Aalto University
School of Science

Department of Computer Science

Combinatorics of Efficient Computations

# Approximation Algorithms

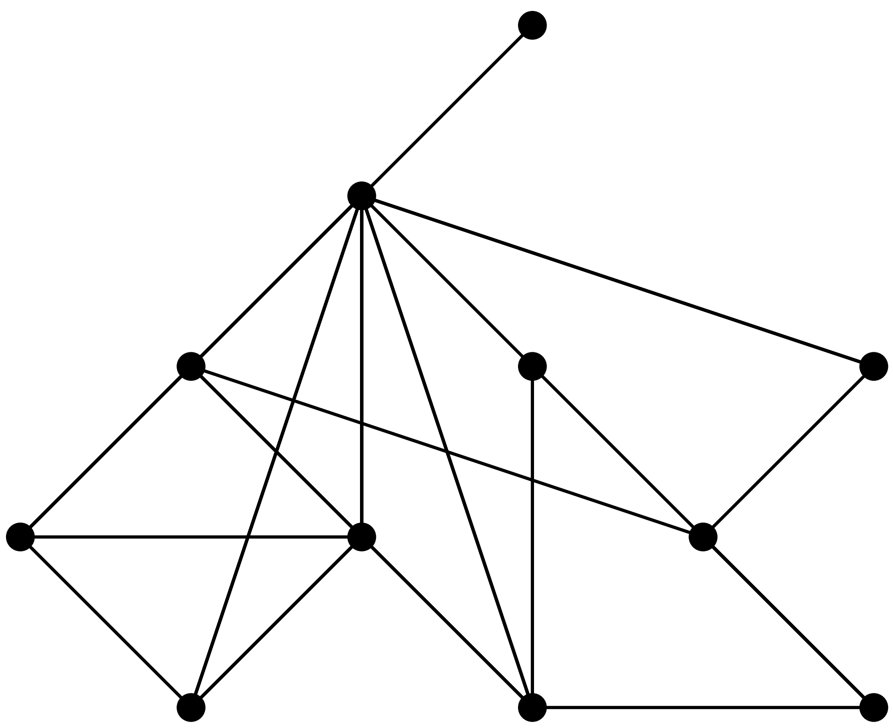
## Lecture 7: Min. Degree Spanning Trees via Local Search

Joachim Spoerhase

2019

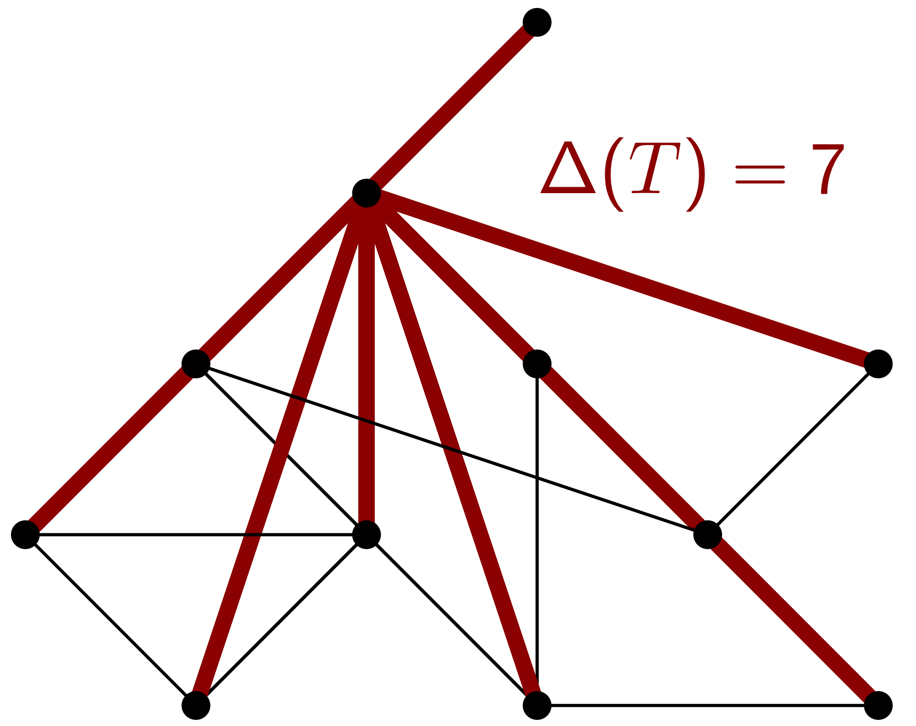
# MinimumDegreeSpanningTree

**Given**: A connected graph Graph $G = (V, E)$.
**Find**: A spanning tree $T$ which has the minimum maximum degree $\Delta(T)$ among all spanning trees of G.



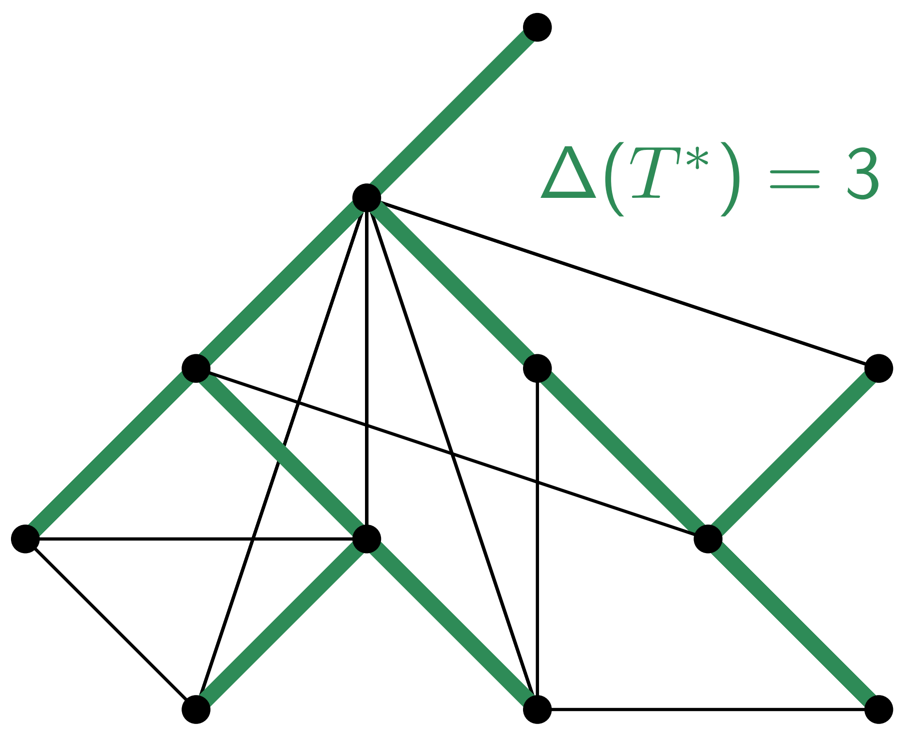
# MinimumDegreeSpanningTree

**Given**: A connected graph Graph $G = (V, E)$.
**Find**: A spanning tree $T$ which has the minimum maximum degree $\Delta(T)$ among all spanning trees of G.



# MinimumDegreeSpanningTree

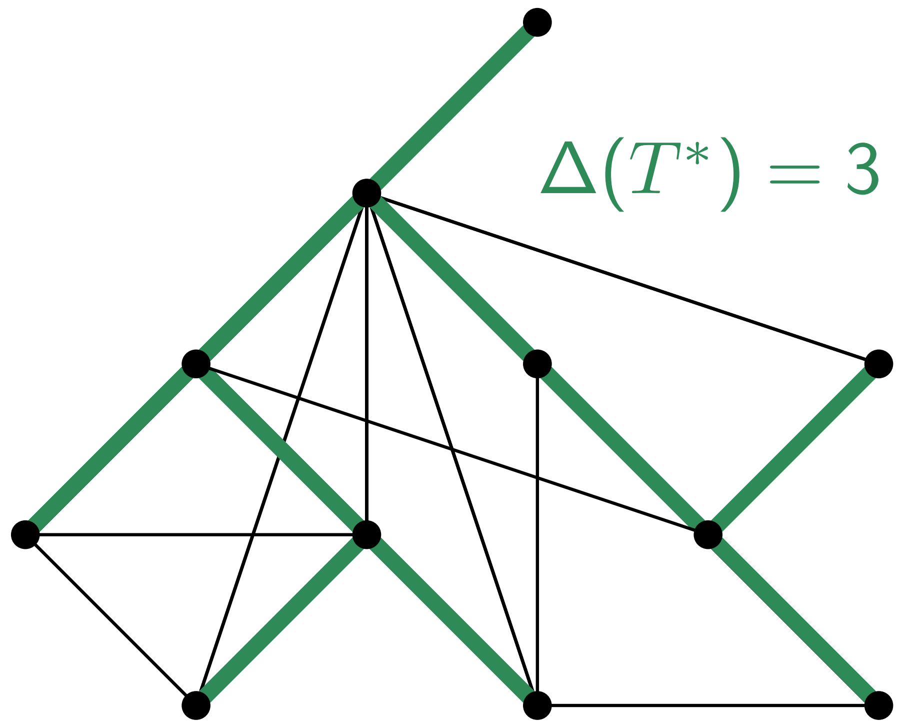
**Given**: A connected graph Graph $G = (V, E)$.
**Find**: A spanning tree $T$ which has the minimum maximum degree $\Delta(T)$ among all spanning trees of G.



# MinimumDegreeSpanningTree

**Given**: A connected graph Graph $G = (V, E)$.
**Find**: A spanning tree $T$ which has the minimum maximum degree $\Delta(T)$ among all spanning trees of G.

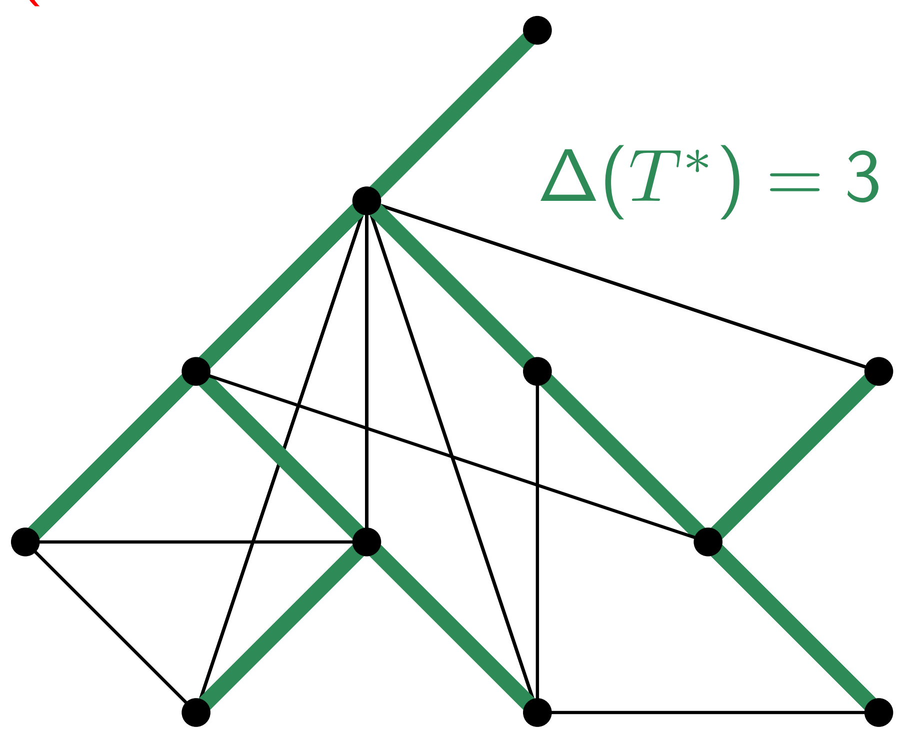
NP-hard :-(



# MinimumDegreeSpanningTree

**Given**: A connected graph Graph $G = (V, E)$.
**Find**: A spanning tree $T$ which has the minimum maximum degree $\Delta(T)$ among all spanning trees of G.

