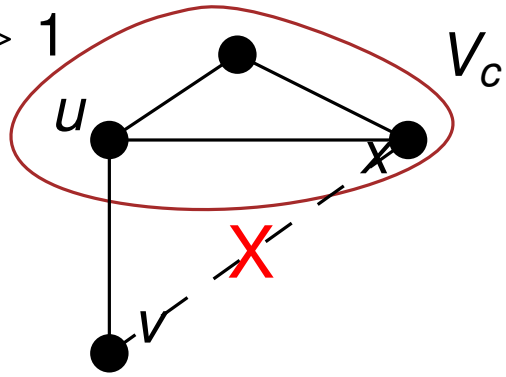
- $n = |V'| > (2 \cdot k + 1) \cdot k$ (1)
- k_d Löschoperationen $\rightsquigarrow$ höchstens $k_d + 1$ Cliquen
- $|V_c| \geq \dfrac{|V'|}{(k_d + 1)}$ (2)

$$|V_c| \overset{(2)}{\geq} \frac{|V'|}{(k_d + 1)} \overset{(1)}{>} (2 \cdot k + 1) \cdot \frac{k}{(k_d + 1)}$$

$$\overset{(k_d = k)}{=} \frac{(2 \cdot k^2 + k)}{k + 1} = \frac{(k^2 + k)}{(k + 1)} + \frac{k^2}{k + 1} = k + \frac{k^2}{k + 1} \overset{(k > 1)}{\geq} k + 1$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

$\nexists \{v, x\}$ mit $x \neq u, x \in V_c \wedge k_d = k > 1$



Z.z: $\left|V'\right| \leq (2 \cdot k + 1) \cdot k$

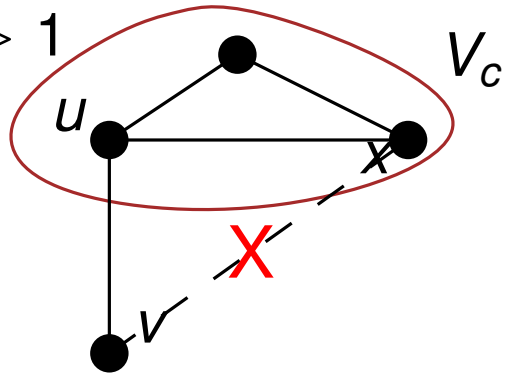
$\rightsquigarrow \left|V_c\right| \geq k + 2$

$\rightsquigarrow \left|N(u) \backslash v\right| \geq k + 1$

$\rightsquigarrow$ d.h. u und v haben mehr als k nicht-gemeinsame Nachbarn

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

$\nexists \{v, x\}$ mit $x \neq u, x \in V_c \wedge k_d = k > 1$



Z.z: $|V'| \leq (2 \cdot k + 1) \cdot k$

$\rightsquigarrow |V_c| \geq k + 2$

$\rightsquigarrow |N(u) \setminus v| \geq k + 1$

$\rightsquigarrow$ d.h. u und v haben mehr als k nicht-gemeinsame Nachbarn

$\rightsquigarrow$ Widerspruch zur Regel 1

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wir betrachten jetzt den Fall $k_d = k = 1$:

- genau zwei Cliquen
- $|V'| > (2 \cdot k + 1) \cdot k > 3$

Wir unterscheiden zwei Fälle in G'':

1. jede Clique hat mindestens 2 Knoten
2. eine Clique hat ein Knoten, die andere Clique mindestens 3

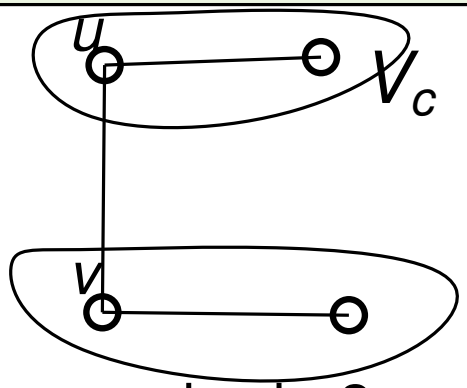
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wir betrachten jetzt den Fall $k_d = k = 1$:

