Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Approximationsalgorithmen

Vorlesung 11:
Minimalgrad-Spannbaum per lokale Suche

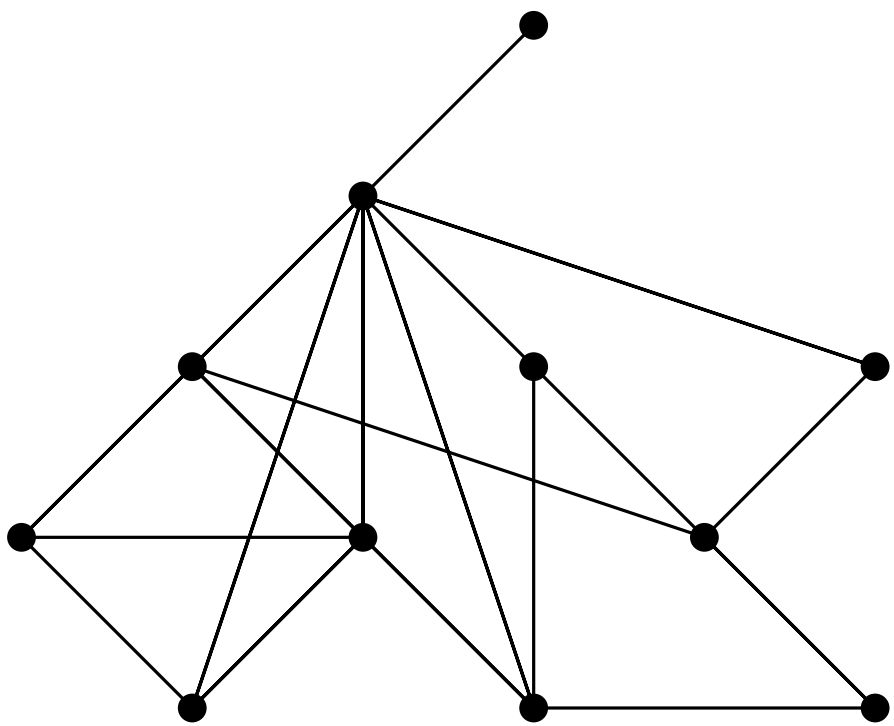
Folien von Joachim Spoerhase und Steven Chaplick

[Williamson & Shmoys: §2.6]

Alexander Wolff Lehrstuhl für Informatik I WS 2018/19

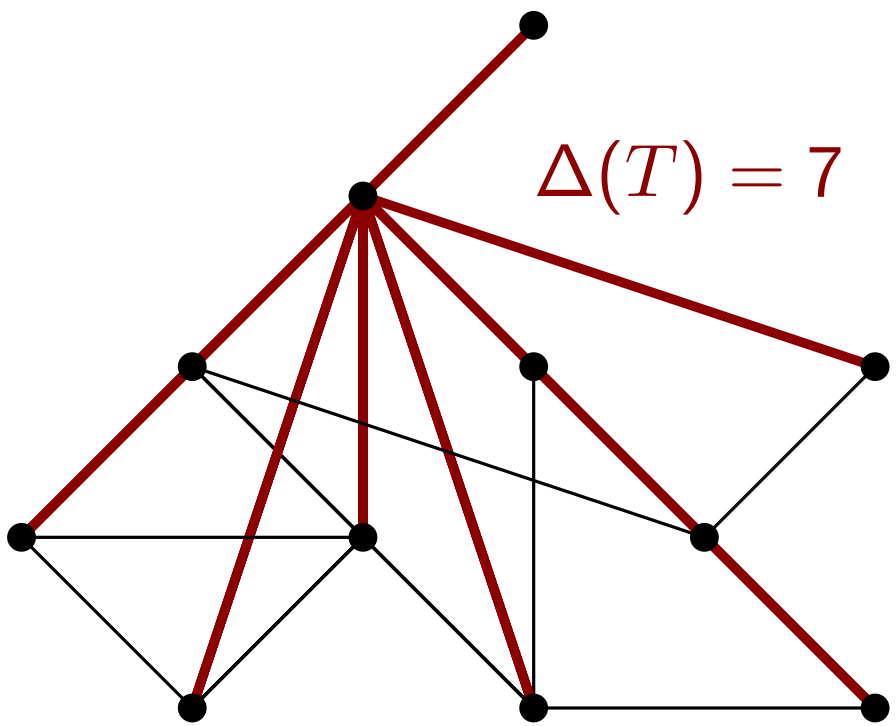
Minimum-Degree Spanning Tree

Gegeben sei ein zusammenhängender Graph $G = (V, E)$. Gesucht ist ein Spannbaum T, dessen Maximalgrad $\Delta(T)$ minimal unter allen Spannbäumen ist.



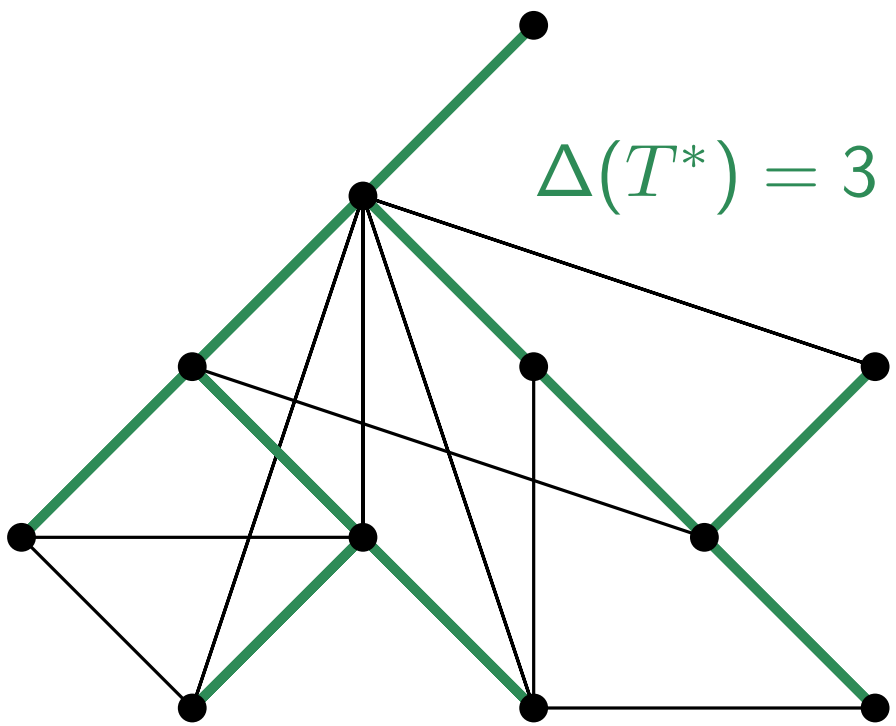
Minimum-Degree Spanning Tree

Gegeben sei ein zusammenhängender Graph $G = (V, E)$. Gesucht ist ein Spannbaum T, dessen Maximalgrad $\Delta(T)$ minimal unter allen Spannbäumen ist.



Minimum-Degree Spanning Tree

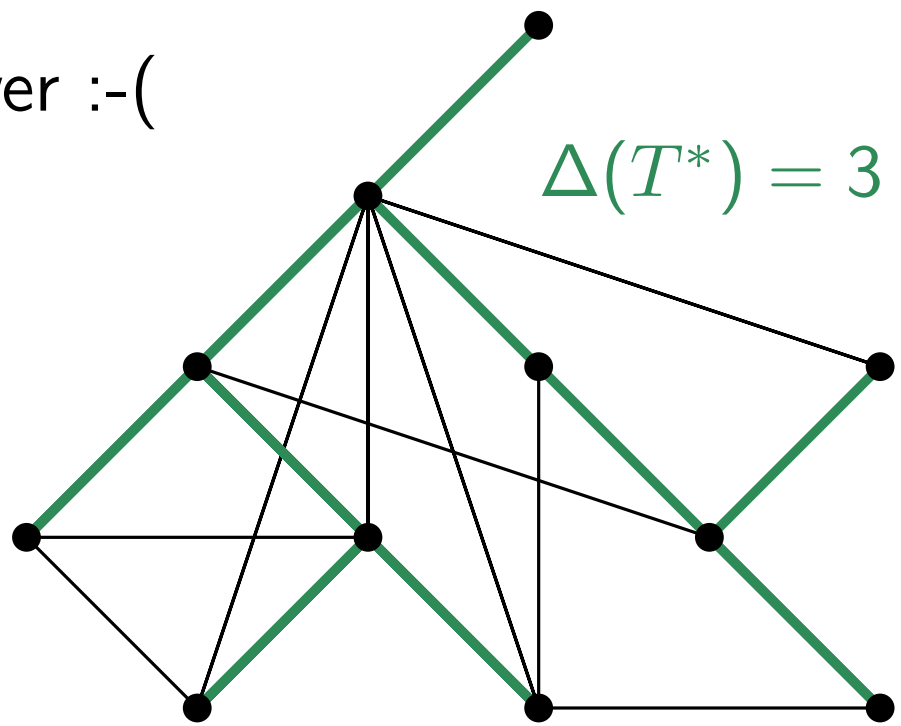
Gegeben sei ein zusammenhängender Graph $G = (V, E)$. Gesucht ist ein Spannbaum T, dessen Maximalgrad $\Delta(T)$ minimal unter allen Spannbäumen ist.



Minimum-Degree Spanning Tree

Gegeben sei ein zusammenhängender Graph $G = (V, E)$. Gesucht ist ein Spannbaum T, dessen Maximalgrad $\Delta(T)$ minimal unter allen Spannbäumen ist.

NP-schwer :-(

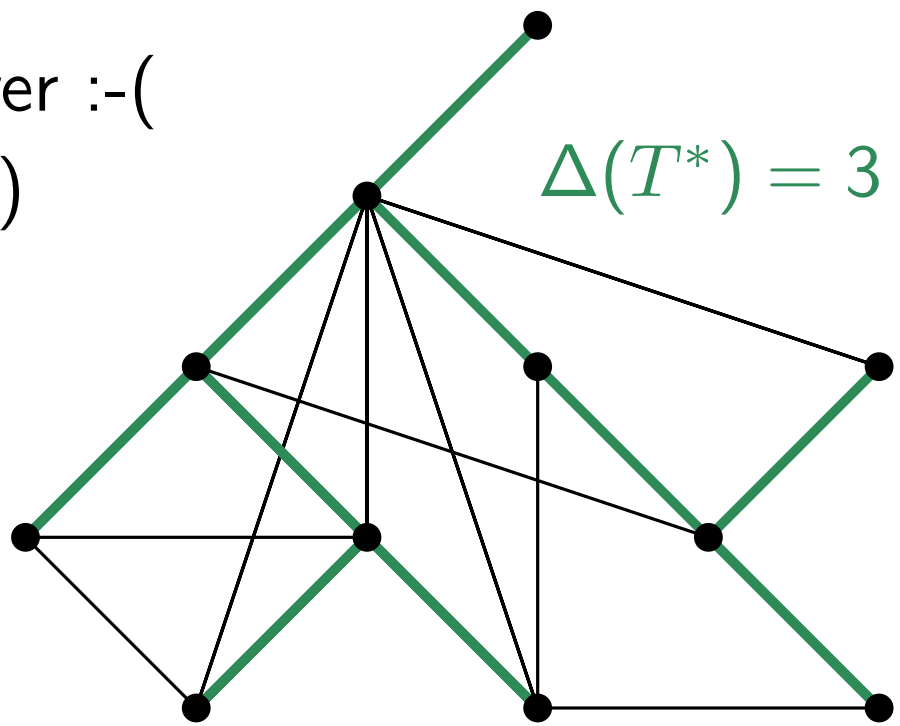


Minimum-Degree Spanning Tree

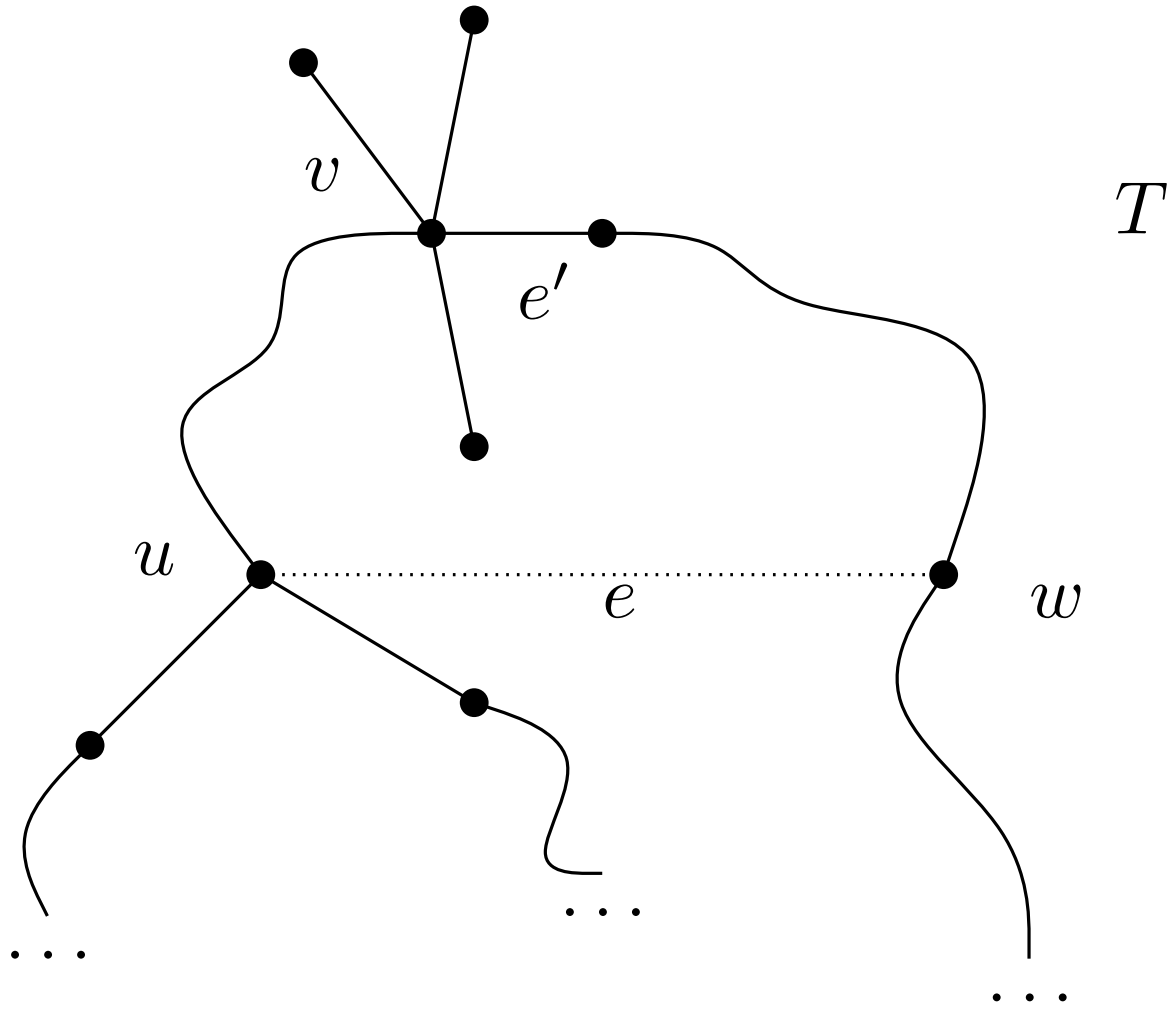
Gegeben sei ein zusammenhängender Graph $G = (V, E)$. Gesucht ist ein Spannbaum T, dessen Maximalgrad $\Delta(T)$ minimal unter allen Spannbäumen ist.