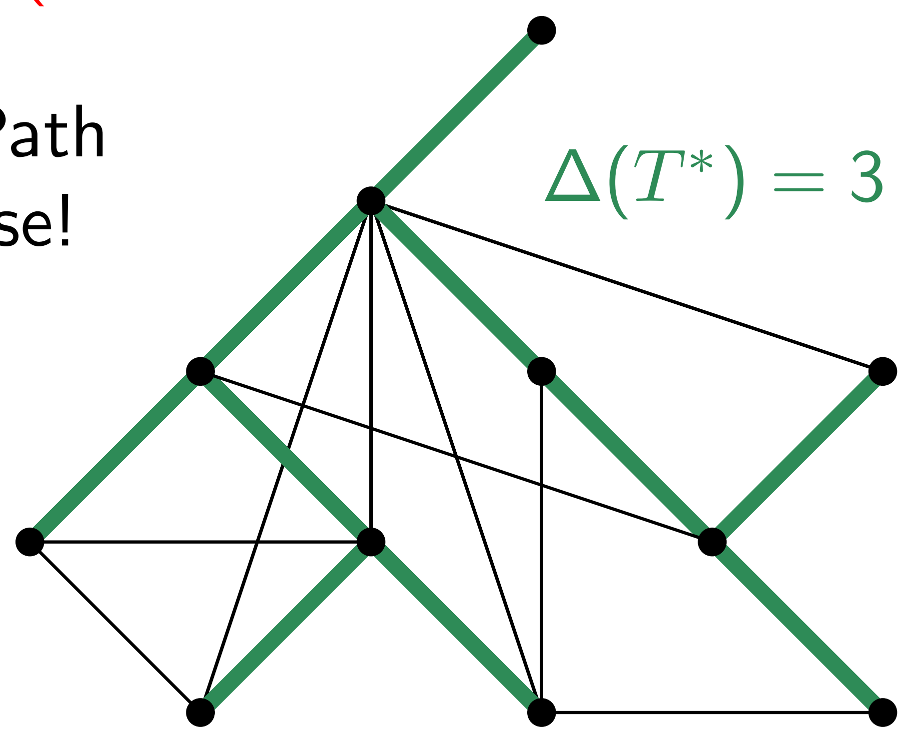
NP-hard :-(
Why?



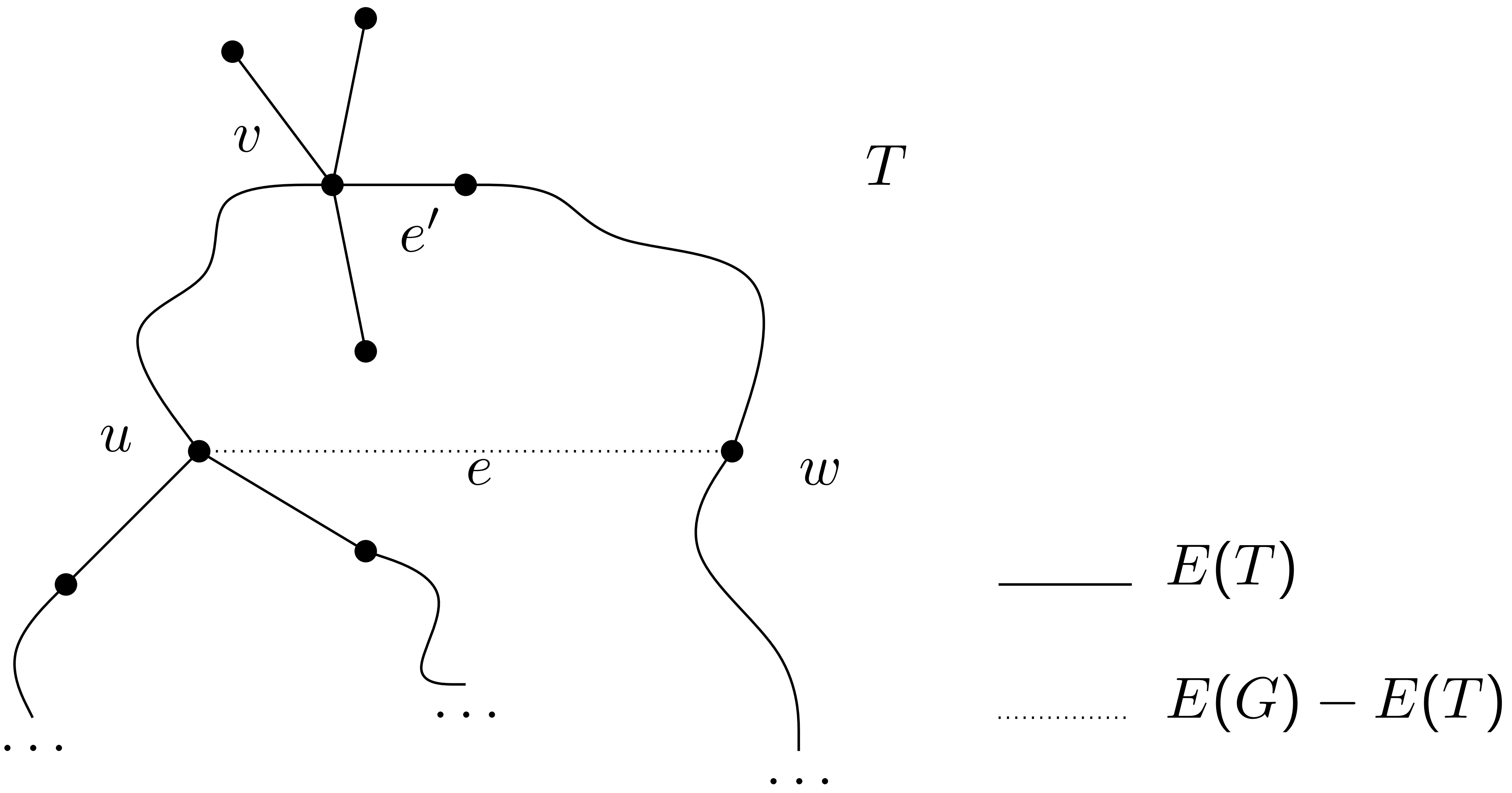
# MinimumDegreeSpanningTree

**Given**: A connected graph Graph $G = (V, E)$.
**Find**: A spanning tree $T$ which has the minimum maximum degree $\Delta(T)$ among all spanning trees of G.

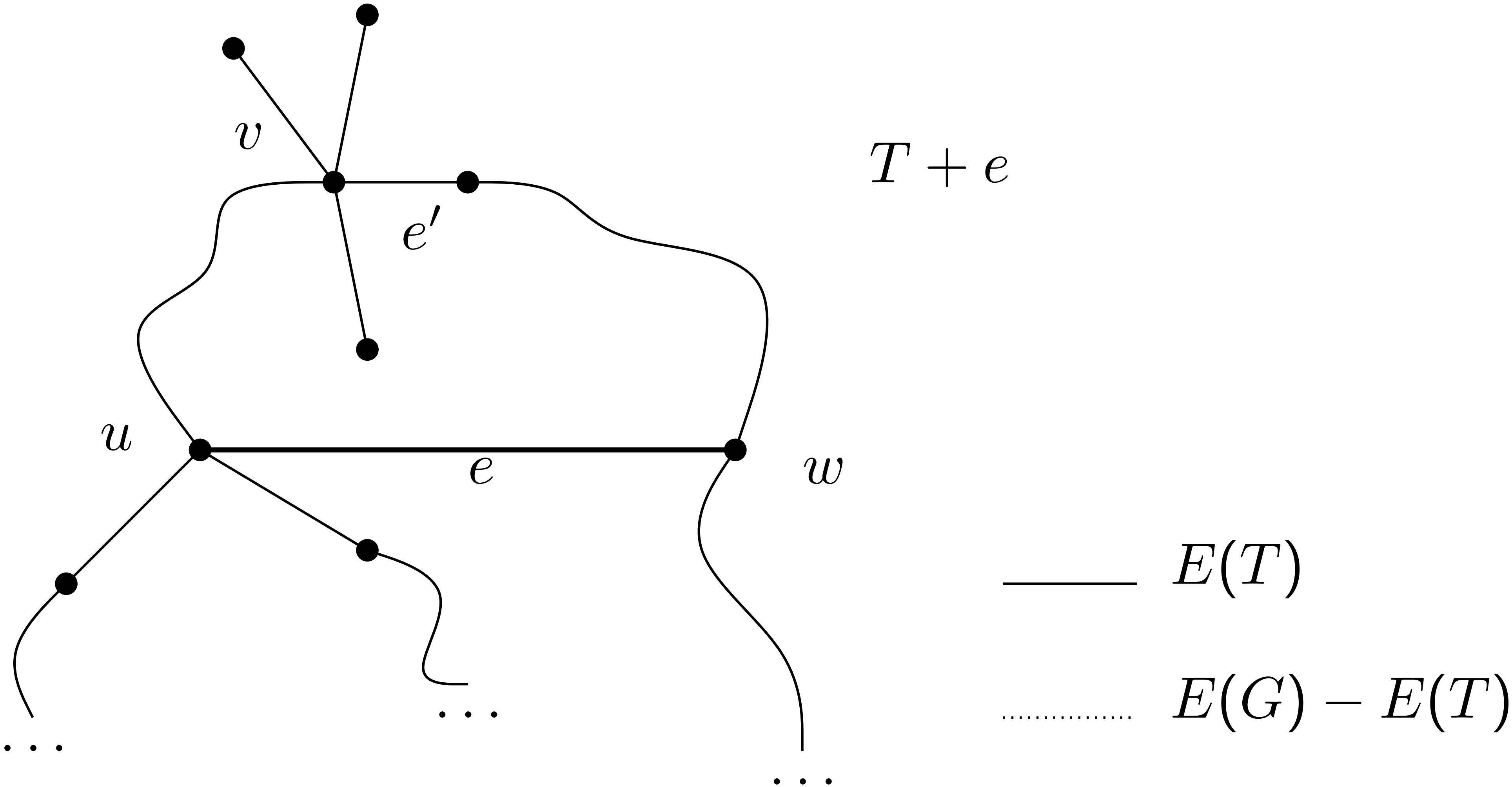
NP-hard :-(
Why?
Hamiltonian Path
is a special case!