- genau zwei Cliquen
- $|V'| > (2 \cdot k + 1) \cdot k > 3$

Wir unterscheiden zwei Fälle in G'':

1. jede Clique hat mindestens 2 Knoten
2. eine Clique hat ein Knoten, die andere Clique mindestens 3



$\rightsquigarrow$ Widerspruch, da $2 > k$ nicht gemeinsame Nachbarn und $\{u, v\}$ trotzdem existiert

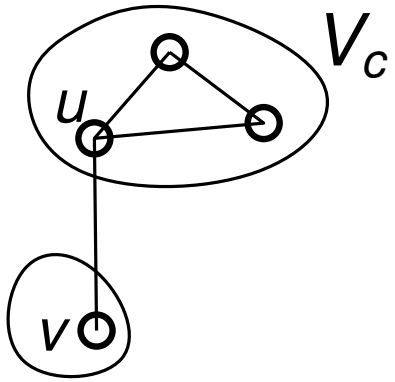
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wir betrachten jetzt den Fall $k_d = k = 1$:

- genau zwei Cliquen
- $|V'| > (2 \cdot k + 1) \cdot k > 3$

Wir unterscheiden zwei Fälle in G'':

1. jede Clique hat mindestens 2 Knoten
2. eine Clique hat ein Knoten, die andere Clique mindestens 3



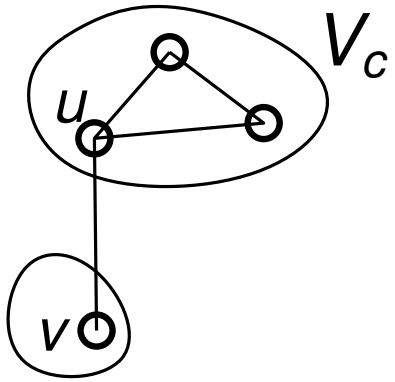
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wir betrachten jetzt den Fall $k_d = k = 1$:

- genau zwei Cliquen
- $|V'| > (2 \cdot k + 1) \cdot k > 3$

Wir unterscheiden zwei Fälle in G'':

1. jede Clique hat mindestens 2 Knoten
2. eine Clique hat ein Knoten, die andere Clique mindestens 3



$\rightsquigarrow$ Widerspruch, da 2 nicht gemeinsame Nachbarn und $\{u, v\}$ trotzdem existiert

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Lemma 8.2

Ein Graph $G = (V, E)$ besteht aus einer Vereinigung von disjunkten Cliquen $\Leftrightarrow$ es existieren keine drei verschiedene Knoten $u, v, w \in V$ mit $\{v, u\}, \{u, w\} \in E$, jedoch $\{v, w\} \notin E$

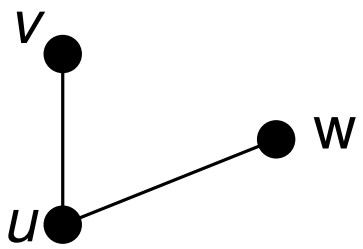
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Lemma 8.2

Ein Graph $G = (V, E)$ besteht aus einer Vereinigung von disjunkten Cliquen $\Leftrightarrow$ es existieren keine drei verschiedene Knoten $u, v, w \in V$ mit $\{v, u\}, \{u, w\} \in E$, jedoch $\{v, w\} \notin E$

"$\Rightarrow$"

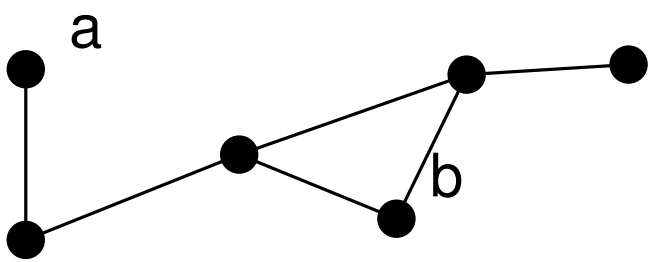
- Annahme: es existieren drei verschiedene Knoten $u, v, w \in V$ mit $\{v, u\}, \{u, w\} \in E$ und $\{v, w\} \notin E$.
- d.h. G kann keine disjunkte Vereinigung von Cliquen sein, denn sonst müsste $\{v, w\} \in E$ sein