NP-schwer :-(

(warum?)



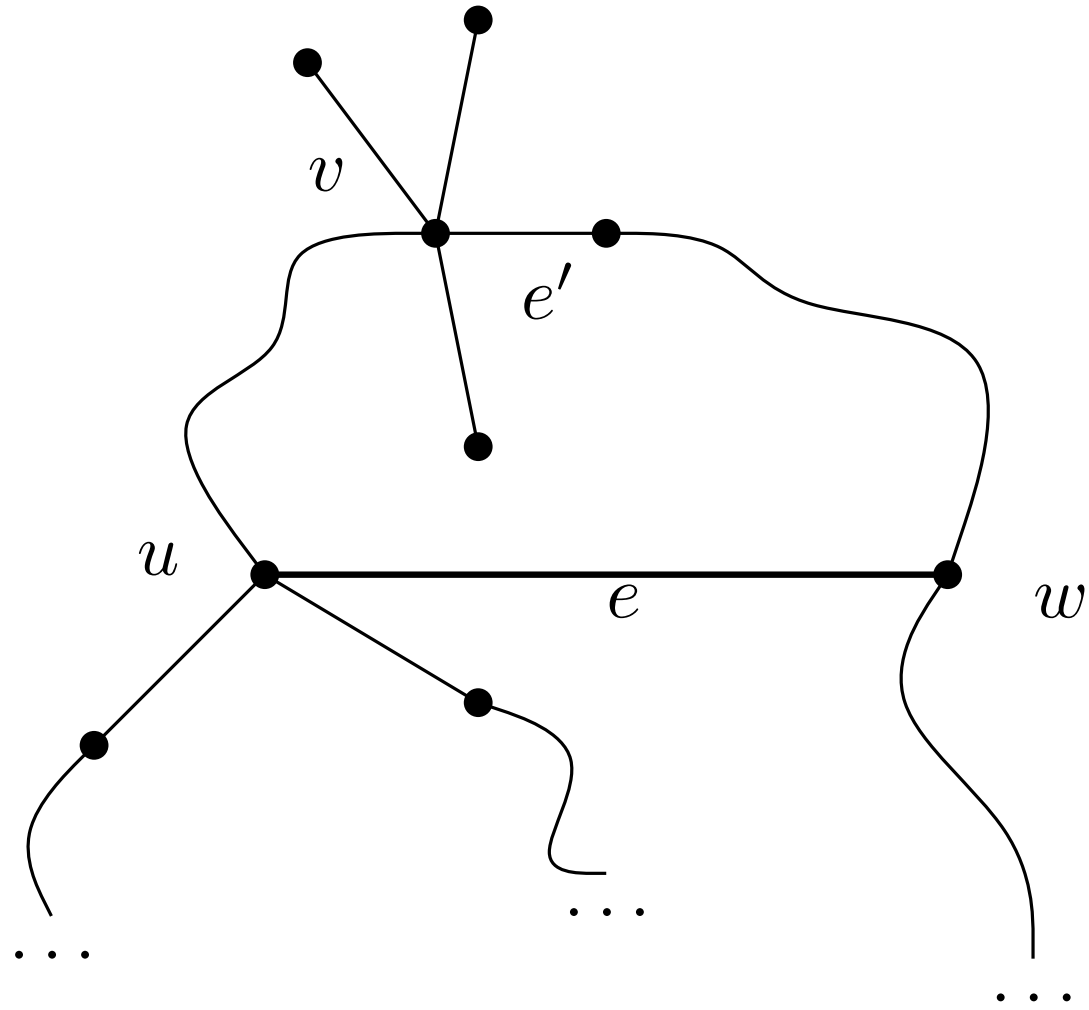
Kantenflip



$E(T)$

$E(G) - E(T)$

Kantenflip

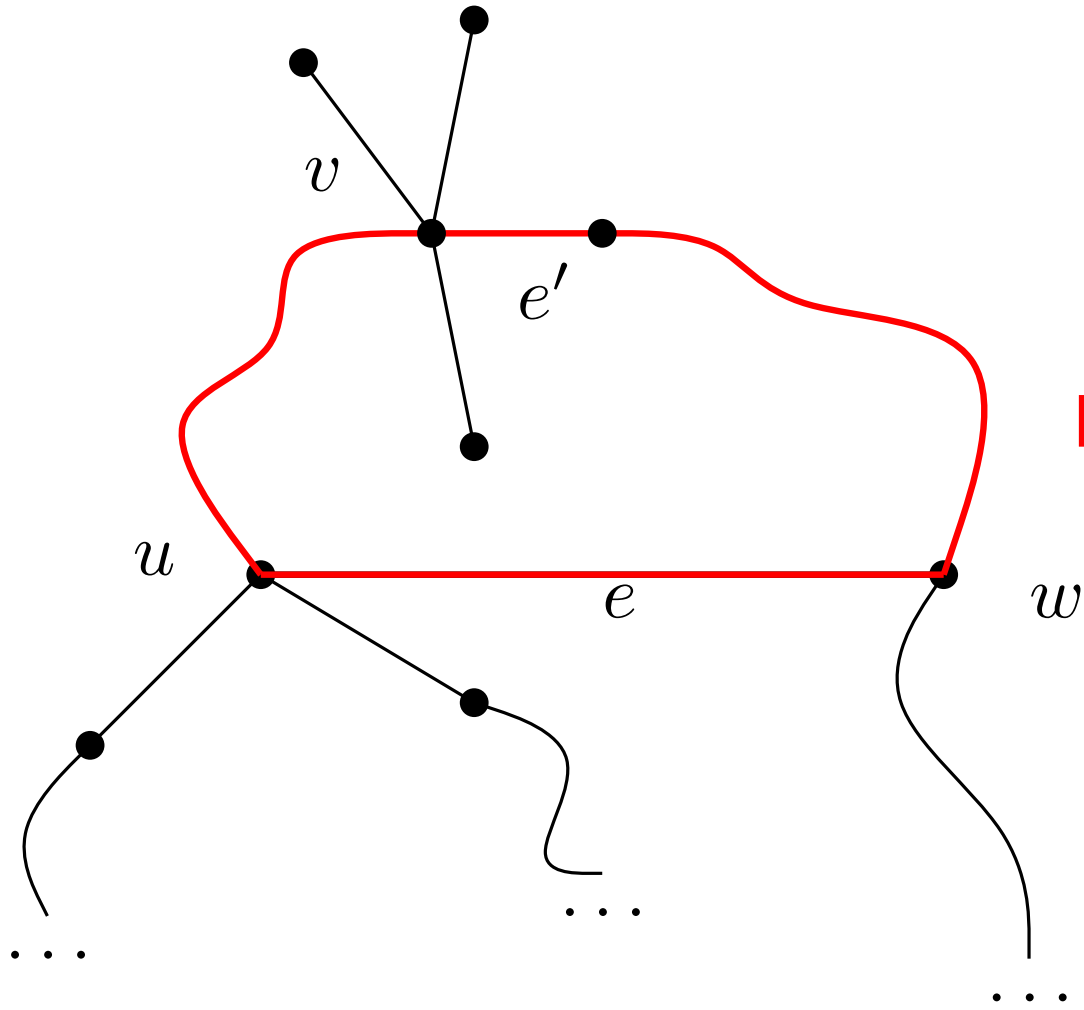


$T + e$

—— $E(T)$

..... $E(G) - E(T)$

Kantenflip



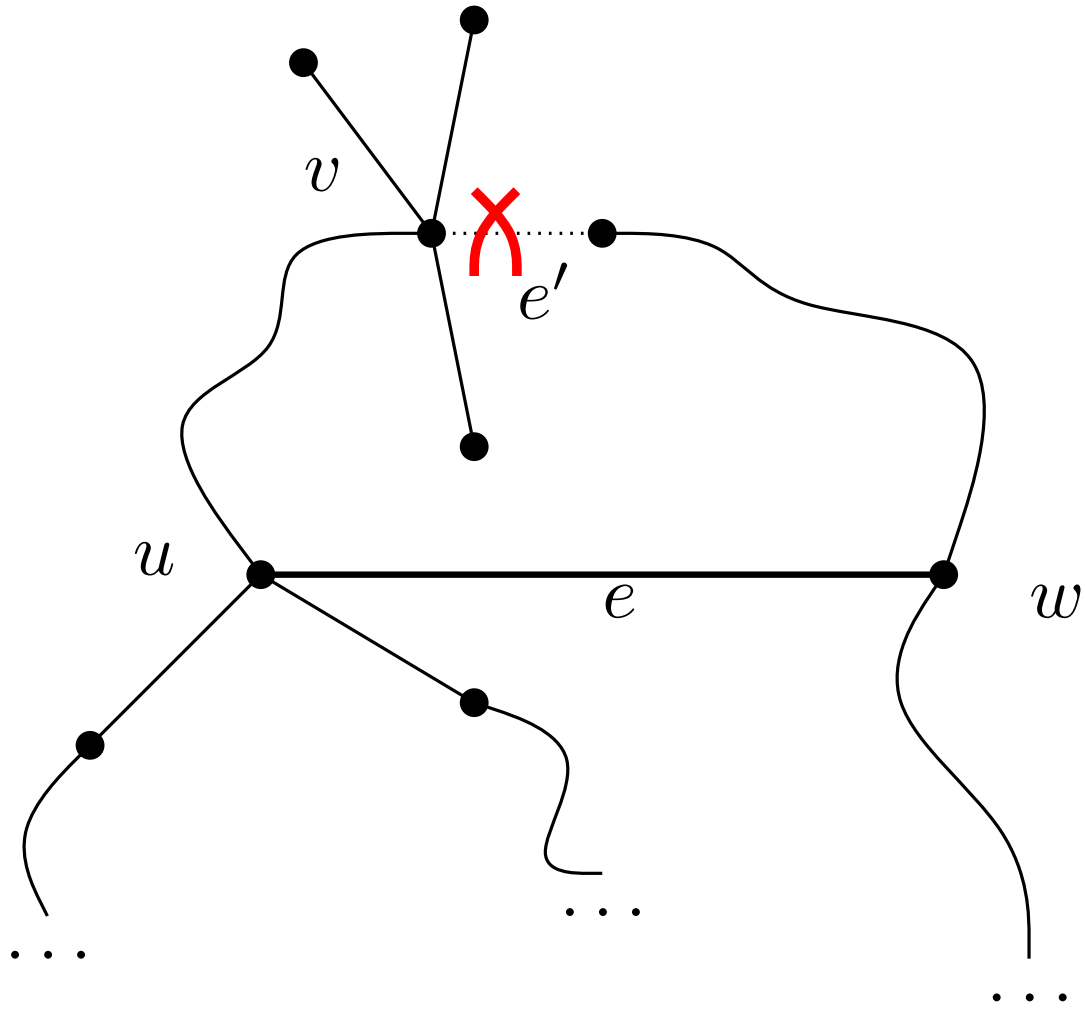
$T + e$

Kreis!