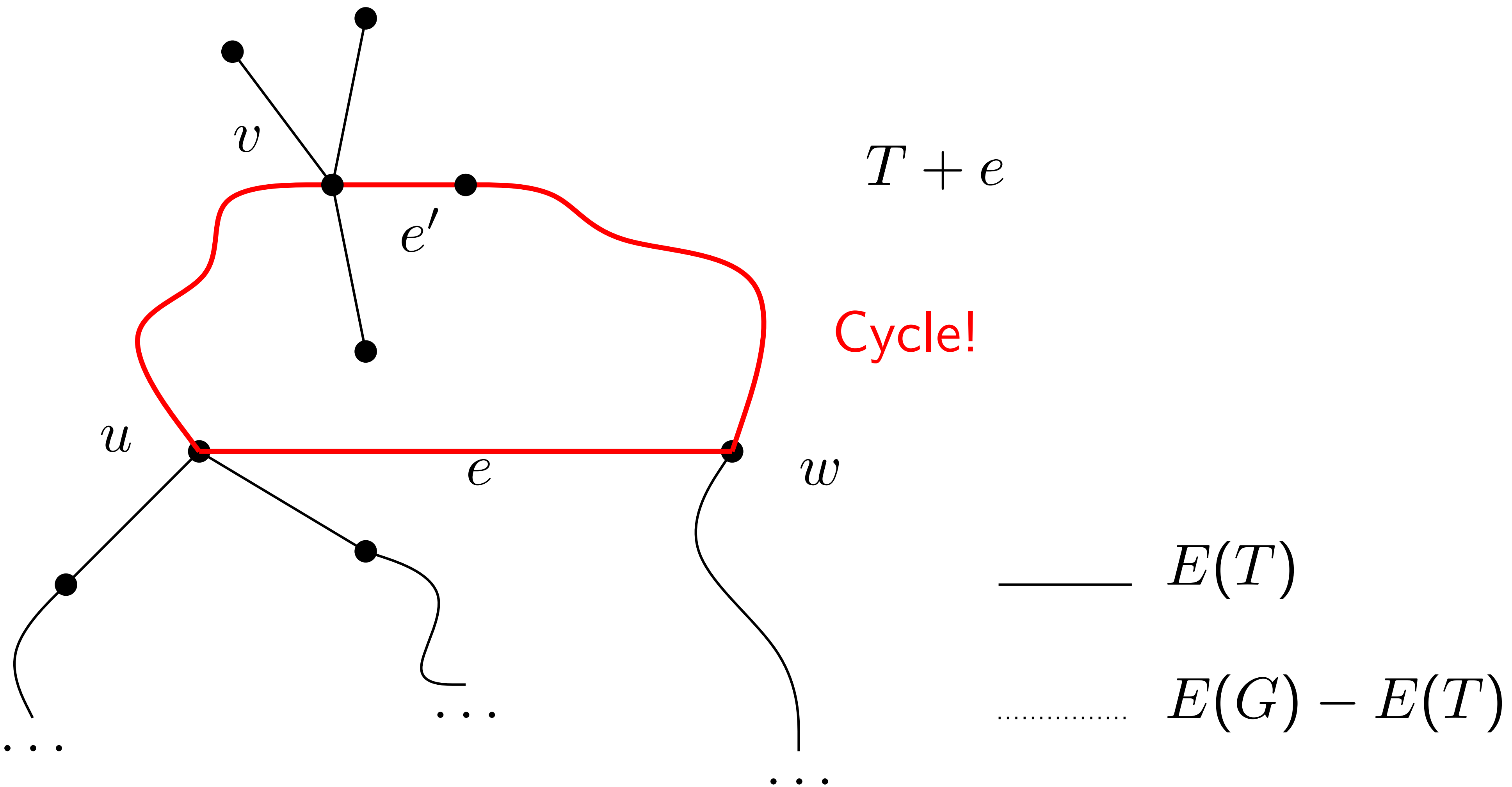
# Local Adjustment via Edge Flips



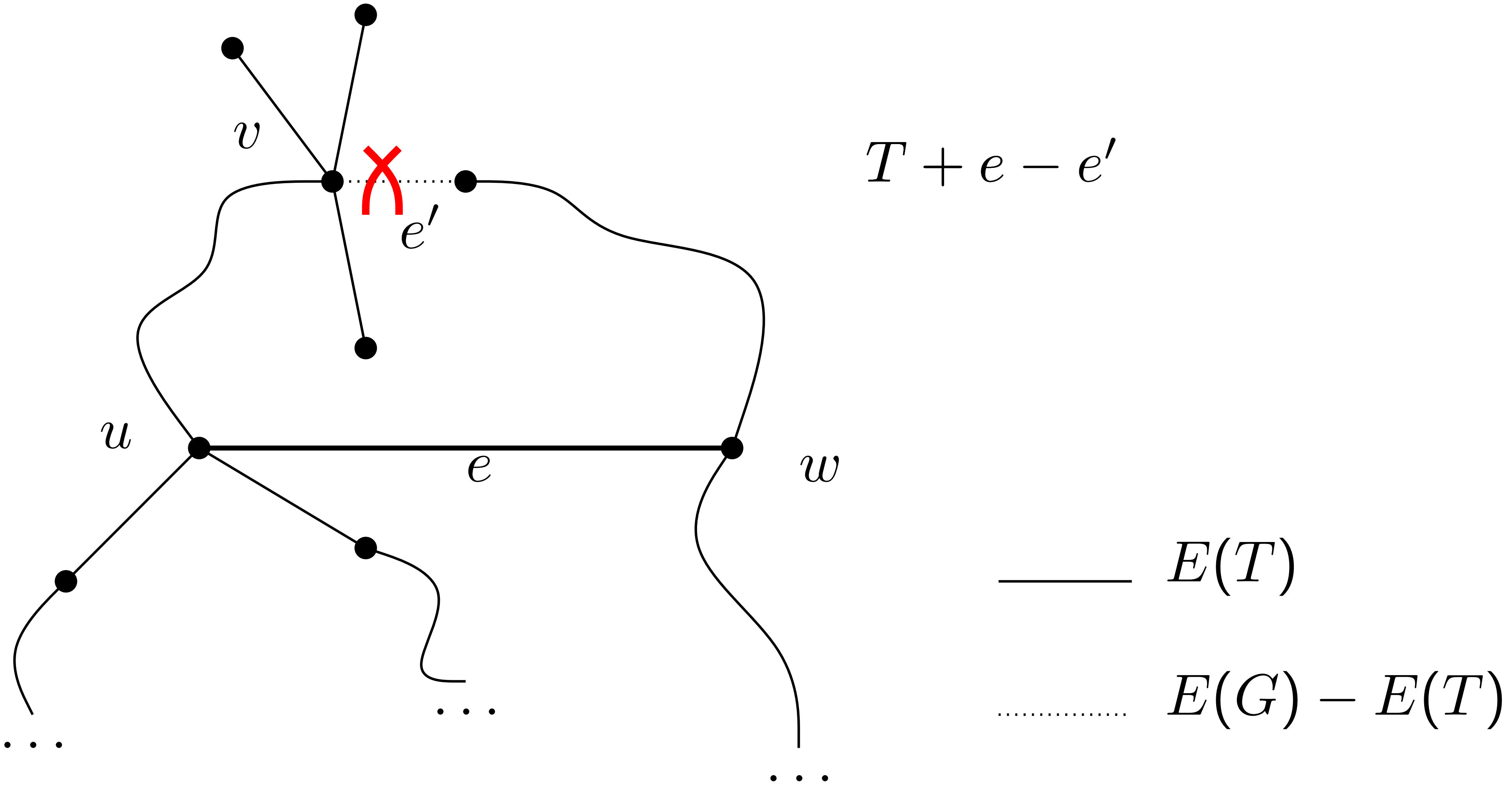
# Local Adjustment via Edge Flips



# Local Adjustment via Edge Flips

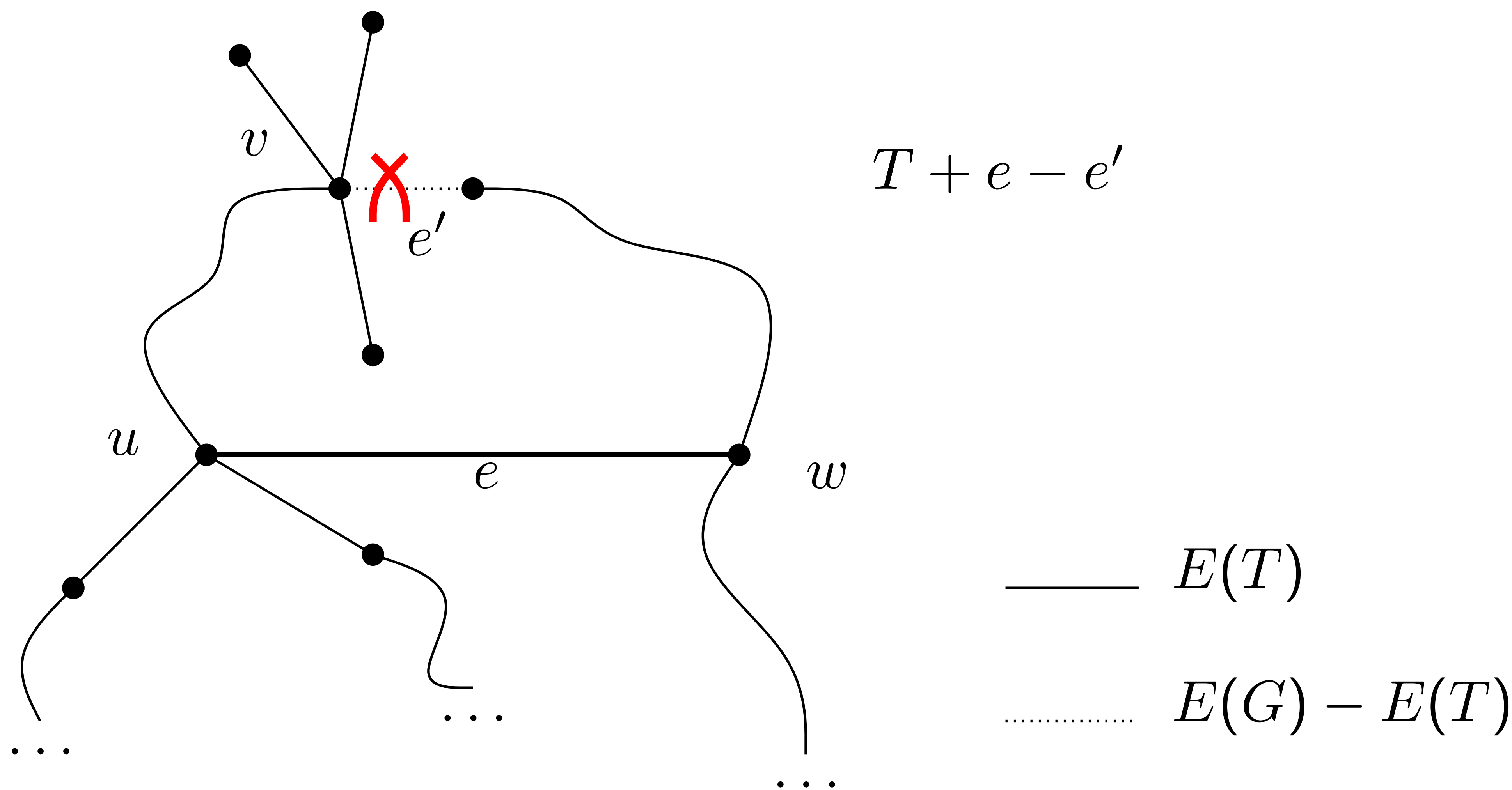


# Local Adjustment via Edge Flips



# Local Adjustment via Edge Flips

Improvement when $\deg_T(v) - 1 > \max\{\deg_T(u), \deg_T(w)\}$

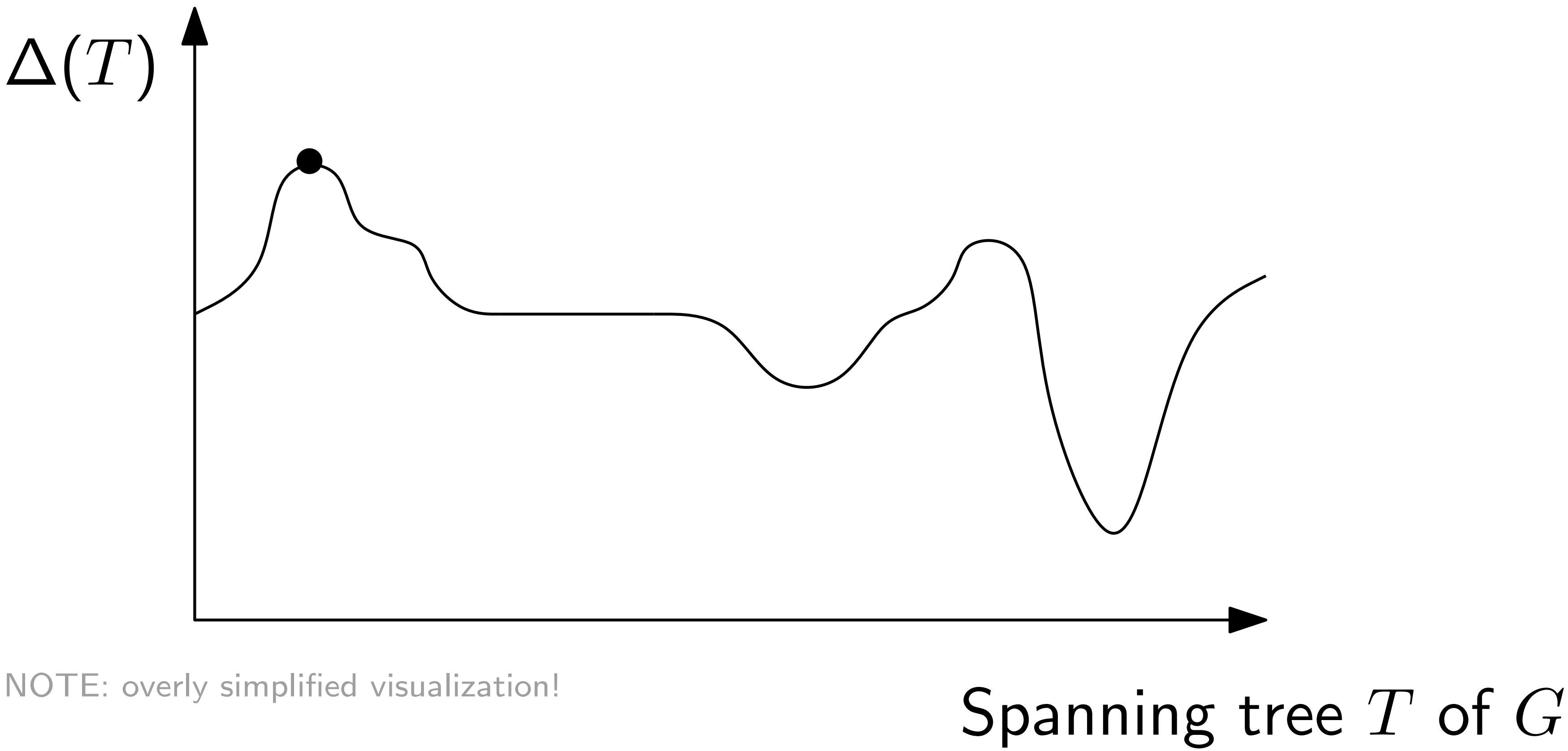