Lemma 8.2
Ein Graph $G = (V, E)$ besteht aus einer Vereinigung von disjunkten Cliquen $\Leftrightarrow$ es existieren keine drei verschiedene Knoten $u, v, w \in V$ mit $\{v, u\}, \{u, w\} \in E$, jedoch $\{v, w\} \notin E$

"$\Leftarrow$"

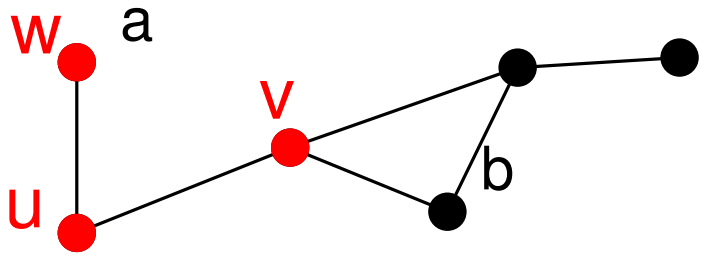
- G besteht nicht aus einer disjunkten Vereinigung von Cliquen
- $\exists a, b \in V$ so dass a und b durch einen Pfad der Länge mindestens 3 verbunden sind
- man wähle von a aus die ersten 3 Knoten in Richtung b (die Teil eines kürzesten Weges von a zu b sind) als w, u und v



Lemma 8.2
Ein Graph $G = (V, E)$ besteht aus einer Vereinigung von disjunkten Cliquen $\Leftrightarrow$ es existieren keine drei verschiedene Knoten $u, v, w \in V$ mit $\{v, u\}, \{u, w\} \in E$, jedoch $\{v, w\} \notin E$

"$\Leftarrow$"

- G besteht nicht aus einer disjunkten Vereinigung von Cliquen
- $\exists a, b \in V$ so dass a und b durch einen Pfad der Länge mindestens 3 verbunden sind
- man wähle von a aus die ersten 3 Knoten in Richtung b (die Teil eines kürzesten Weges von a zu b sind) als w, u und v



Algorithmus

- G ist bereits Vereinigung von diskunkten Cliquen: gib G zurück als Lösung
- falls $k \geq 0$ dann return (keine Lösung in diesem Aufruf)
- Ansonsten identifiziere $u, v, w \in V$ mit $\{u, v\}, \{u, w\} \in E$ jedoch $\{v, w\} \notin E$. Rufe folgende drei Instanzen rekursiv auf:
  - $E' := E \setminus \{\{u, v\}\}$ und $k' := k - 1$
  - $E' := E \setminus \{\{u, w\}\}$ und $k' := k - 1$
  - $E' := E \cup \{\{v, w\}\}$ und $k' := k - 1$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Algorithmus

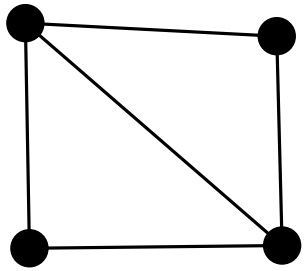
- G ist bereits Vereinigung von diskunkten Cliquen: gib G zurück als Lösung
- falls $k \geq 0$ dann return (keine Lösung in diesem Aufruf)
- Ansonsten identifiziere $u, v, w \in V$ mit $\{u, v\}, \{u, w\} \in E$ jedoch $\{v, w\} \notin E$. Rufe folgende drei Instanzen rekursiv auf:
  - $E' := E \setminus \{\{u, v\}\}$ und $k' := k - 1$
  - $E' := E \setminus \{\{u, w\}\}$ und $k' := k - 1$
  - $E' := E \cup \{\{v, w\}\}$ und $k' := k - 1$

$\rightsquigarrow$ Laufzeit: $O(3^k \cdot n^3)$ (n ist Anzahl Knoten von G)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Algorithmus Beispiel

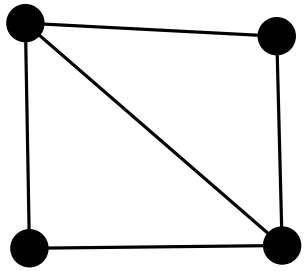
Gegeben sei ein reduzierter Graph und k = 1