$E(T)$

$E(G) - E(T)$

Kantenflip



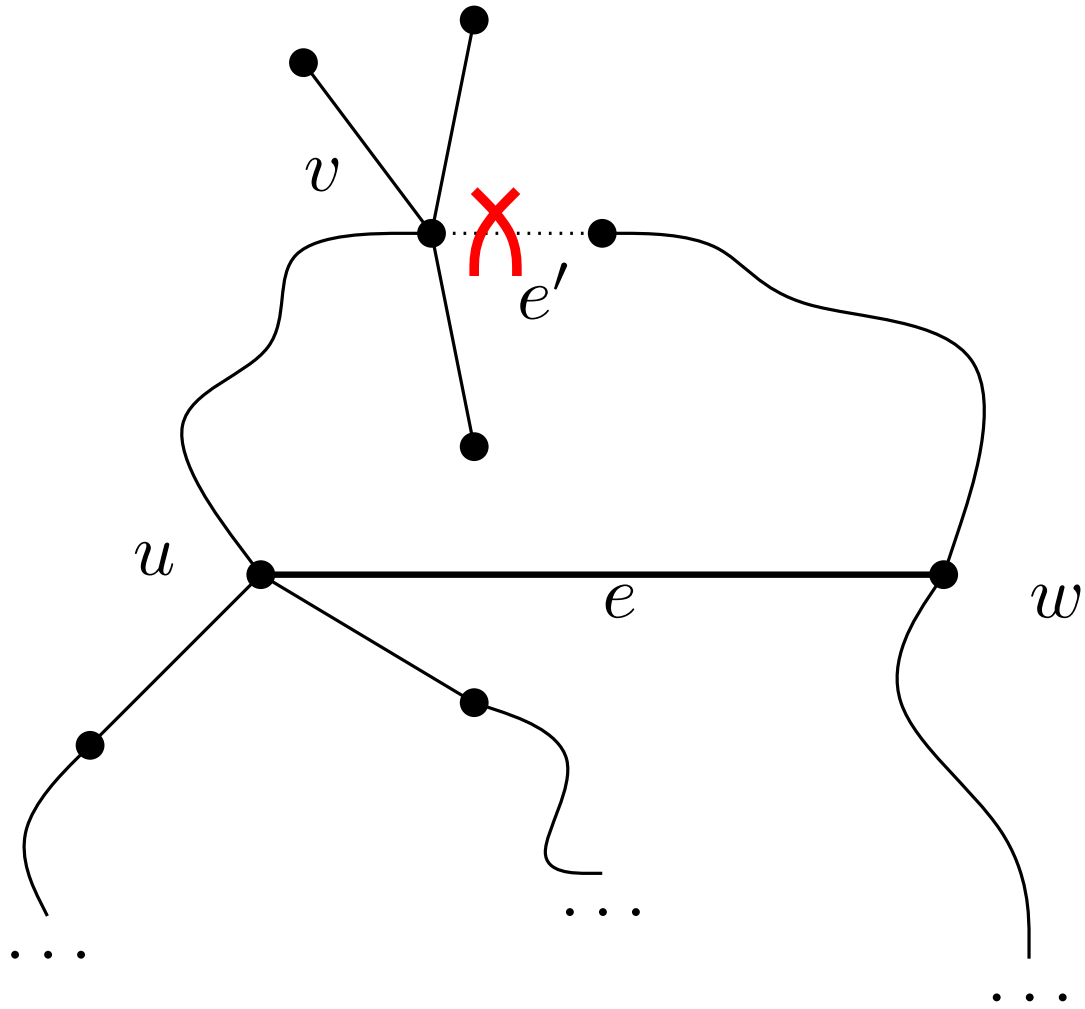
$T + e - e'$

$E(T)$

$E(G) - E(T)$

Kantenflip

Verbesserung, falls $\deg_T(v) >$



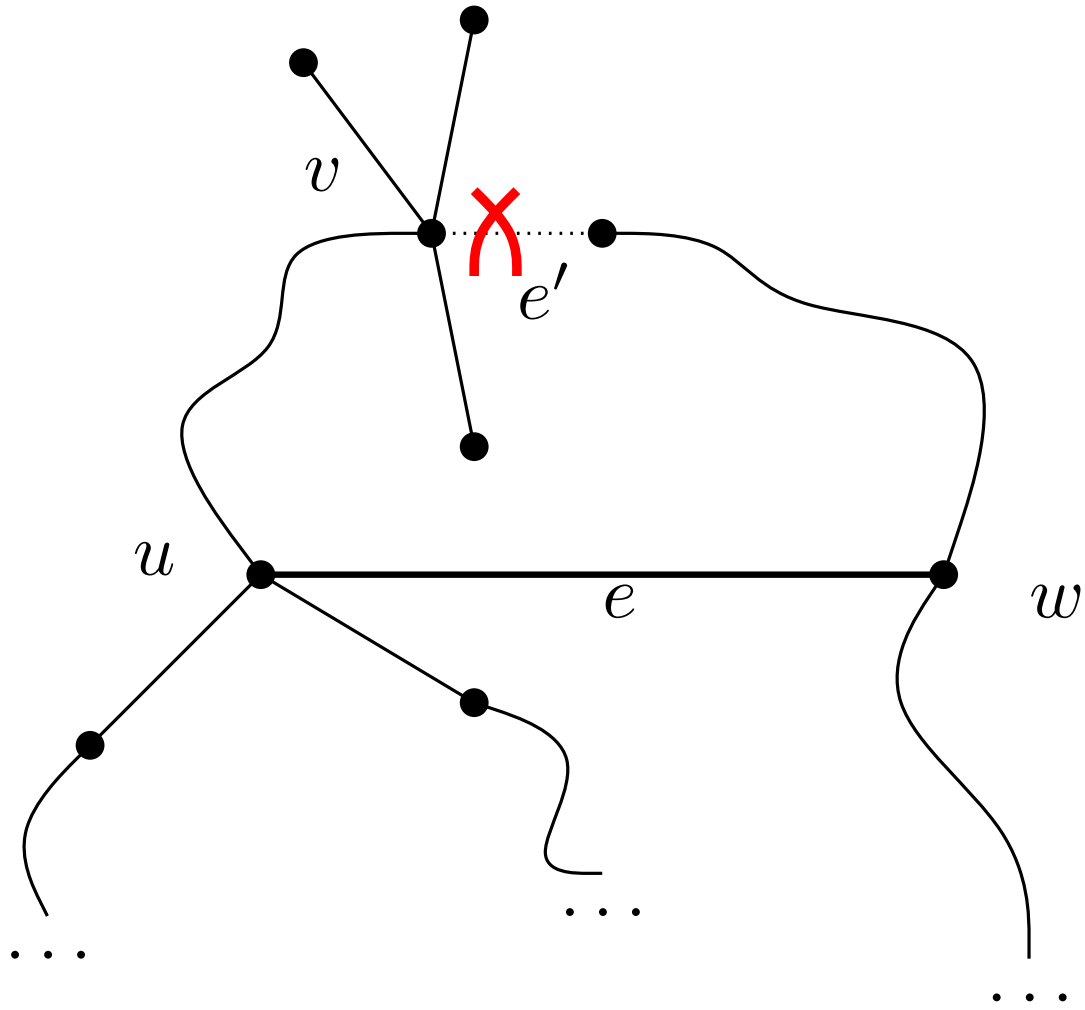
$T + e - e'$

$E(T)$

$E(G) - E(T)$

Kantenflip

Verbesserung, falls $\deg_T(v) > \max\{\deg_T(u) + 1, \deg_T(w) + 1\}$



$T + e - e'$

$E(T)$

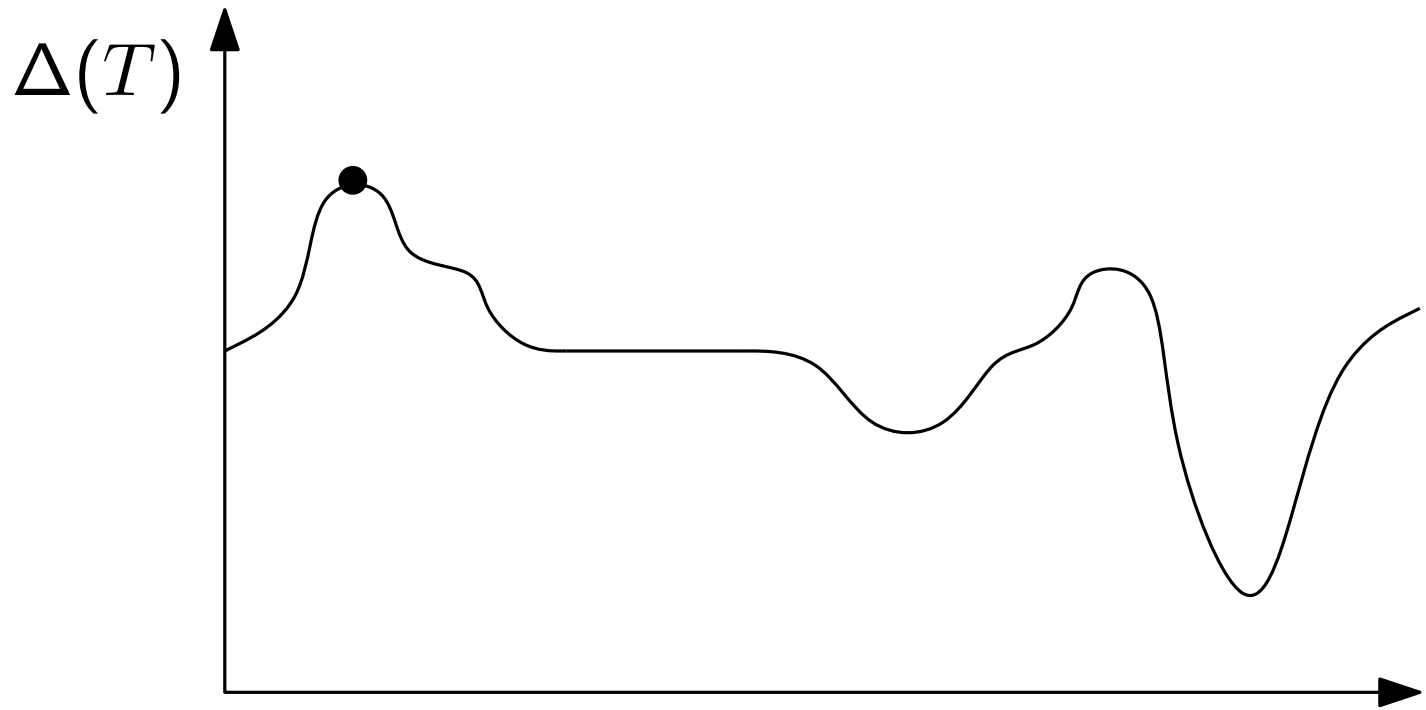
$E(G) - E(T)$

Lokale Suche

- Beginne mit beliebigem Spannbaum T von G

Lokale Suche

- Beginne mit beliebigem Spannbaum T von G
- führe Kantenflips durch, solange dadurch Verbesserung

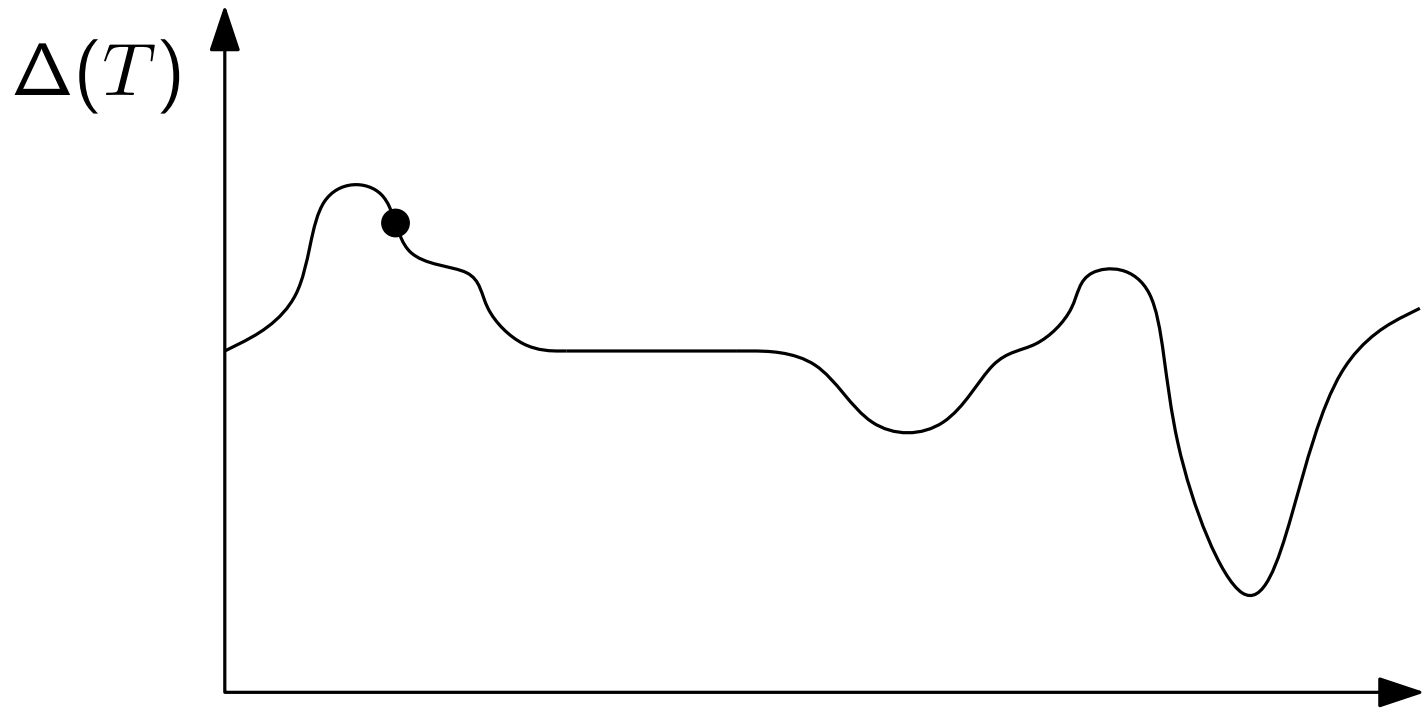


Achtung: Stark vereinfachte Darstellung!