# Local Search

- Start from any spanning tree $T$ of $G$

# Local Search

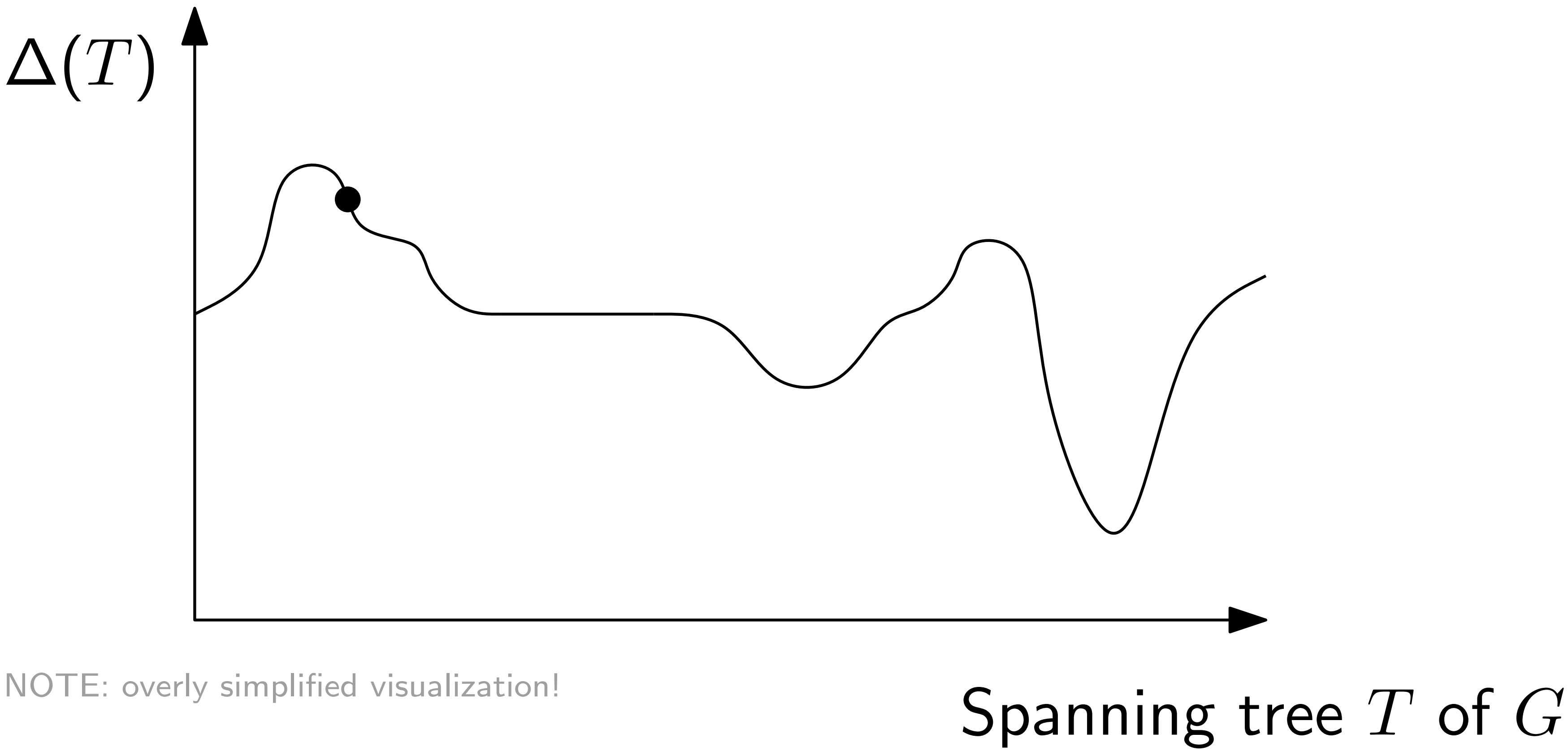
- Start from any spanning tree $T$ of $G$
- Perform edge flips until no flip improves the solution.




NOTE: overly simplified visualization!

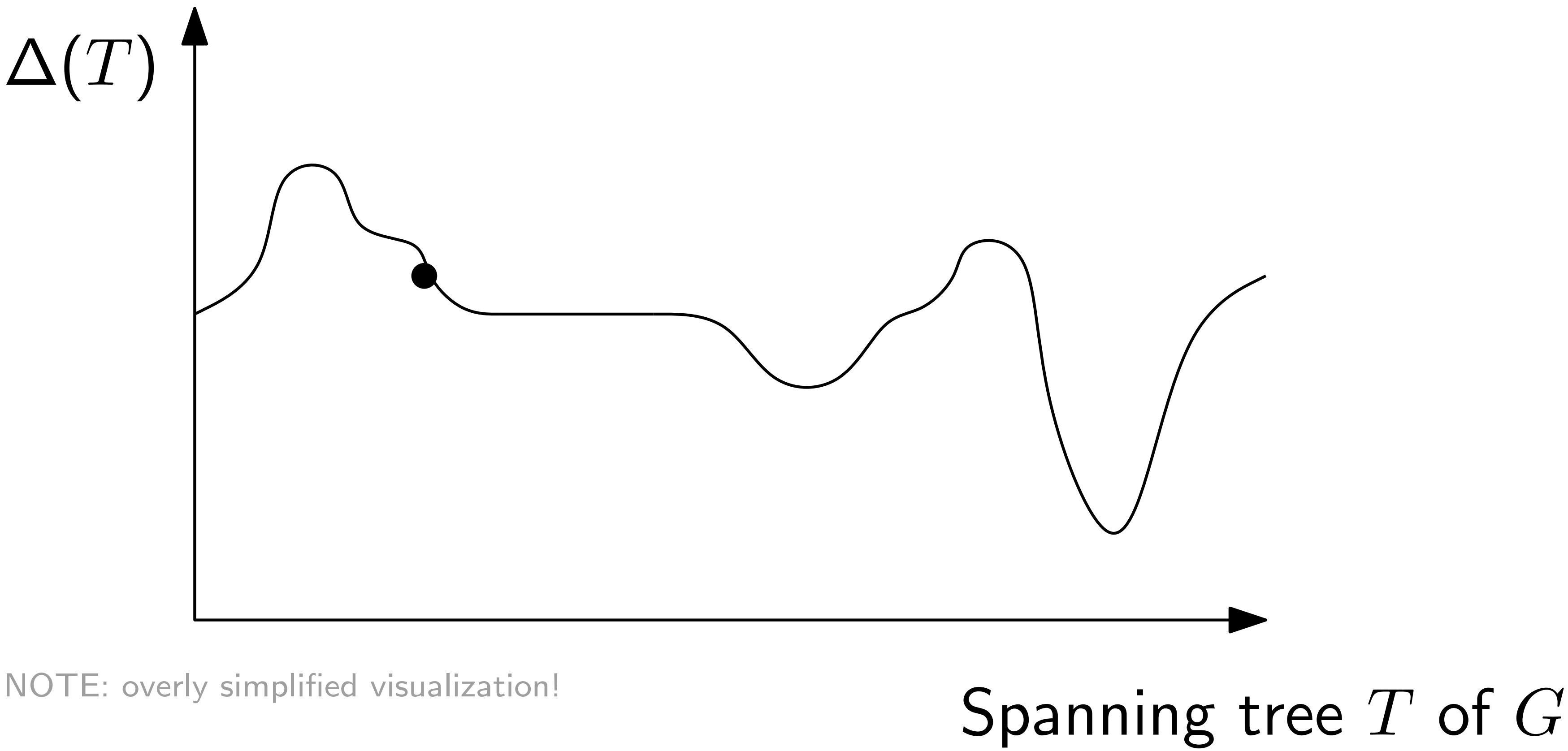
# Local Search

- Start from any spanning tree $T$ of $G$
- Perform edge flips until no flip improves the solution.