Algorithmus Beispiel

Gegeben sei ein reduzierter Graph und k = 1

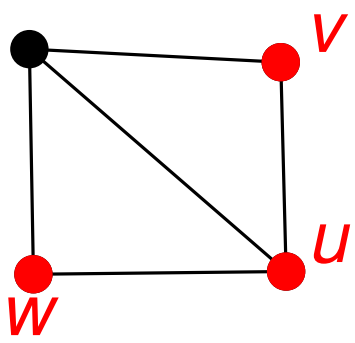
- identifiziere $u, v, w \in V$ mit $\{u, v\}, \{u, w\} \in E$ jedoch $\{v, w\} \notin E$



Algorithmus Beispiel

Gegeben sei ein reduzierter Graph und k = 1

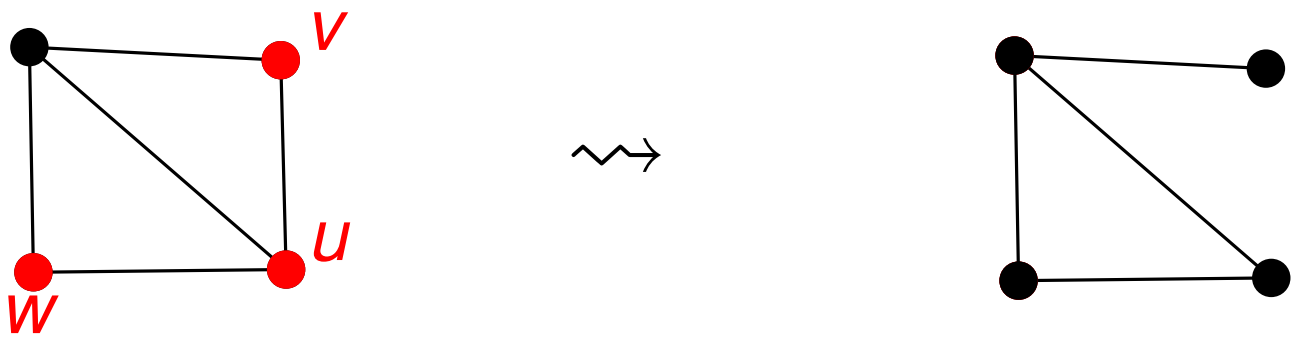
- identifiziere $u, v, w \in V$ mit $\{u, v\}, \{u, w\} \in E$ jedoch $\{v, w\} \notin E$



Algorithmus Beispiel

Gegeben sei ein reduzierter Graph und k = 1

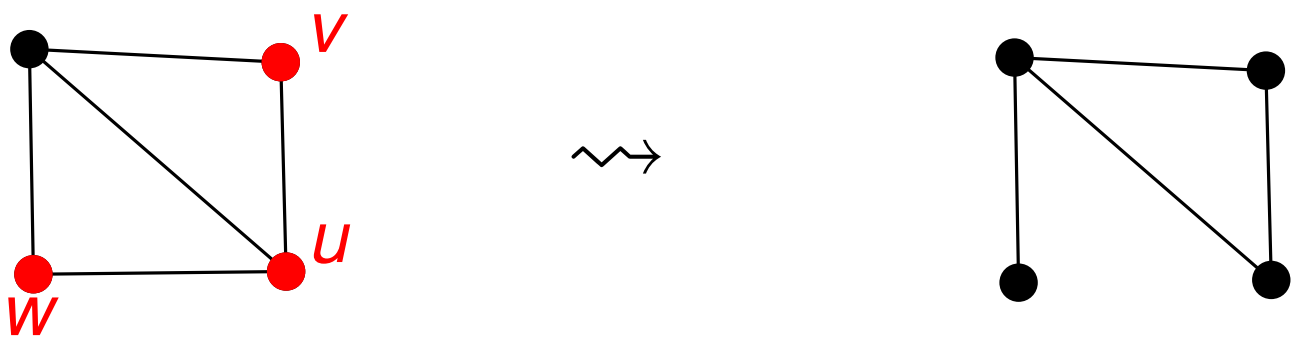
- $E' := E \setminus \{\{u, v\}\}$ und $k' := 1 - 1 = 0$