Spannbäume T von G

Lokale Suche

- Beginne mit beliebigem Spannbaum T von G
- führe Kantenflips durch, solange dadurch Verbesserung

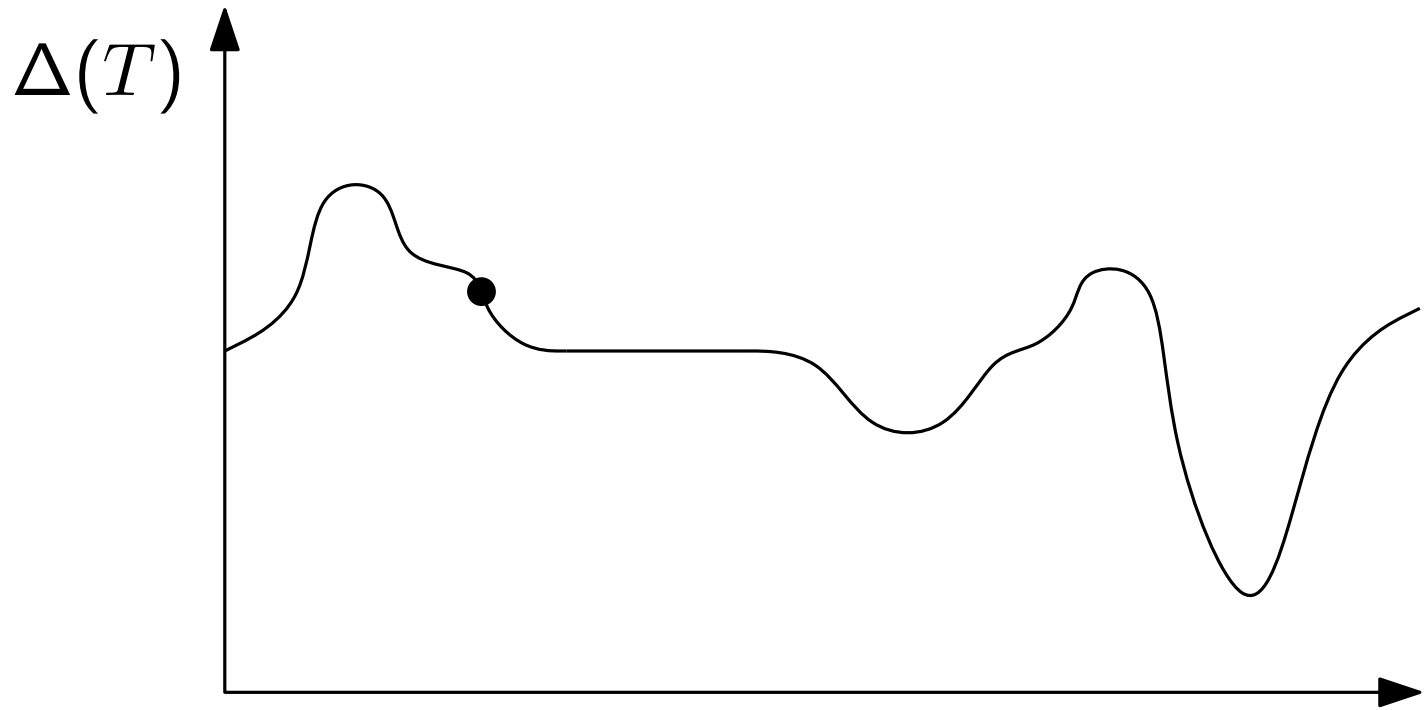


Achtung: Stark vereinfachte Darstellung!

Spannbäume T von G

Lokale Suche

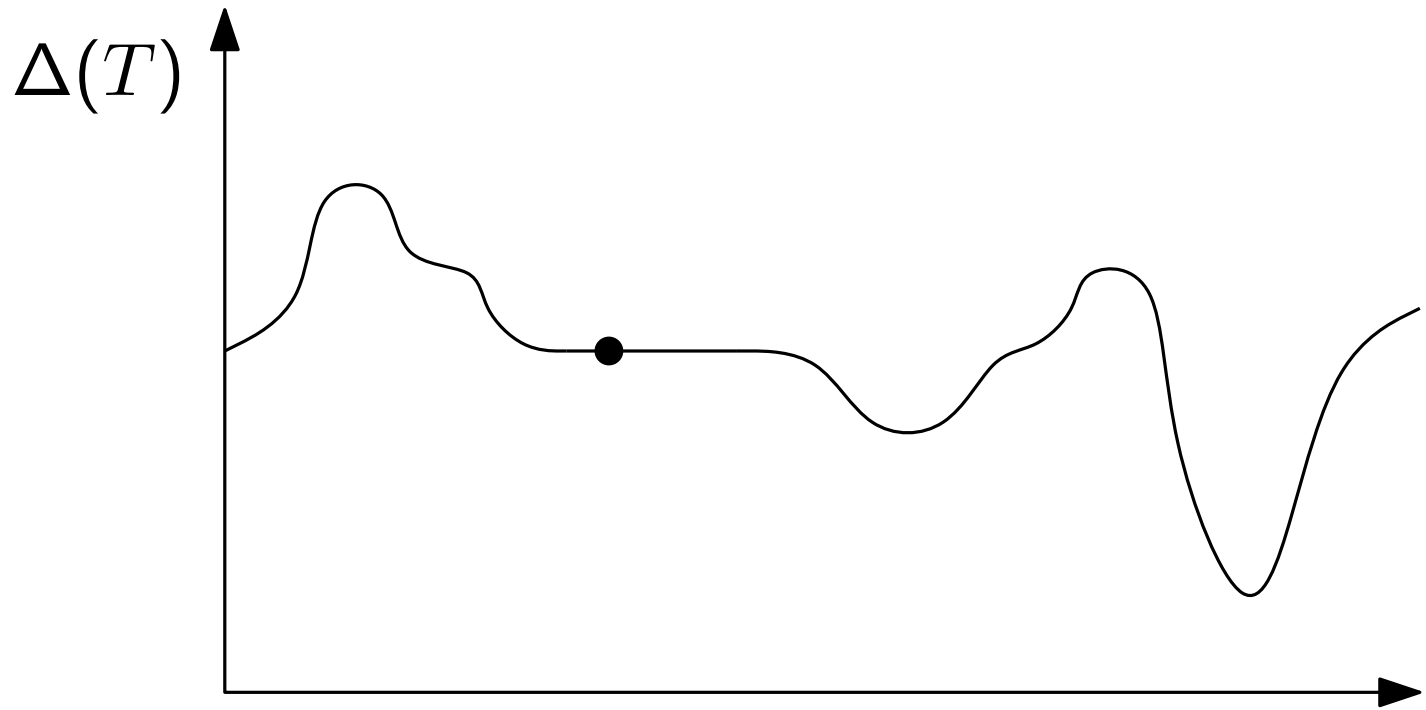
- Beginne mit beliebigem Spannbaum T von G
- führe Kantenflips durch, solange dadurch Verbesserung



Achtung: Stark vereinfachte Darstellung!

Lokale Suche

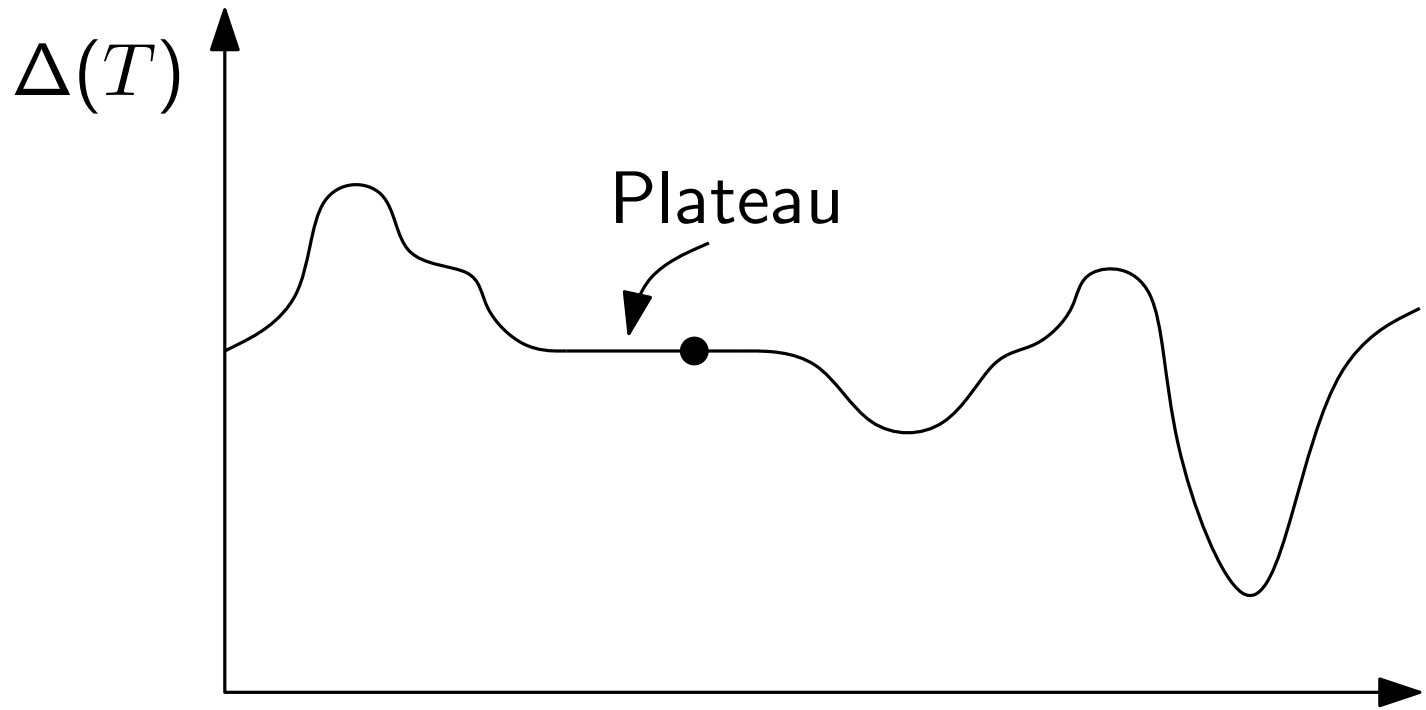
- Beginne mit beliebigem Spannbaum T von G
- führe Kantenflips durch, solange dadurch Verbesserung



Achtung: Stark vereinfachte Darstellung!

Lokale Suche

- Beginne mit beliebigem Spannbaum T von G
- führe Kantenflips durch, solange dadurch Verbesserung

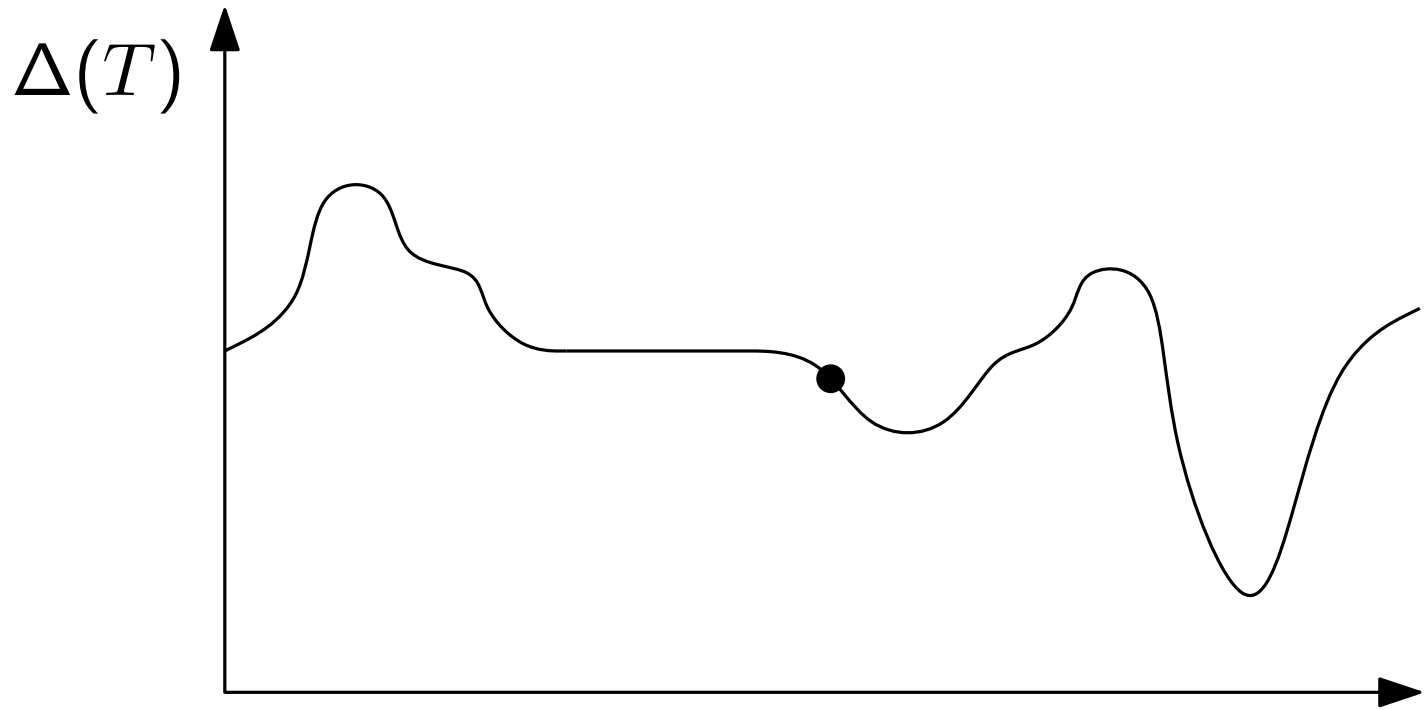


Achtung: Stark vereinfachte Darstellung!

Flip verringert $\Delta(T)$ nicht immer!