# Local Search

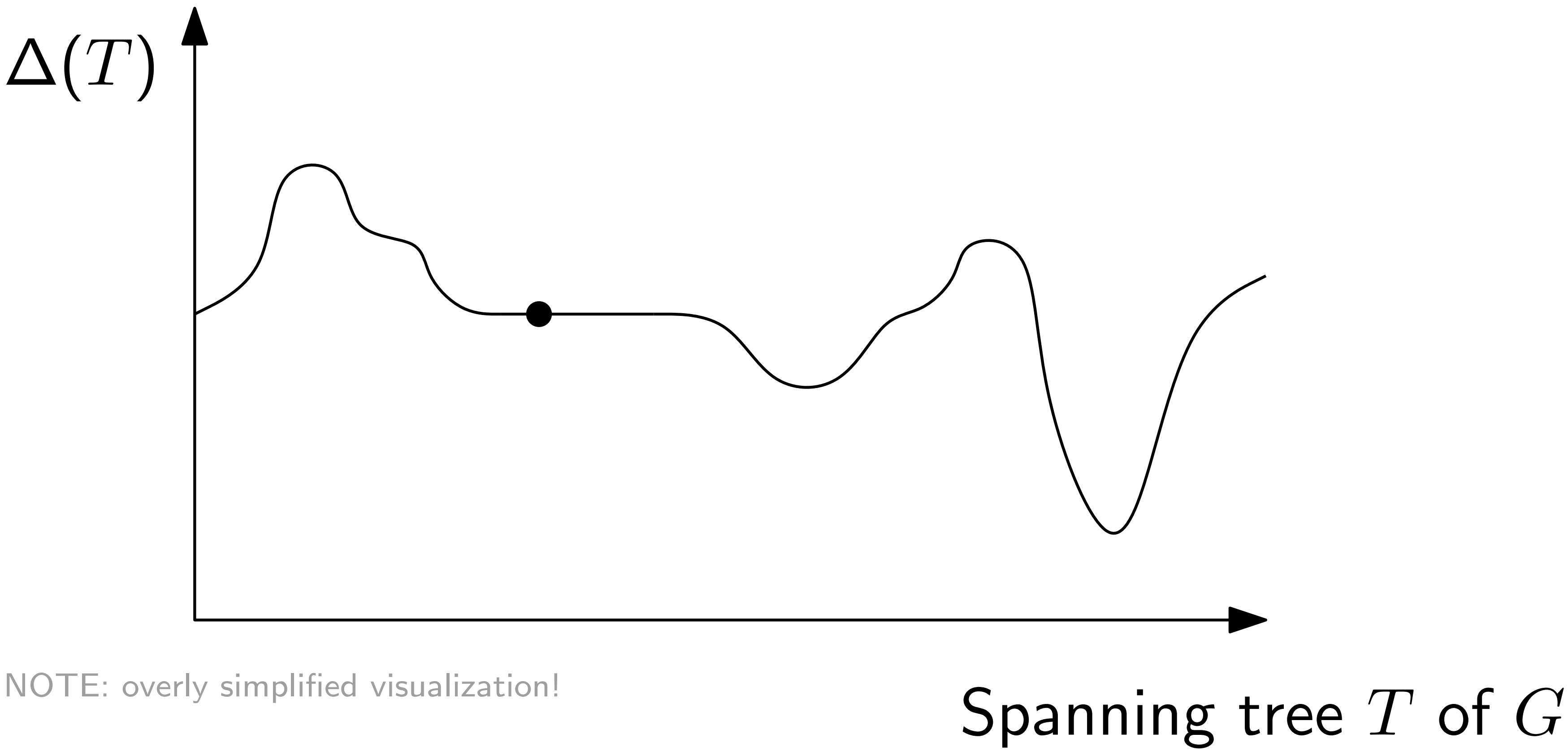
- Start from any spanning tree $T$ of $G$
- Perform edge flips until no flip improves the solution.




NOTE: overly simplified visualization!

# Local Search

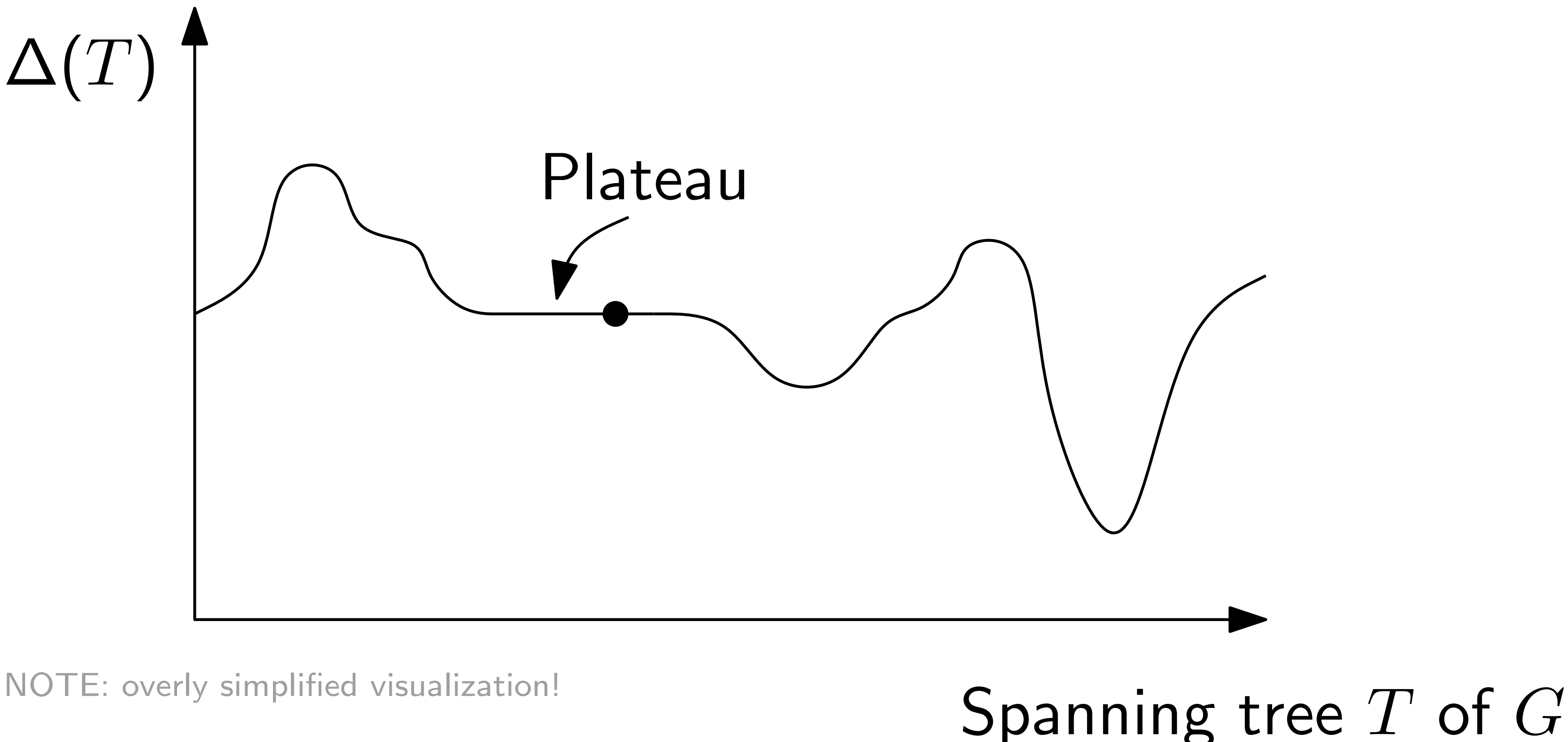
- Start from any spanning tree $T$ of $G$
- Perform edge flips until no flip improves the solution.




NOTE: overly simplified visualization!

# Local Search

- Start from any spanning tree $T$ of $G$
- Perform edge flips until no flip improves the solution.

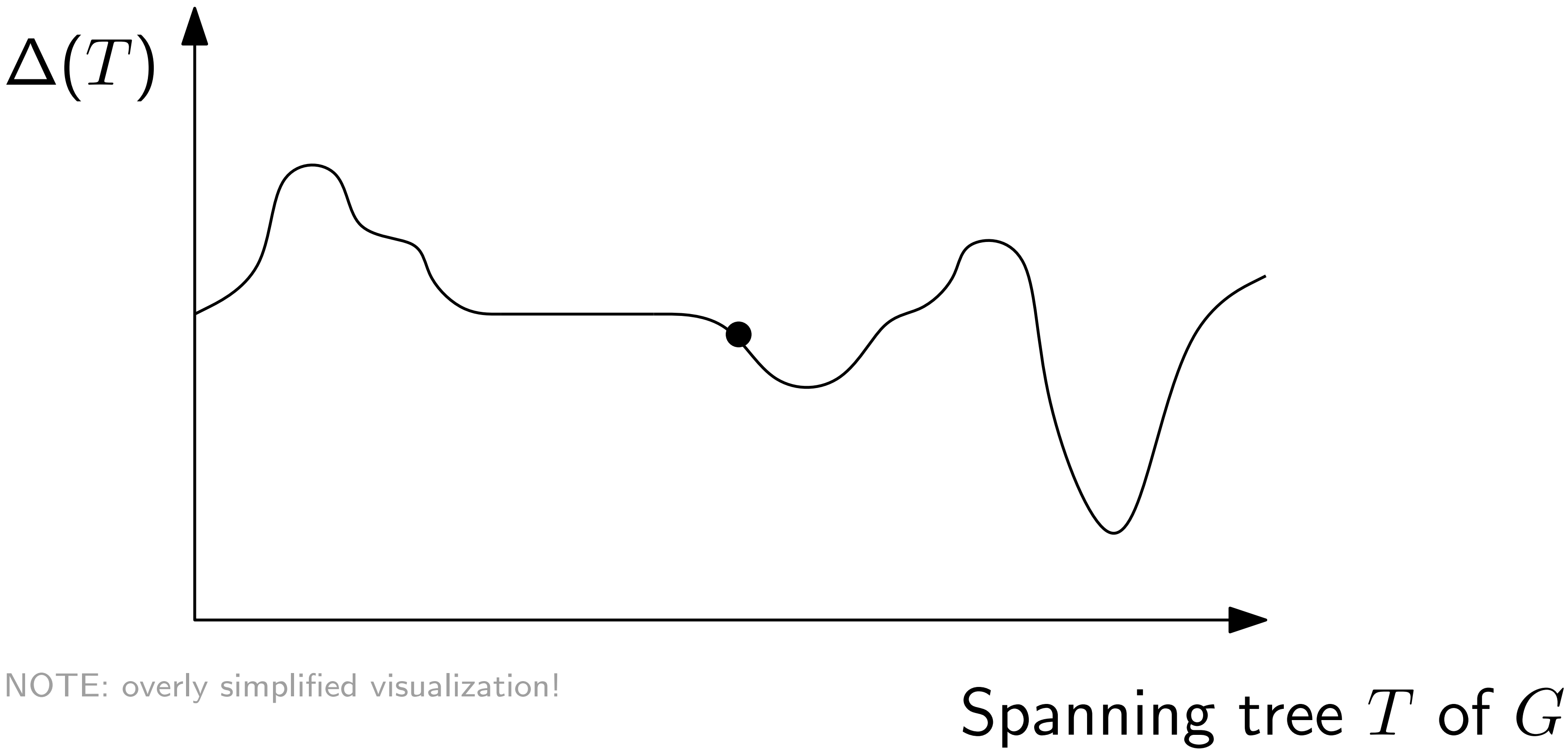



NOTE: overly simplified visualization!

Flips don't always improve $\Delta(T)$ !!