Algorithmus Beispiel

Gegeben sei ein reduzierter Graph und k = 1

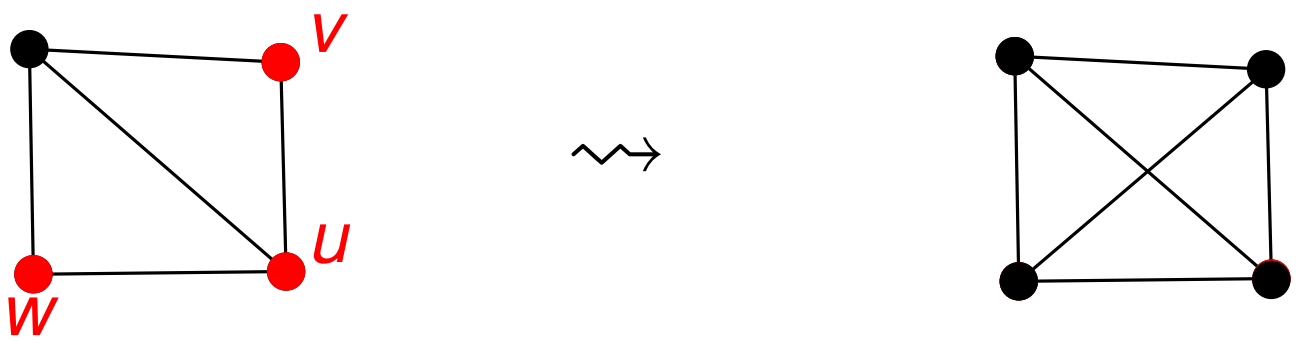
- $E' := E \setminus \{\{u, w\}\}$ und $k' := 1 - 1 = 0$



Algorithmus Beispiel

Gegeben sei ein reduzierter Graph und k = 1

- $E' := E \cup \{\{v, w\}\}$ und $k' := 1 - 1 = 0$



Fixed-Parameter Tractable

Sei L ein parametrisiertes Problem der Form (I, k). Das Problem L gehört zur FPT Klasse falls folgendes gilt:

- man in Zeit $f(k) \cdot |L|^{O(1)}$ entscheiden kann ob L eine Lösung besitzt
- Funktion f hängt nur vom Parameter k ab

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Fixed-Parameter Tractable

Sei L ein parametrisiertes Problem der Form (I, k). Das Problem L gehört zur FPT Klasse falls folgendes gilt:

- man in Zeit $f(k) \cdot |L|^{O(1)}$ entscheiden kann ob L eine Lösung besitzt
- Funktion f hängt nur vom Parameter k ab

Ein Problem L ist in FPT $\iff$ L besitzt einen Kernel und ist entscheidbar

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Fixed-Parameter Tractable

Sei L ein parametrisiertes Problem der Form (I, k). Das Problem L gehört zur FPT Klasse falls folgendes gilt:

- man in Zeit $f(k) \cdot |L|^{O(1)}$ entscheiden kann ob L eine Lösung besitzt
- Funktion f hängt nur vom Parameter k ab

Ein Problem L ist in FPT $\iff$ L besitzt einen Kernel und ist entscheidbar

"$\Leftarrow$"

- Kernelization Algorithmus läuft in $|L|^{O(1)}$
- reduzierte Instanzgröße sei $g(k)$
- sei f ein beliebiger Entscheidungsalgorithmus
- Gesamtlaufzeit: $f(g(k)) + |L|^{O(1)}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Zusammenfassung - Cluster Editing

- Kernelization
  - Vereinfachung des Problems
  - 3 Regeln
  - Größengarantie - Beweis

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Zusammenfassung - Cluster Editing

- Kernelization
  - Vereinfachung des Problems
  - 3 Regeln
  - Größengarantie - Beweis

- Algorithmus
  - Lösung des Problems
  - FPT vs Kernelization

Institut für Theoretische Informatik
Lehrstuhl Algorithmik