Lokale Suche

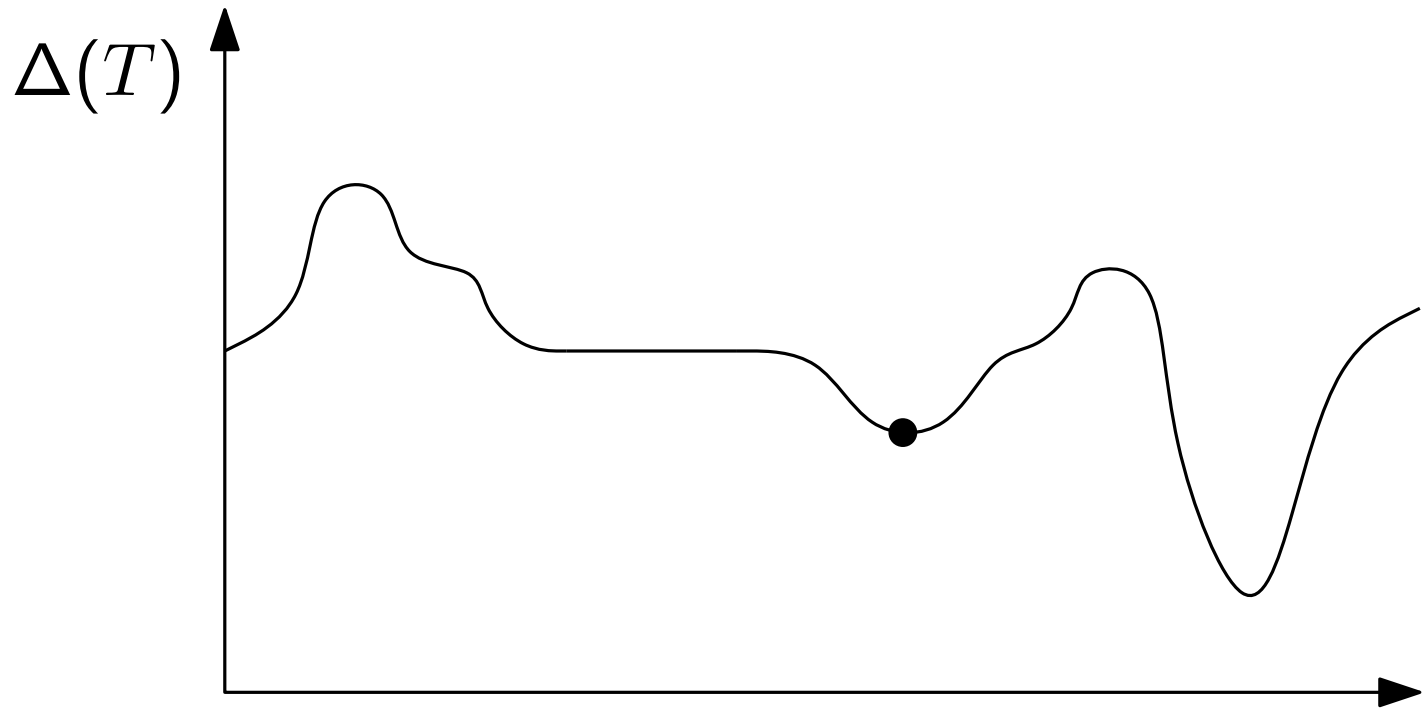
- Beginne mit beliebigem Spannbaum T von G
- führe Kantenflips durch, solange dadurch Verbesserung



Achtung: Stark vereinfachte Darstellung!

Lokale Suche

- Beginne mit beliebigem Spannbaum T von G
- führe Kantenflips durch, solange dadurch Verbesserung

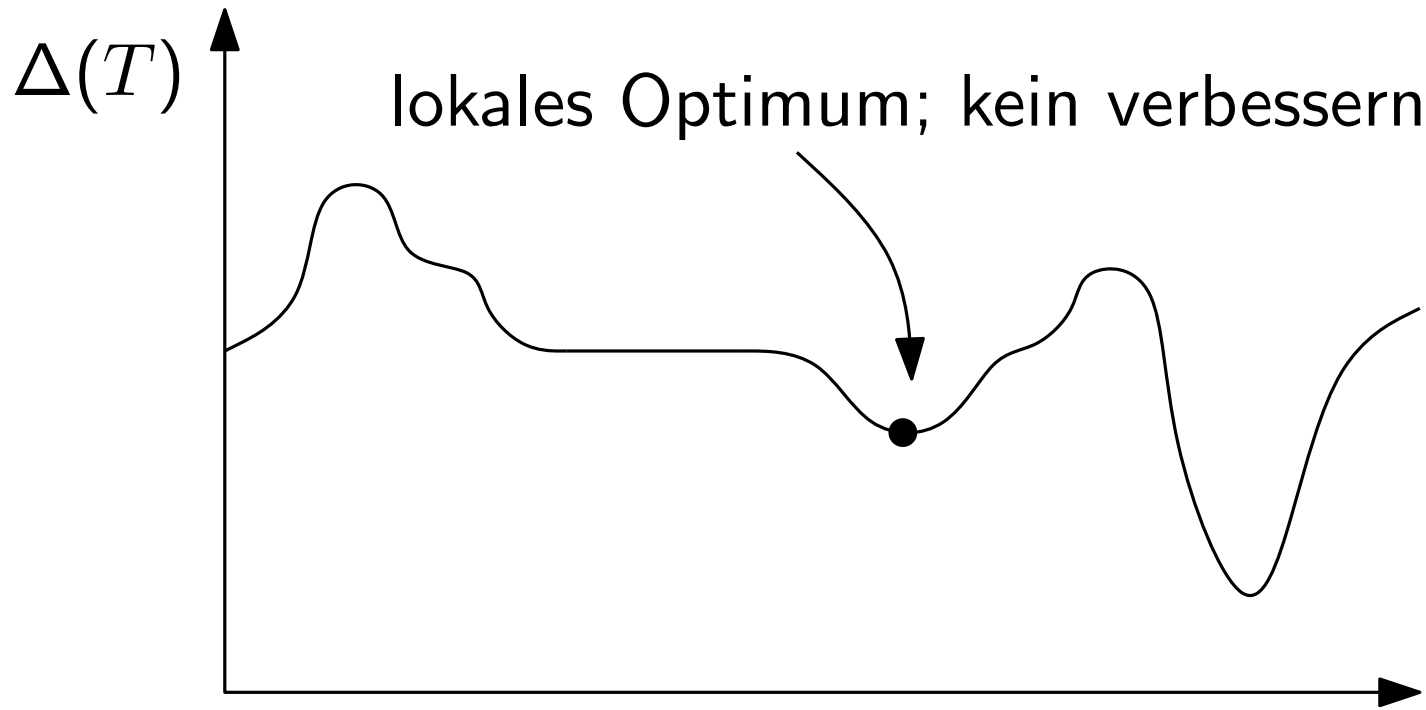


Achtung: Stark vereinfachte Darstellung!

Spannbäume T von G

Lokale Suche

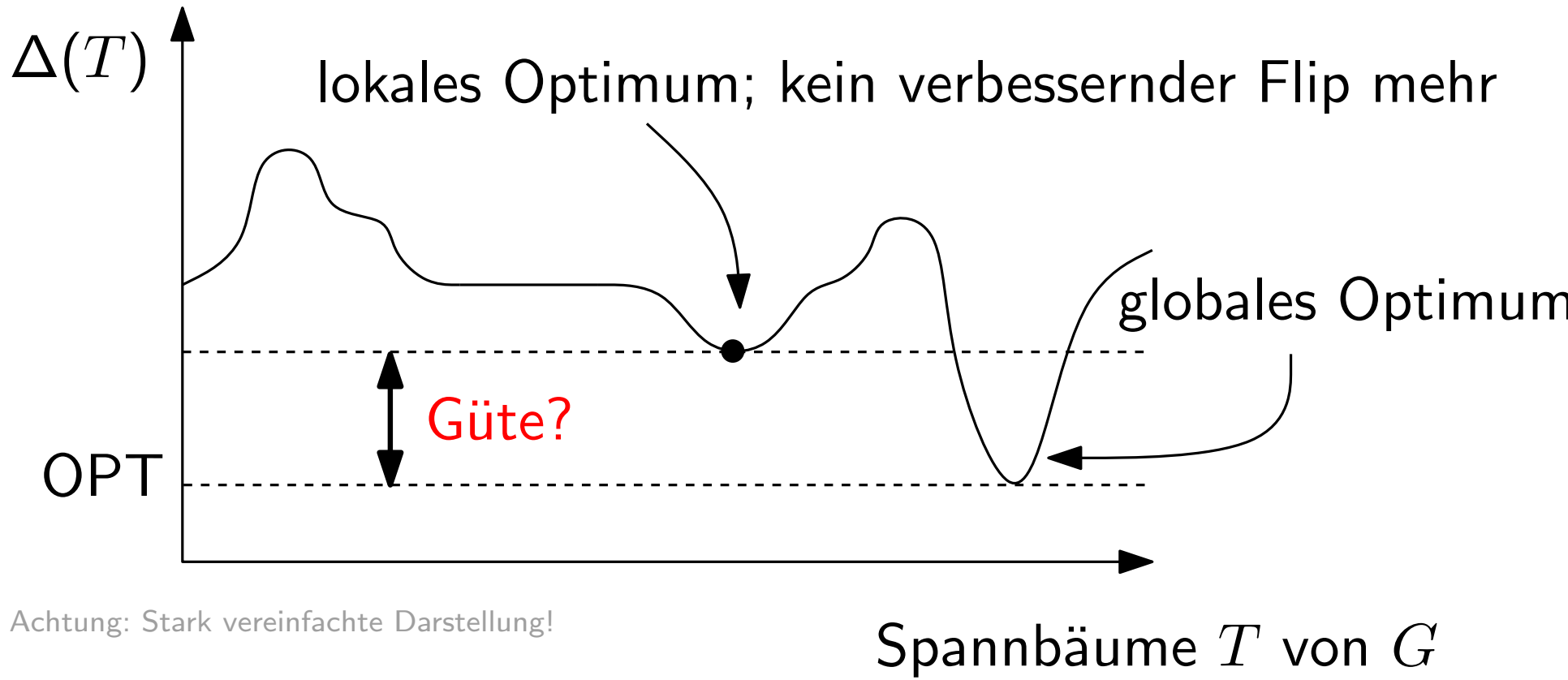
- Beginne mit beliebigem Spannbaum T von G
- führe Kantenflips durch, solange dadurch Verbesserung



Achtung: Stark vereinfachte Darstellung!

Lokale Suche

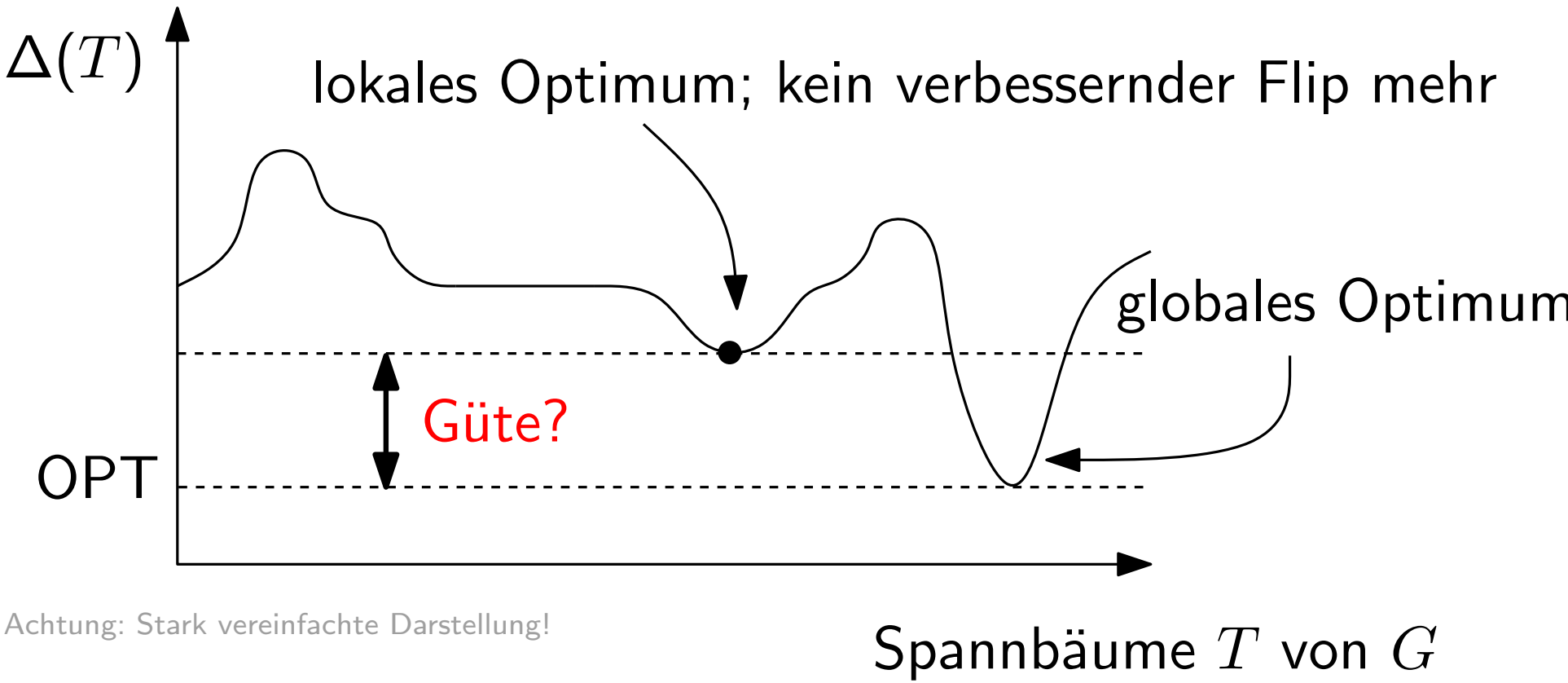
- Beginne mit beliebigem Spannbaum T von G
- führe Kantenflips durch, solange dadurch Verbesserung



Achtung: Stark vereinfachte Darstellung!

Lokale Suche

- Beginne mit beliebigem Spannbaum T von G
- führe Kantenflips durch, solange dadurch Verbesserung



Achtung: Stark vereinfachte Darstellung!

Terminierung trotz Plateaus?! Laufzeit?

Lokale Suche – Terminierung

MinDegSpanningTreeLocalSearch(T)
  while $\exists$ "verbessernder Flip"(*) in T für einen Knoten v
    mit $\deg_T(v) \geq \Delta(T) - \ell$ **do**
    führe den Flip durch

(*) d.h. $\exists uw \in E(G) \setminus E(T)$ mit
$\deg_T(v) - 1 > \max\{\deg_T(u), \deg_T(w)\}$,
so dass $T \cup \{uw\}$ einen Kreis durch v enthält.

Lokale Suche – Terminierung

MinDegSpanningTreeLocalSearch(T)
  while $\exists$ "verbessernder Flip"(*) in T für einen Knoten v
    mit $\deg_T(v) \geq \Delta(T) - \ell$ **do**
    führe den Flip durch

(*) d.h. $\exists uw \in E(G) \setminus E(T)$ mit
$\deg_T(v) - 1 > \max\{\deg_T(u), \deg_T(w)\}$,
so dass $T \cup \{uw\}$ einen Kreis durch v enthält.

- Flippe nur, falls Grad eines Knotens v mit
  $\deg(v) \geq \Delta(T) - \ell$ reduziert wird; $\ell := \lceil \log_2 n \rceil$.