# Local Search

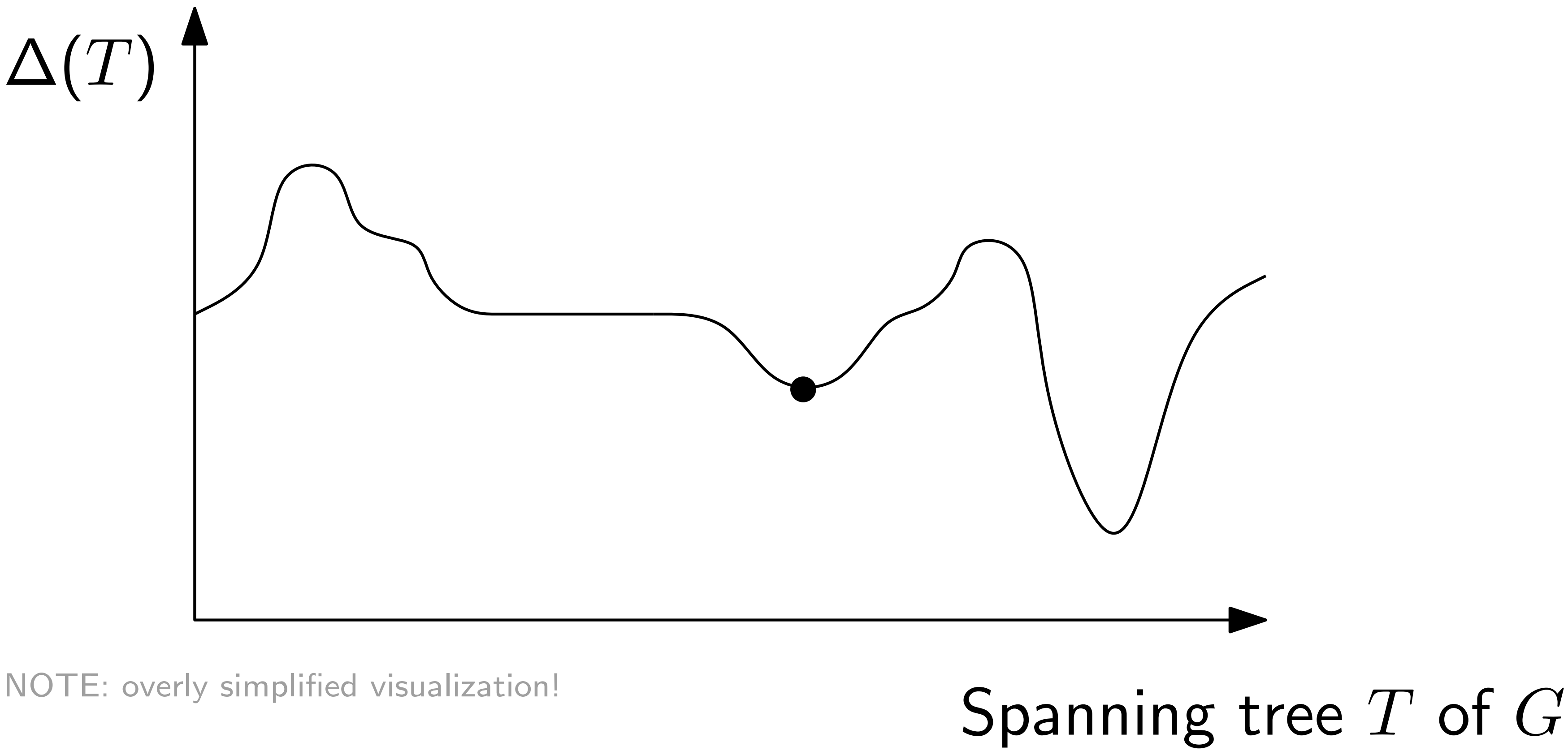
- Start from any spanning tree $T$ of $G$
- Perform edge flips until no flip improves the solution.




NOTE: overly simplified visualization!

# Local Search

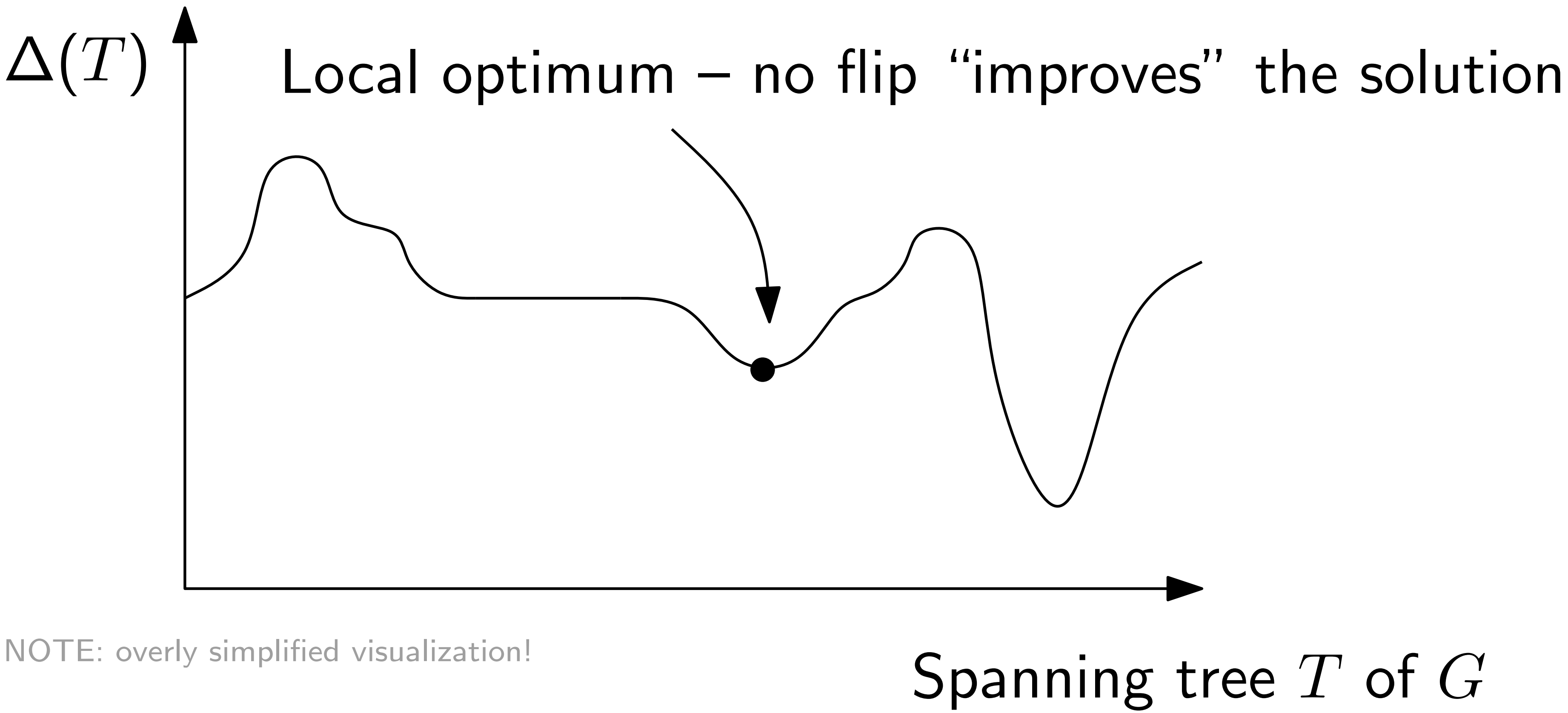
- Start from any spanning tree $T$ of $G$
- Perform edge flips until no flip improves the solution.




NOTE: overly simplified visualization!

# Local Search

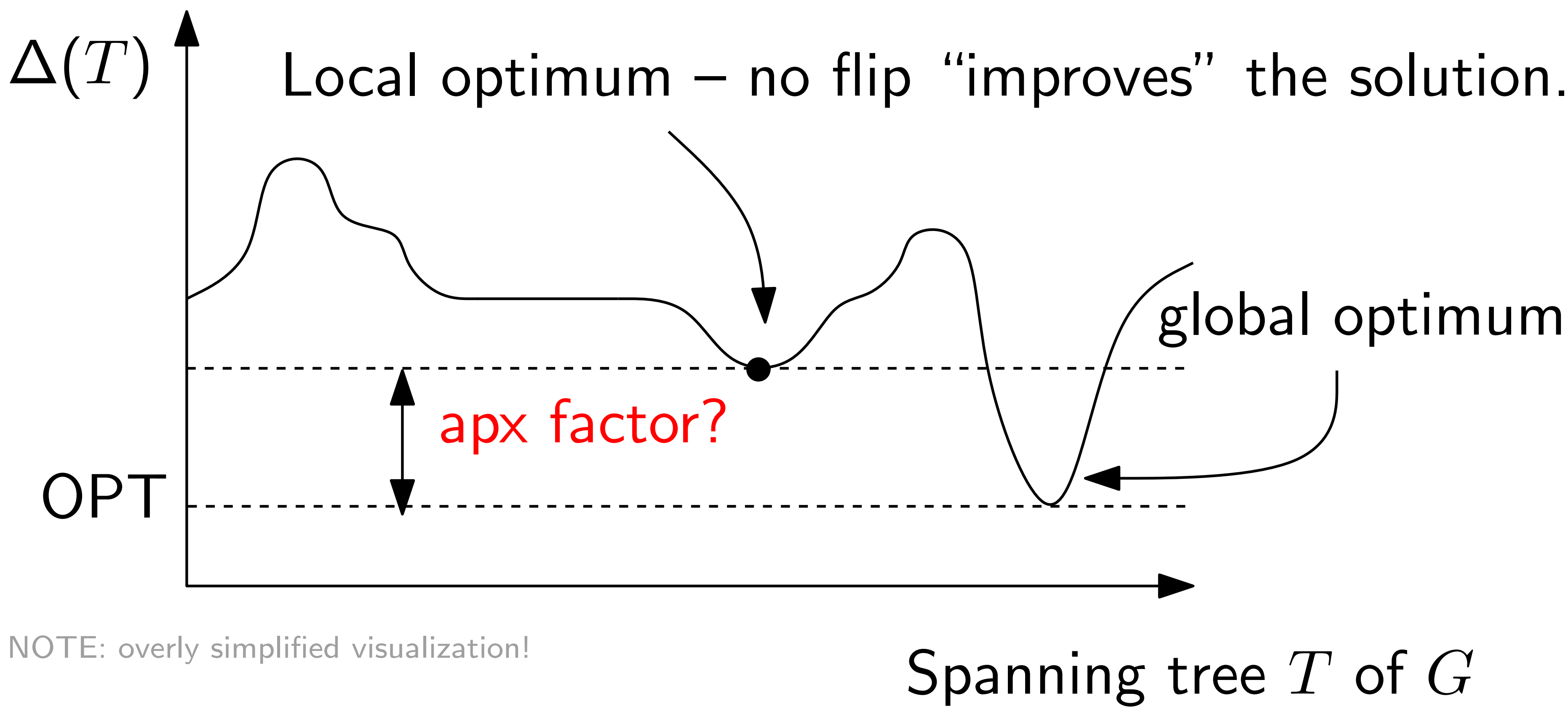
- Start from any spanning tree $T$ of $G$
- Perform edge flips until no flip improves the solution.




NOTE: overly simplified visualization!

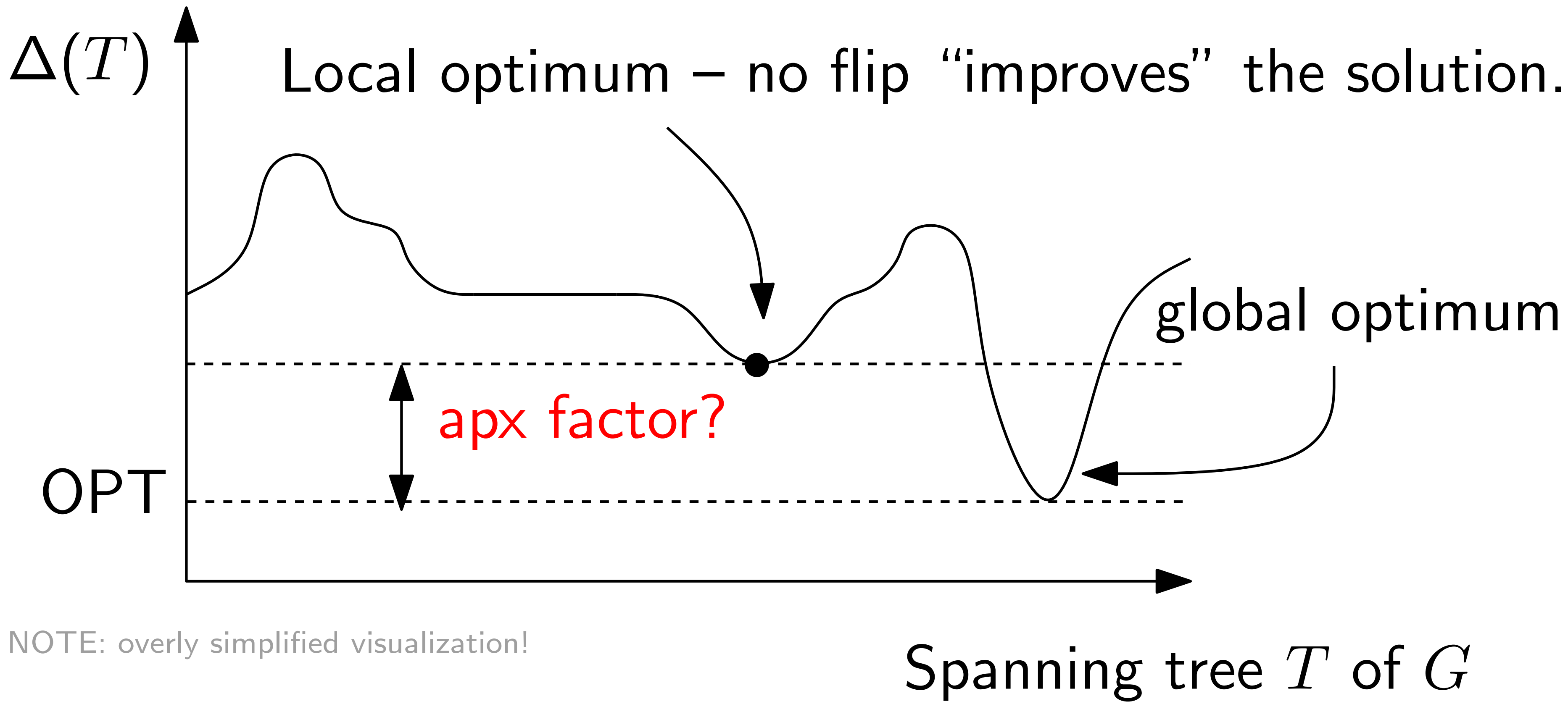
# Local Search

- Start from any spanning tree $T$ of $G$
- Perform edge flips until no flip improves the solution.