Lokale Suche – Terminierung

MinDegSpanningTreeLocalSearch(T)
  while $\exists$ "verbessernder Flip"(*) in T für einen Knoten v
    mit $\deg_T(v) \geq \Delta(T) - \ell$ **do**
    führe den Flip durch

(*) d.h. $\exists uw \in E(G) \setminus E(T)$ mit
$\deg_T(v) - 1 > \max\{\deg_T(u), \deg_T(w)\}$,
so dass $T \cup \{uw\}$ einen Kreis durch v enthält.

- Flippe nur, falls Grad eines Knotens v mit $\deg(v) \geq \Delta(T) - \ell$ reduziert wird; $\ell := \lceil \log_2 n \rceil$.
- Unklar, ob in Polyzeit!
- Nachweis Effizienz später.

Güte

[Fürer & Raghvachari: SODA'92, JA'94]

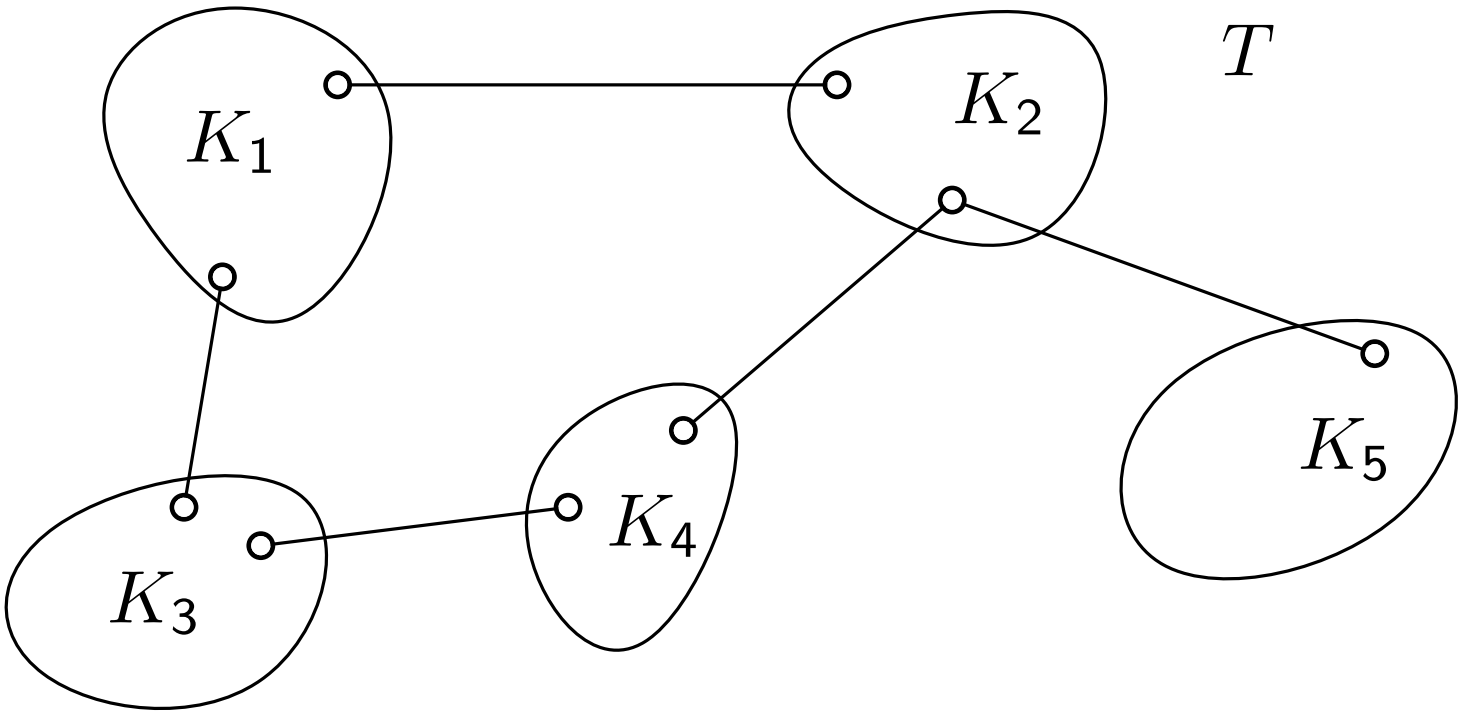
Satz. Sei T ein lokal optimaler Spannbaum.
Dann gilt $\Delta(T) \leq 2 \cdot \mathsf{OPT} + \ell$.

Güte

[Fürer & Raghvachari: SODA'92, JA'94]

Satz. Sei T ein lokal optimaler Spannbaum.
Dann gilt $\Delta(T) \leq 2 \cdot \mathrm{OPT} + \ell$.

Beweis. Untere Schranke für OPT



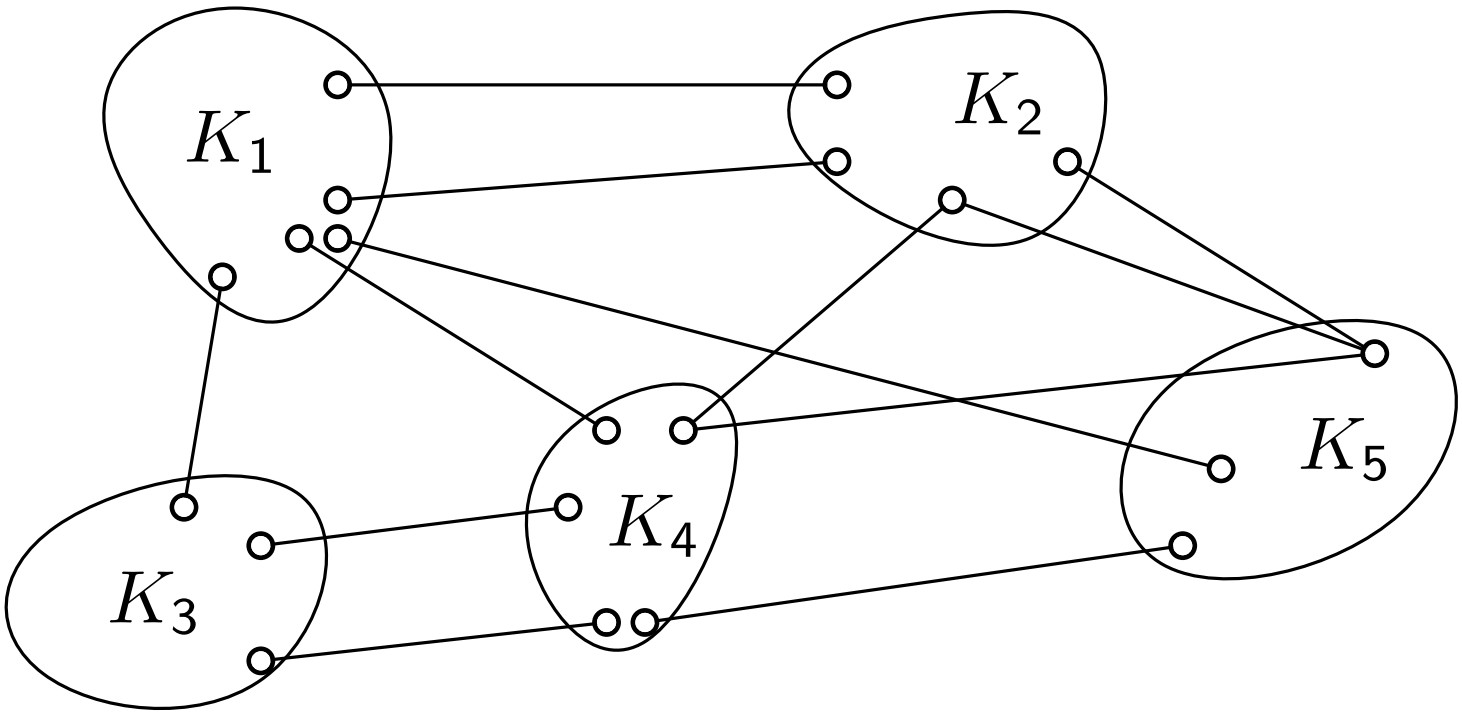
Entfernung von k Kanten zerlegt T in $k+1$ ZK.

Güte

[Fürer & Raghvachari: SODA'92, JA'94]

Satz. Sei T ein lokal optimaler Spannbaum.
Dann gilt $\Delta(T) \leq 2 \cdot \mathsf{OPT} + \ell$.

$E' :=$ Menge aller Kanten in G zw. verschiedenen ZK $K_i \neq K_j$.

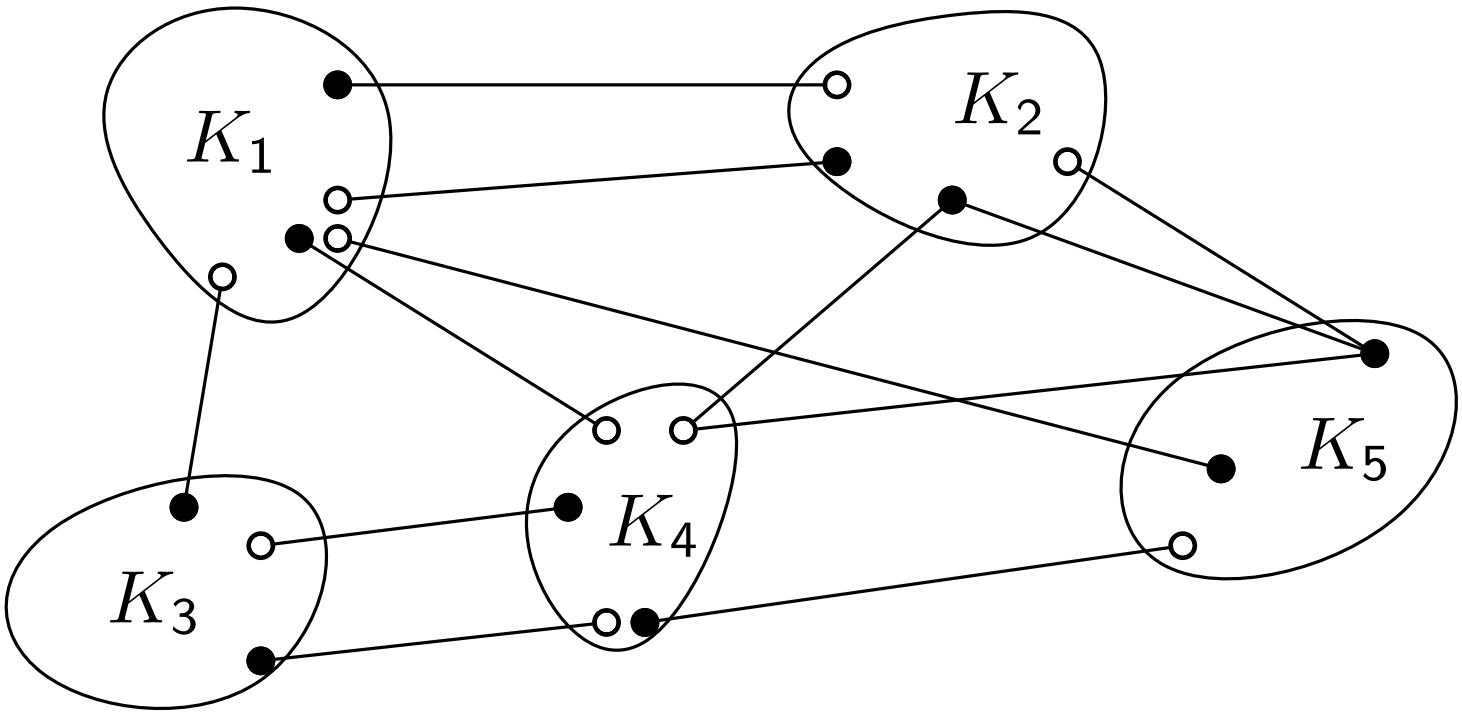


Güte

[Fürer & Raghvachari: SODA'92, JA'94]

Satz. Sei T ein lokal optimaler Spannbaum.
Dann gilt $\Delta(T) \leq 2 \cdot \mathsf{OPT} + \ell$.

$E' :=$ Menge aller Kanten in G zw. verschiedenen ZK $K_i \neq K_j$.



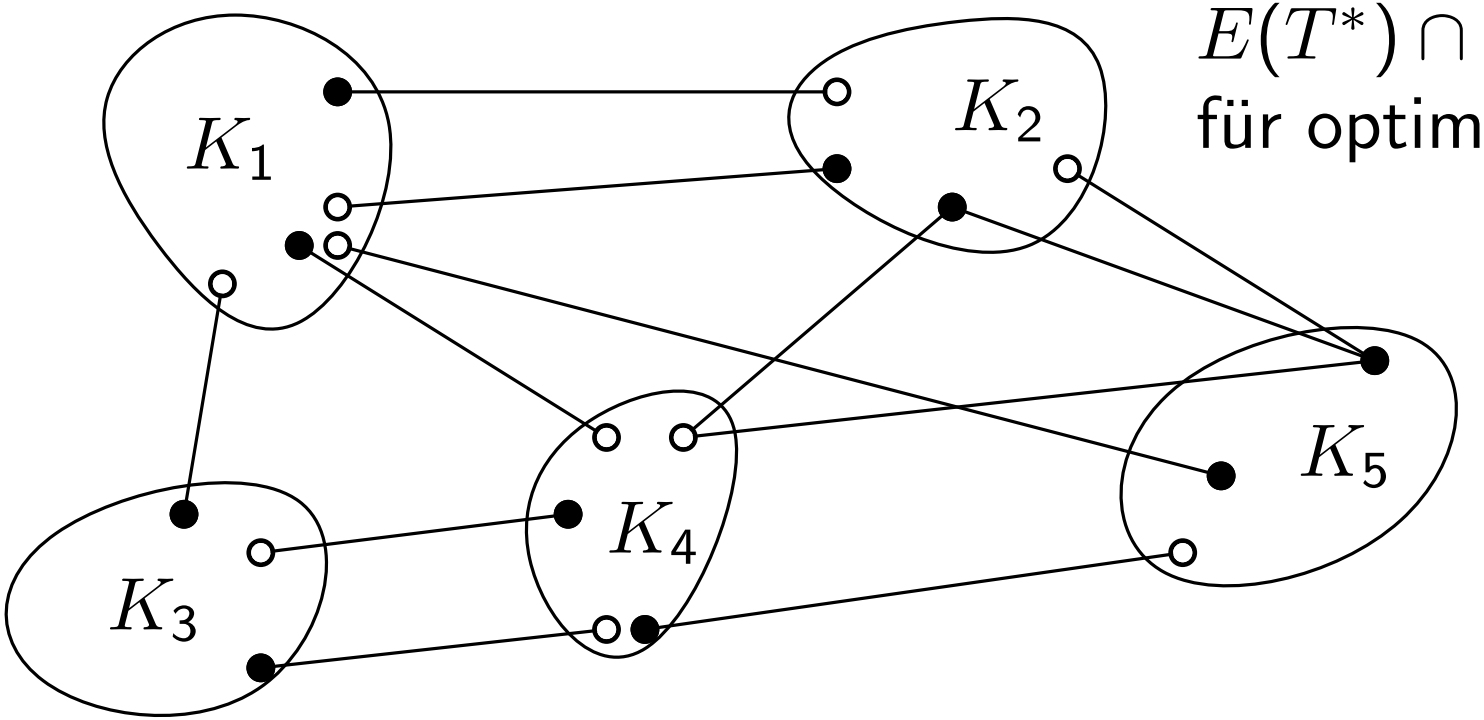
$S :=$ Knotenüberdeckung für E'.

Güte

[Fürer & Raghvachari: SODA'92, JA'94]

Satz. Sei T ein lokal optimaler Spannbaum.
Dann gilt $\Delta(T) \leq 2 \cdot \mathrm{OPT} + \ell$.

$E' :=$ Menge aller Kanten in G zw. verschiedenen ZK $K_i \neq K_j$.



$E(T^*) \cap E' \geq k$
für optimalen SB T^*

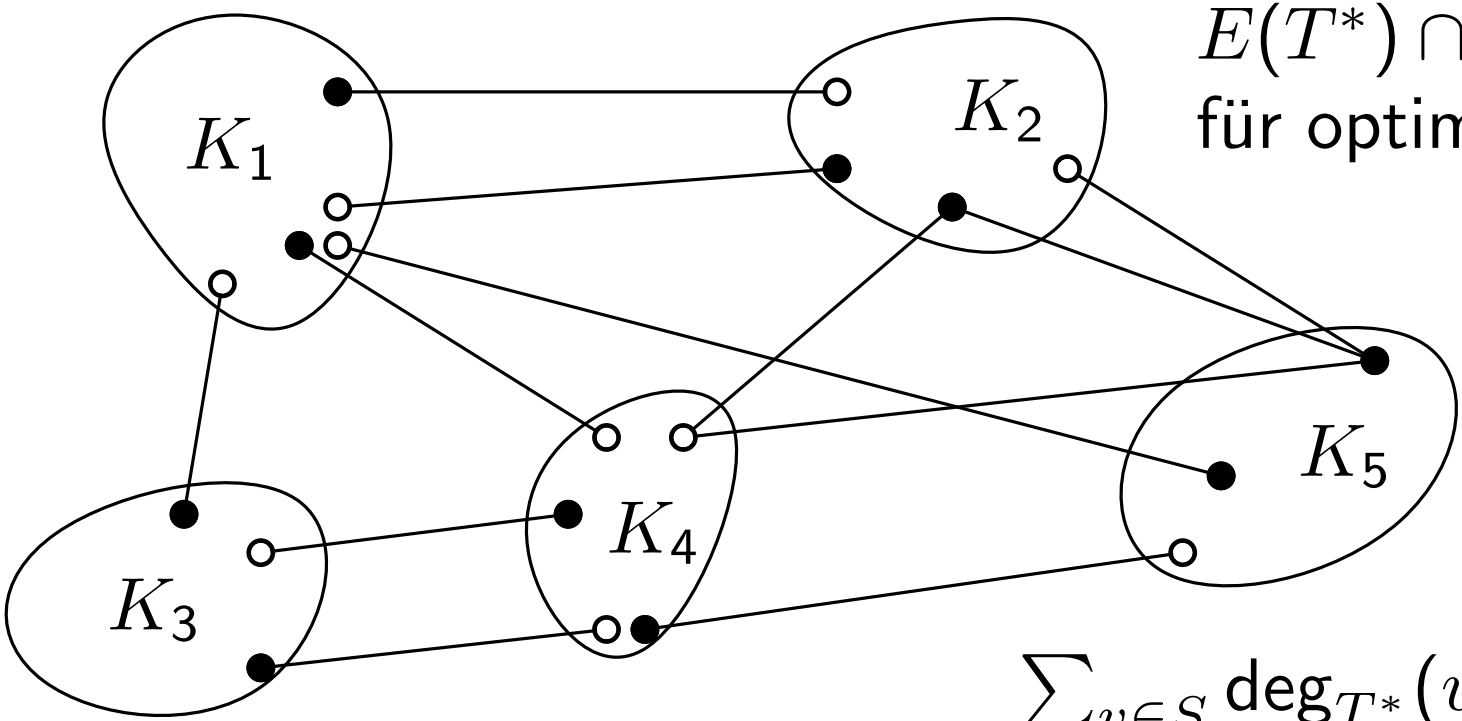
$S :=$ Knotenüberdeckung für E'.

Güte

[Fürer & Raghvachari: SODA'92, JA'94]

Satz. Sei T ein lokal optimaler Spannbaum.
Dann gilt $\Delta(T) \leq 2 \cdot \mathsf{OPT} + \ell$.

$E' :=$ Menge aller Kanten in G zw. verschiedenen ZK $K_i \neq K_j$.



$E(T^*) \cap E' \geq k$
für optimalen SB T^*

$\sum_{v \in S} \deg_{T^*}(v) \geq k$

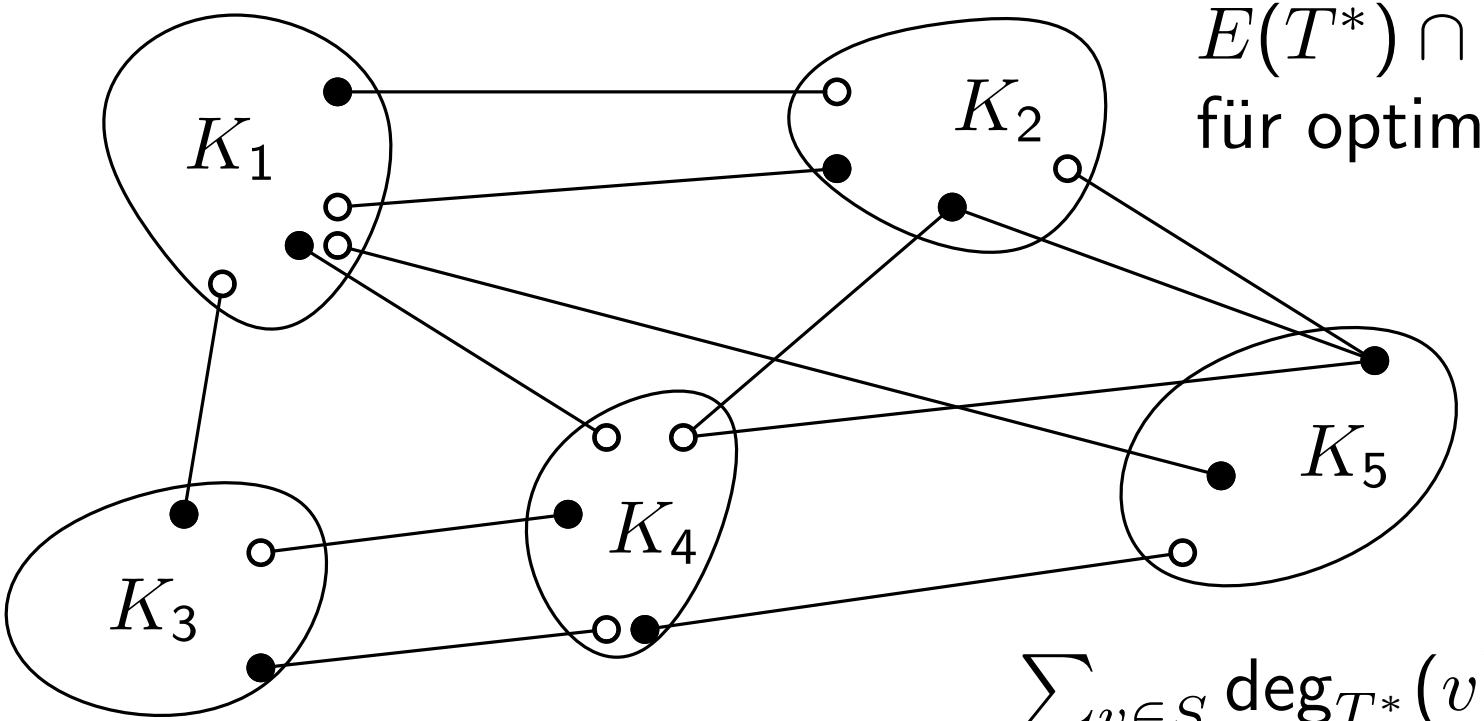
$S :=$ Knotenüberdeckung für E'.

Güte

[Fürer & Raghvachari: SODA'92, JA'94]

Satz. Sei T ein lokal optimaler Spannbaum.
Dann gilt $\Delta(T) \leq 2 \cdot \mathsf{OPT} + \ell$.

$E' :=$ Menge aller Kanten in G zw. verschiedenen ZK $K_i \neq K_j$.



$E(T^*) \cap E' \geq k$
für optimalen SB T^*

$\sum_{v \in S} \deg_{T^*}(v) \geq k$

$S :=$ Knotenüberdeckung für E'.

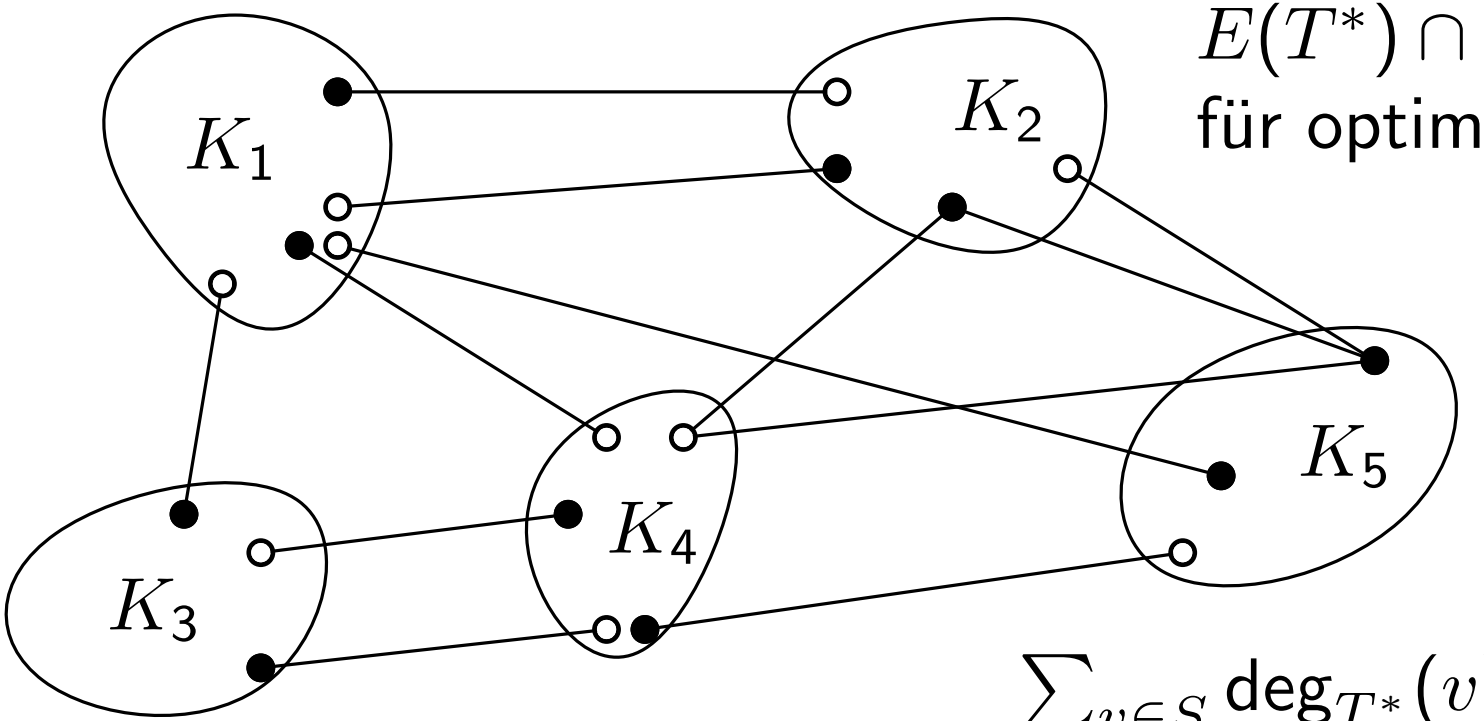
$\Rightarrow \mathsf{OPT} \geq$

Güte

[Fürer & Raghvachari: SODA'92, JA'94]

Satz. Sei T ein lokal optimaler Spannbaum.
Dann gilt $\Delta(T) \leq 2 \cdot \mathsf{OPT} + \ell$.

$E' :=$ Menge aller Kanten in G zw. verschiedenen ZK $K_i \neq K_j$.



$E(T^*) \cap E' \geq k$
für optimalen SB T^*

$\sum_{v \in S} \deg_{T^*}(v) \geq k$

$S :=$ Knotenüberdeckung für E'.

$\Rightarrow \mathsf{OPT} \geq k/|S|$

Approximation Factor

Thm. If T is a locally optimal spanning tree, then $\Delta(T) \leq 2 \cdot \mathrm{OPT} + \ell$, where $\ell = \lceil \log_2 n \rceil$.

Proof. **Part 1:** $\mathrm{OPT} \geq k/|S|$, where $k = |\text{removed edges}|$
Part 2: Applying the bound.