# Local Search

- Start from any spanning tree $T$ of $G$
- Perform edge flips until no flip improves the solution.




How to handle plateaus? What is the runtime?

# Local Search

Algorithm MinDegSTLocalSearch($T$)

**while** there is an "improving flip"(*) in $T$ for a vertex $v$ with $d_T(v) \geq \Delta(T) - \ell$ **do**

perform the flip.

(*) $uw \in E(G) \setminus E(T)$ with $d_T(v) - 1 > \max\{d_T(u), d_T(w)\}$ such that $T \cup \{uw\}$ forms a cycle containing $v$.

# Local Search

Algorithm MinDegSTLocalSearch($T$)
**while** there is an "improving flip" (*) in $T$ for a vertex $v$ with $d_T(v) \geq \Delta(T) - \ell$ **do**
perform the flip.

(*) $uw \in E(G) \setminus E(T)$ with $d_T(v) - 1 > \max\{d_T(u), d_T(w)\}$ such that $T \cup \{uw\}$ forms a cycle containing $v$.

- unclear whether it completes in polynomial time ...

# Local Search

Algorithm MinDegSTLocalSearch($T$)

**while** there is an "improving flip" (*) in $T$ for a vertex $v$ with $d_T(v) \geq \Delta(T) - \ell$ **do**

perform the flip.

(*) $uw \in E(G) \setminus E(T)$ with $d_T(v) - 1 > \max\{d_T(u), d_T(w)\}$ such that $T \cup \{uw\}$ forms a cycle containing $v$.

- unclear whether it completes in polynomial time ...
- idea: flip only when the degree of $v$ with $\deg(v) \geq \Delta(T) - \ell$ is reduced where $\ell := \lceil \log_2 n \rceil$

# Local Search

Algorithm MinDegSTLocalSearch($T$)
**while** there is an "improving flip" (*) in $T$ for a vertex $v$ with $d_T(v) \geq \Delta(T) - \ell$ **do**
  perform the flip.

(*) $uw \in E(G) \setminus E(T)$ with $d_T(v) - 1 > \max\{d_T(u), d_T(w)\}$ such that $T \cup \{uw\}$ forms a cycle containing $v$.

- unclear whether it completes in polynomial time ...
- idea: flip only when the degree of $v$ with $\deg(v) \geq \Delta(T) - \ell$ is reduced where $\ell := \lceil \log_2 n \rceil$
- first the approximation factor, then the runtime

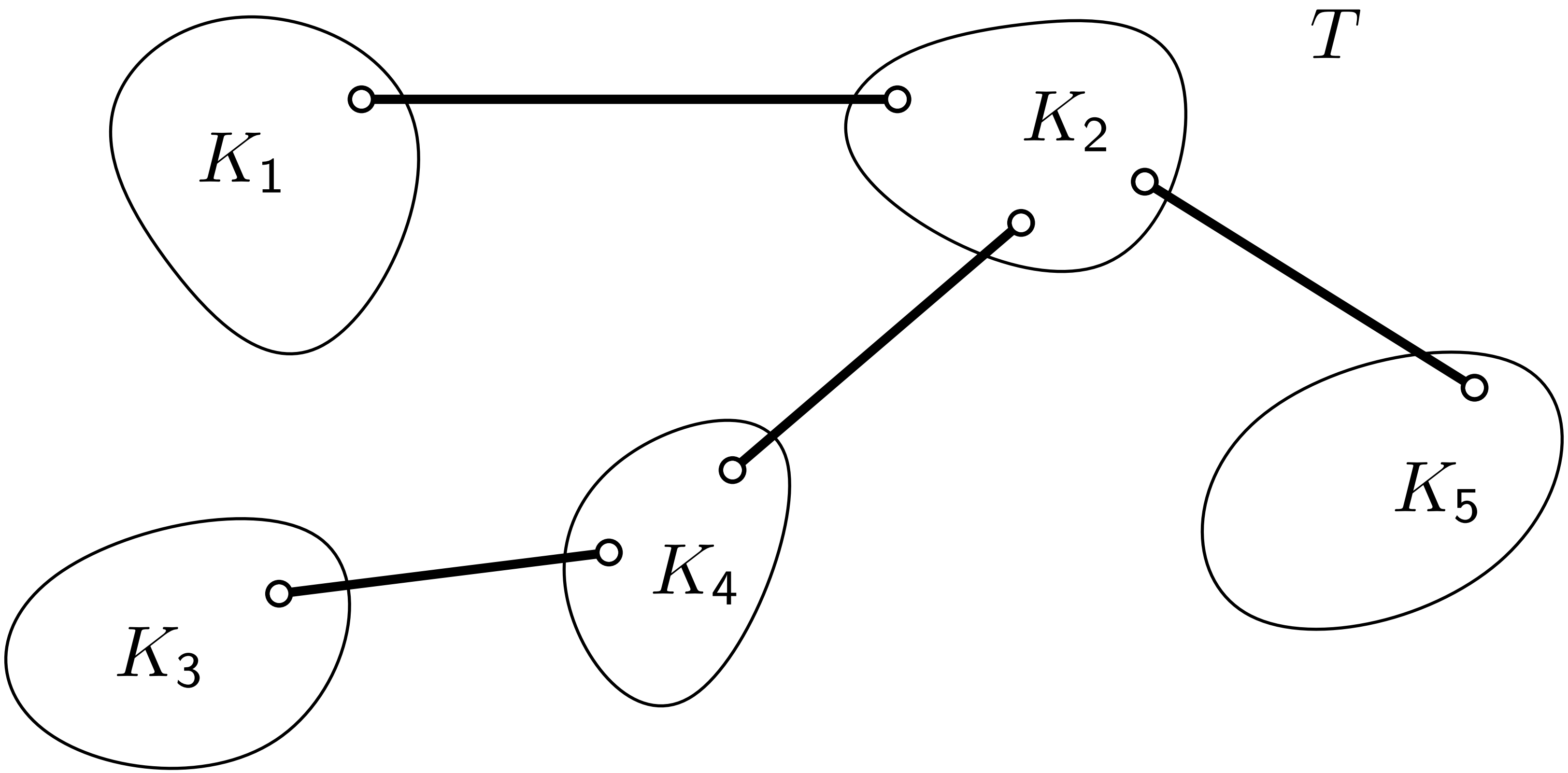
# Approximation Factor

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

# 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:** Lower bound on OPT

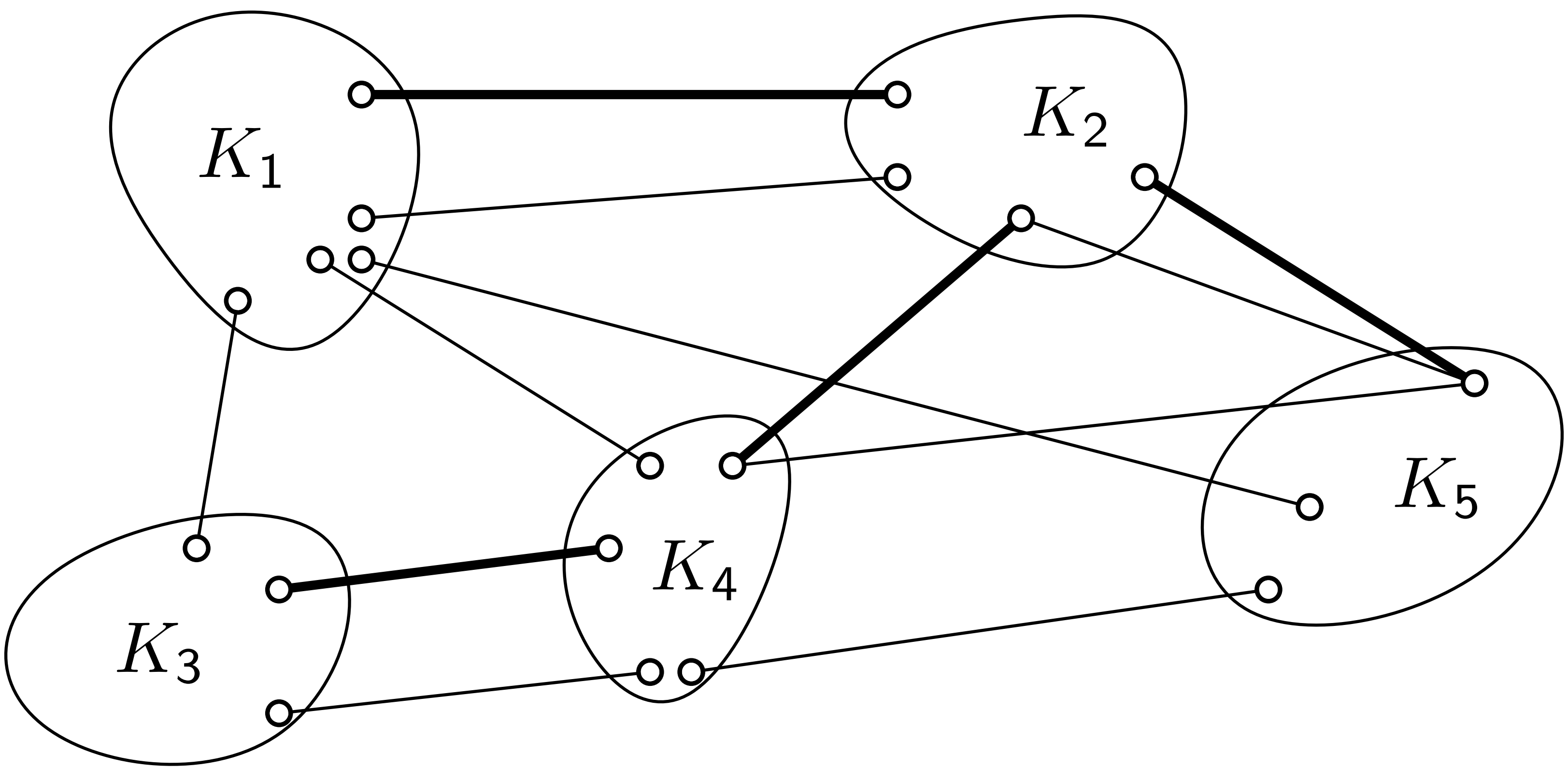



Removing $k$ edges partitions $T$ in $k+1$ components

# Approximation Factor

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

Let $E'$ be the edges of $G$ between distinct components ($K_i \neq K_j$).