Approximation Factor

Thm. If T is a locally optimal spanning tree, then $\Delta(T) \leq 2 \cdot \mathrm{OPT} + \ell$, where $\ell = \lceil \log_2 n \rceil$.

Proof. **Part 1:** $\mathrm{OPT} \geq k/|S|$, where $k = |\text{removed edges}|$
Part 2: Applying the bound.

Let S_i be the nodes in T with $\deg_T(v) \geq i$.
Let E_i be the edges of T incident to S_i.

Approximation Factor

Thm. If T is a locally optimal spanning tree, then $\Delta(T) \leq 2 \cdot \mathrm{OPT} + \ell$, where $\ell = \lceil \log_2 n \rceil$.

Proof. **Part 1:** $\mathrm{OPT} \geq k/|S|$, where $k = |\text{removed edges}|$
Part 2: Applying the bound.

Let S_i be the nodes in T with $\deg_T(v) \geq i$.
Let E_i be the edges of T incident to S_i.

$\Rightarrow S_1 \supseteq S_2 \supseteq \dots$
$\Rightarrow S_1 = V(G)$

Approximation Factor

Thm. If T is a locally optimal spanning tree, then $\Delta(T) \leq 2 \cdot \mathrm{OPT} + \ell$, where $\ell = \lceil \log_2 n \rceil$.

Proof. **Part 1:** $\mathrm{OPT} \geq k/|S|$, where $k = |\text{removed edges}|$
Part 2: Applying the bound.

Let S_i be the nodes in T with $\deg_T(v) \geq i$.
Let E_i be the edges of T incident to S_i.

$\Rightarrow S_1 \supseteq S_2 \supseteq \dots$
$\Rightarrow S_1 = V(G)$
$\Rightarrow E_1 = E(T)$

Approximation Factor

Thm. If T is a locally optimal spanning tree, then $\Delta(T) \leq 2 \cdot \mathrm{OPT} + \ell$, where $\ell = \lceil \log_2 n \rceil$.

Proof. **Part 1:** $\mathrm{OPT} \geq k/|S|$, where $k = |\text{removed edges}|$
Part 2: Applying the bound.

Let S_i be the nodes in T with $\deg_T(v) \geq i$.
Let E_i be the edges of T incident to S_i.

$\Rightarrow S_1 \supseteq S_2 \supseteq \dots$
$\Rightarrow S_1 = V(G)$
$\Rightarrow E_1 = E(T)$

Claim 1: For $i \geq \Delta(T) - \ell + 1$,
(i) $|E_i| \geq (i-1)|S_i| + 1$,
(ii) Each $e \in E(G) \setminus E_i$ connecting distinct components of $T \setminus E_i$ is incident to a node of S_{i-1}.

Approximation Factor

Thm. If T is a locally optimal spanning tree, then $\Delta(T) \leq 2 \cdot \mathrm{OPT} + \ell$, where $\ell = \lceil \log_2 n \rceil$.

Proof. **Part 1:** $\mathrm{OPT} \geq k/|S|$, where $k = |\text{removed edges}|$
Part 2: Applying the bound.

Let S_i be the nodes in T with $\deg_T(v) \geq i$.
Let E_i be the edges of T incident to S_i.

$\Rightarrow S_1 \supseteq S_2 \supseteq \dots$
$\Rightarrow S_1 = V(G)$
$\Rightarrow E_1 = E(T)$

Claim 1: For $i \geq \Delta(T) - \ell + 1$,
(i) $|E_i| \geq (i-1)|S_i| + 1$,
(ii) Each $e \in E(G) \setminus E_i$ connecting distinct components of $T \setminus E_i$ is incident to a node of S_{i-1}.

Claim 2: There is an $i \geq \Delta(T) - \ell + 1$ such that $|S_{i-1}| \leq 2|S_i|$.

Approximation Factor

Thm. If T is a locally optimal spanning tree, then $\Delta(T) \leq 2 \cdot \mathrm{OPT} + \ell$, where $\ell = \lceil \log_2 n \rceil$.

Proof. **Part 1:** $\mathrm{OPT} \geq k/|S|$, where $k = |\text{removed edges}|$
Part 2: Applying the bound.

Let S_i be the nodes in T with $\deg_T(v) \geq i$.
Let E_i be the edges of T incident to S_i.

$\Rightarrow S_1 \supseteq S_2 \supseteq \dots$
$\Rightarrow S_1 = V(G)$
$\Rightarrow E_1 = E(T)$

Claim 1: For $i \geq \Delta(T) - \ell + 1$,
(i) $|E_i| \geq (i-1)|S_i| + 1$,
(ii) Each $e \in E(G) \setminus E_i$ connecting distinct components of $T \setminus E_i$ is incident to a node of S_{i-1}.

Claim 2: There is an $i \geq \Delta(T) - \ell + 1$ such that $|S_{i-1}| \leq 2|S_i|$.

By Part 1, and Claims 1 & 2 ... how do we choose k and S?

Approximation Factor

Thm. If T is a locally optimal spanning tree, then $\Delta(T) \leq 2 \cdot \mathrm{OPT} + \ell$, where $\ell = \lceil \log_2 n \rceil$.

Proof. **Part 1:** $\mathrm{OPT} \geq k/|S|$, where $k = |\text{removed edges}|$
Part 2: Applying the bound.

Let S_i be the nodes in T with $\deg_T(v) \geq i$.
Let E_i be the edges of T incident to S_i.

$\Rightarrow S_1 \supseteq S_2 \supseteq \dots$
$\Rightarrow S_1 = V(G)$
$\Rightarrow E_1 = E(T)$

Claim 1: For $i \geq \Delta(T) - \ell + 1$,
(i) $|E_i| \geq (i-1)|S_i| + 1$,
(ii) Each $e \in E(G) \setminus E_i$ connecting distinct components of $T \setminus E_i$ is incident to a node of S_{i-1}.

Claim 2: There is an $i \geq \Delta(T) - \ell + 1$ such that $|S_{i-1}| \leq 2|S_i|$.

By Part 1, and Claims 1 & 2 ... how do we choose k and S?

$$\mathrm{OPT} \geq \frac{k}{|S|} \geq \frac{(i-1)|S_i|+1}{|S_{i-1}|} \geq \frac{(i-1)|S_i|+1}{2|S_i|} > \frac{(i-1)}{2} \geq \frac{(\Delta(T)-\ell)}{2}$$

Claim 1 Claim 2

□

Runtime

Thm. The algorithm finds a locally optimal tree efficiently.

Proof.

Runtime

Thm. The algorithm finds a locally optimal tree efficiently.

Proof.

Via potential function $\Phi(G, T)$, that is, a function measuring the value of a solution where (hopefully):

- each iteration decreases the potential of a solution.

Runtime

Thm. The algorithm finds a locally optimal tree efficiently.

Proof.

Via potential function $\Phi(G, T)$, that is, a function measuring the value of a solution where (hopefully):

- each iteration decreases the potential of a solution.

- the function is bounded both from above and below.

Runtime

Thm. The algorithm finds a locally optimal tree efficiently.

Proof.

Via potential function $\Phi(G, T)$, that is, a function measuring the value of a solution where (hopefully):

- each iteration decreases the potential of a solution.
- the function is bounded both from above and below.
- executing $f(n)$ iterations would exceed this lower bound.

Runtime

Thm. The algorithm finds a locally optimal tree efficiently.

Proof. Our potential function: $\Phi(T) = \sum_{v \in V(G)} 3^{\deg_T(v)}$

Via potential function $\Phi(G, T)$, that is, a function measuring the value of a solution where (hopefully):

- each iteration decreases the potential of a solution.
- the function is bounded both from above and below.
- executing $f(n)$ iterations would exceed this lower bound.

Runtime

Thm. The algorithm finds a locally optimal tree efficiently.

Proof. Our potential function: $\Phi(T) = \sum_{v \in V(G)} 3^{\deg_T(v)}$

Via potential function $\Phi(G, T)$, that is, a function measuring the value of a solution where (hopefully):

- each iteration decreases the potential of a solution.

Lemma: After each flip $T \to T'$, $\Phi(T') \leq (1 - \frac{2}{27n^3})\Phi(T)$.

- the function is bounded both from above and below.

- executing $f(n)$ iterations would exceed this lower bound.

Runtime

Thm. The algorithm finds a locally optimal tree efficiently.

Proof. Our potential function: $\Phi(T) = \sum_{v \in V(G)} 3^{\deg_T(v)}$

Via potential function $\Phi(G, T)$, that is, a function measuring the value of a solution where (hopefully):

- each iteration decreases the potential of a solution.

Lemma: After each flip $T \to T'$, $\Phi(T') \leq (1 - \frac{2}{27n^3})\Phi(T)$.

- the function is bounded both from above and below.

For any spanning tree T, $\Phi(T) \in [3n, n3^n]$.

- executing $f(n)$ iterations would exceed this lower bound.

Runtime

Thm. The algorithm finds a locally optimal tree efficiently.

Proof. Our potential function: $\Phi(T) = \sum_{v \in V(G)} 3^{\deg_T(v)}$

Via potential function $\Phi(G, T)$, that is, a function measuring the value of a solution where (hopefully):

- each iteration decreases the potential of a solution.

Lemma: After each flip $T \to T'$, $\Phi(T') \leq (1 - \frac{2}{27n^3})\Phi(T)$.

- the function is bounded both from above and below.

For any spanning tree T, $\Phi(T) \in [3n, n3^n]$.

- executing $f(n)$ iterations would exceed this lower bound.

How does $\Phi(T)$ change?

Runtime

Thm. The algorithm finds a locally optimal tree efficiently.

Proof. Our potential function: $\Phi(T) = \sum_{v \in V(G)} 3^{\deg_T(v)}$

Via potential function $\Phi(G, T)$, that is, a function measuring the value of a solution where (hopefully):

- each iteration decreases the potential of a solution.

Lemma: After each flip $T \to T'$, $\Phi(T') \leq (1 - \frac{2}{27n^3})\Phi(T)$.

- the function is bounded both from above and below.

For any spanning tree T, $\Phi(T) \in [3n, n3^n]$.

- executing $f(n)$ iterations would exceed this lower bound.

How does $\Phi(T)$ change?

shrinks by: $(1 - \frac{2}{27n^3})^{f(n)} \leq (e^{-\frac{2}{27n^3}})^{f(n)}$

Runtime

Thm. The algorithm finds a locally optimal tree efficiently.

Proof. Our potential function: $\Phi(T) = \sum_{v \in V(G)} 3^{\deg_T(v)}$

Via potential function $\Phi(G, T)$, that is, a function measuring the value of a solution where (hopefully):

- each iteration decreases the potential of a solution.

Lemma: After each flip $T \to T'$, $\Phi(T') \leq (1 - \frac{2}{27n^3})\Phi(T)$.

- the function is bounded both from above and below.

For any spanning tree T, $\Phi(T) \in [3n, n3^n]$.

- executing $f(n)$ iterations would exceed this lower bound.

How does $\Phi(T)$ change?

shrinks by: $(1 - \frac{2}{27n^3})^{f(n)} \leq (e^{-\frac{2}{27n^3}})^{f(n)}$

Goal: After $f(n)$ iterations $\Phi(T) = n < 3n$

Runtime

Thm. The algorithm finds a locally optimal tree efficiently.

Proof. Our potential function: $\Phi(T) = \sum_{v \in V(G)} 3^{\deg_T(v)}$

Via potential function $\Phi(G, T)$, that is, a function measuring the value of a solution where (hopefully):

- each iteration decreases the potential of a solution.

Lemma: After each flip $T \to T'$, $\Phi(T') \leq (1 - \frac{2}{27n^3})\Phi(T)$.

- the function is bounded both from above and below.

For any spanning tree T, $\Phi(T) \in [3n, n3^n]$.

- executing $f(n)$ iterations would exceed this lower bound.

Let $f(n) = \frac{27}{2} n^4 \cdot \ln 3$. How does $\Phi(T)$ change?

shrinks by: $(1 - \frac{2}{27n^3})^{f(n)} \leq (e^{-\frac{2}{27n^3}})^{f(n)}$

Goal: After $f(n)$ iterations $\Phi(T) = n < 3n$

Runtime

Thm. The algorithm finds a locally optimal tree efficiently.

Proof. Our potential function: $\Phi(T) = \sum_{v \in V(G)} 3^{\deg_T(v)}$

Via potential function $\Phi(G, T)$, that is, a function measuring the value of a solution where (hopefully):

- each iteration decreases the potential of a solution.

Lemma: After each flip $T \to T'$, $\Phi(T') \leq (1 - \frac{2}{27n^3})\Phi(T)$.

- the function is bounded both from above and below.

For any spanning tree T, $\Phi(T) \in [3n, n3^n]$.

- executing $f(n)$ iterations would exceed this lower bound.

Let $f(n) = \frac{27}{2} n^4 \cdot \ln 3$. How does $\Phi(T)$ change?

shrinks by: $(1 - \frac{2}{27n^3})^{f(n)} \leq (e^{-\frac{2}{27n^3}})^{f(n)} = e^{-n \ln 3} = 3^{-n}$

Goal: After $f(n)$ iterations $\Phi(T) = n < 3n$ $\square$

Extensions

Cor. For any constant $b > 1$, and $\ell = \lceil \log_b n \rceil$, the local search algorithm runs in polynomial time and produces a spanning tree T with $\Delta(T) \leq b \cdot \mathrm{OPT} + \lceil \log_b n \rceil$.

Extensions

Cor. For any constant $b > 1$, and $\ell = \lceil \log_b n \rceil$, the local search algorithm runs in polynomial time and produces a spanning tree T with $\Delta(T) \leq b \cdot \mathrm{OPT} + \lceil \log_b n \rceil$.

Proof. Similar to before. □

Extensions

Cor. For any constant $b > 1$, and $\ell = \lceil \log_b n \rceil$, the local search algorithm runs in polynomial time and produces a spanning tree T with $\Delta(T) \leq b \cdot \mathsf{OPT} + \lceil \log_b n \rceil$.

Proof. Similar to before. □

Thm. There is a local search algorithm that runs in $O(EV\alpha(E, V) \log V)$ time and produces a spanning tree T with $\Delta(T) \leq \mathsf{OPT} + 1$.