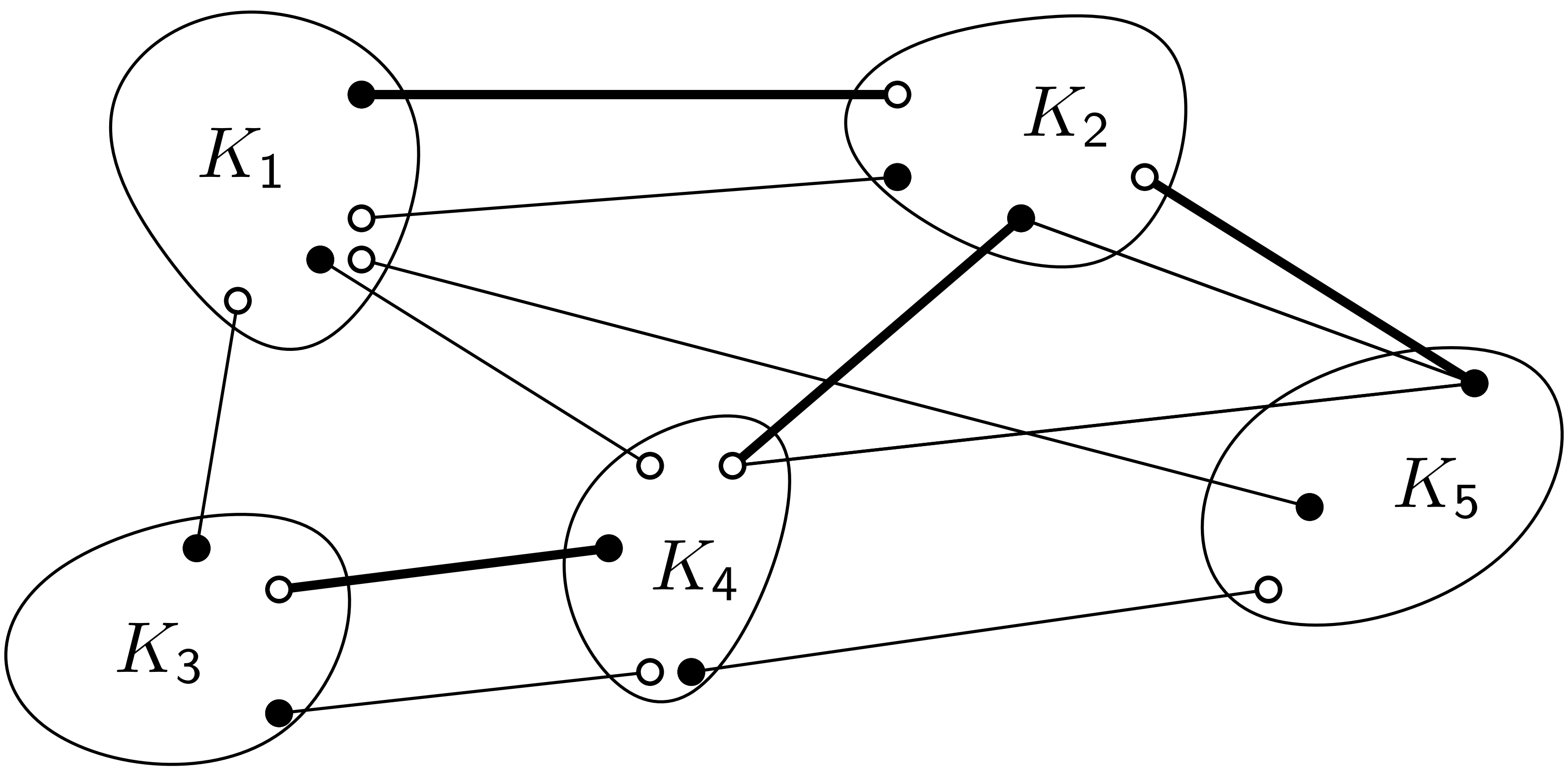


# Approximation Factor

**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:** Lower bound on OPT

Let $E'$ be the edges of $G$ between distinct components ($K_i \neq K_j$).



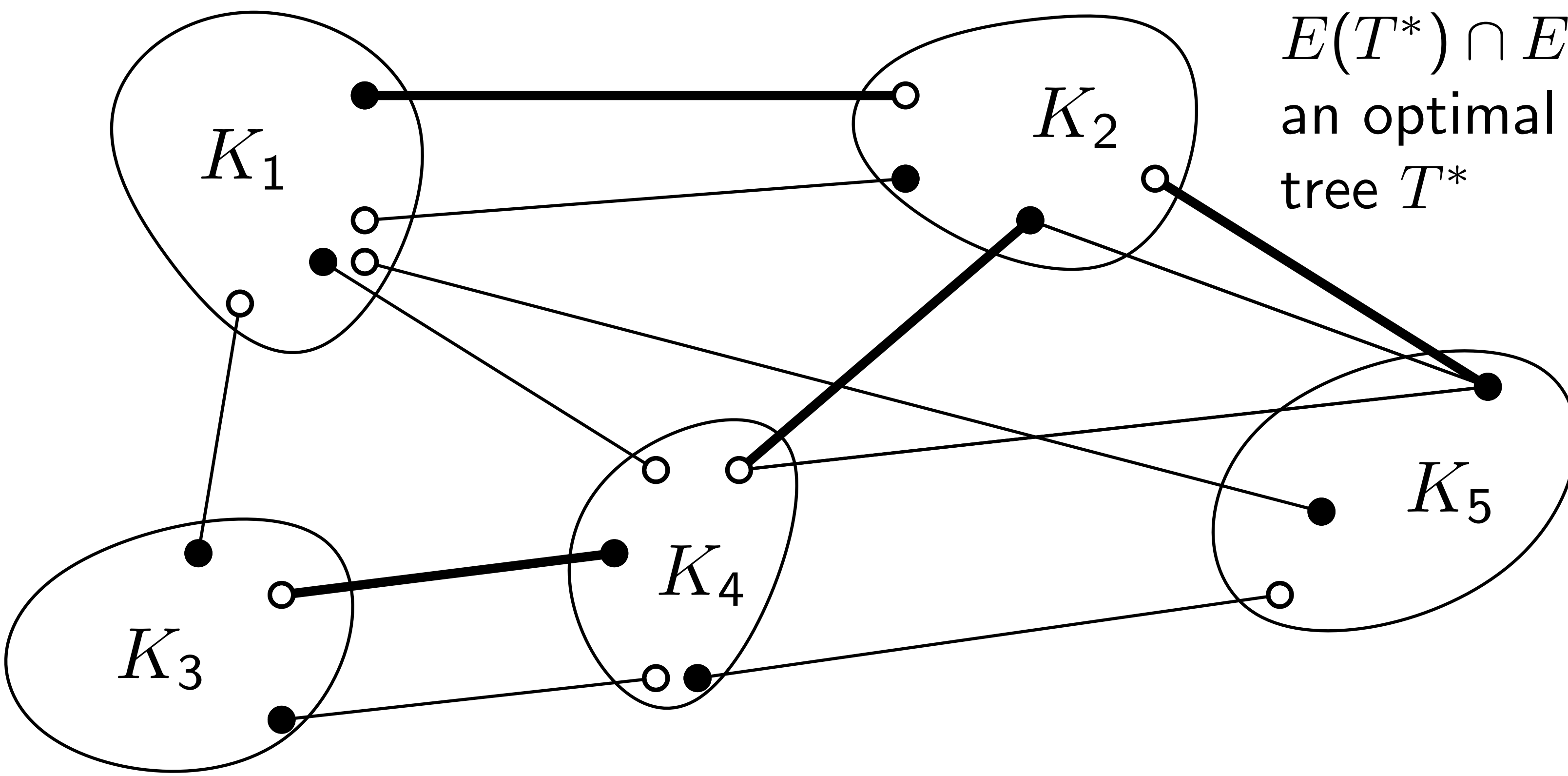

Vertex Cover $S$ of $E'$

# 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:** Lower bound on OPT

Let $E'$ be the edges of $G$ between distinct components ($K_i \neq K_j$).




$E(T^*) \cap E' \geq k$ for an optimal spanning tree $T^*$

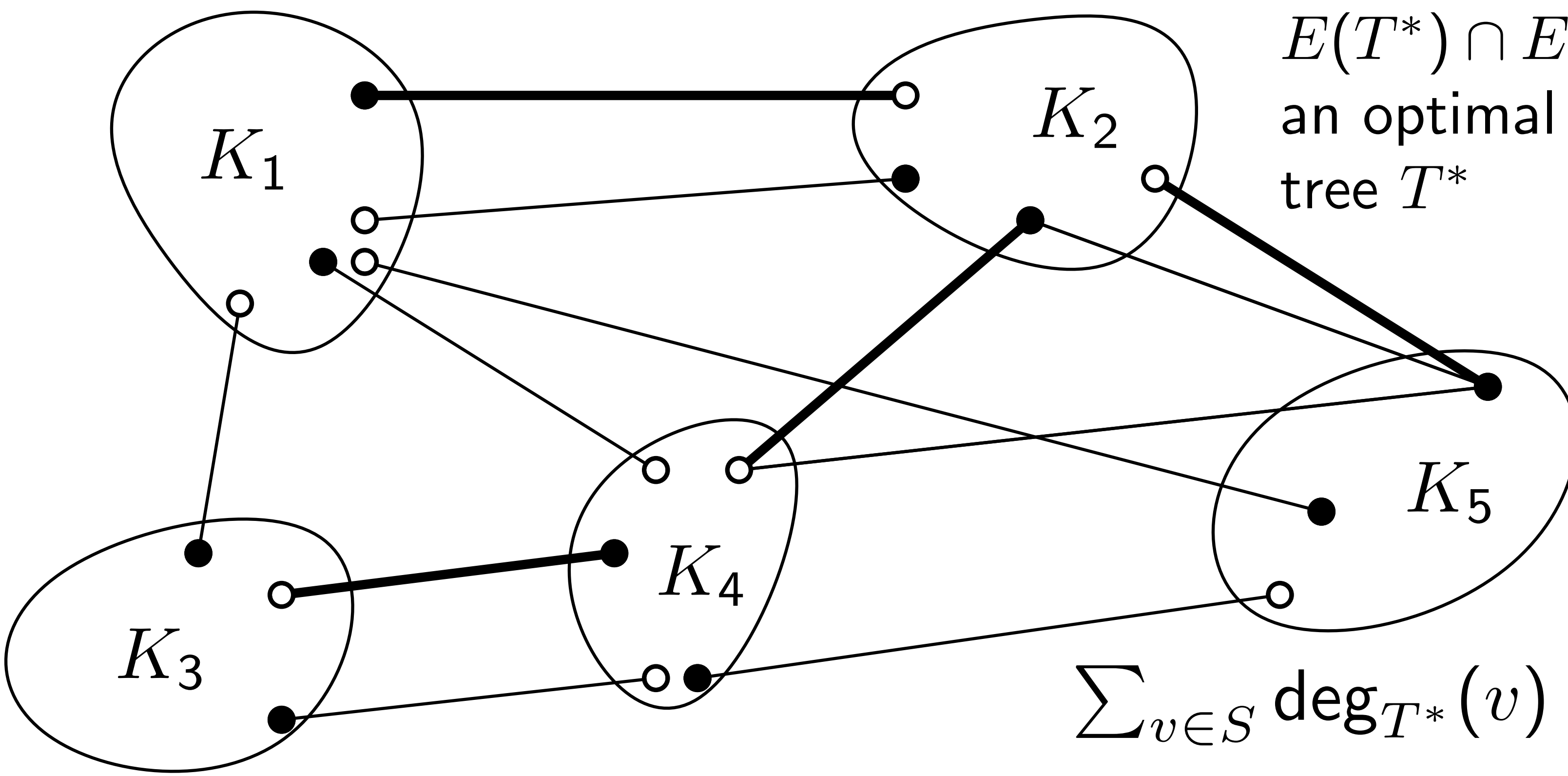
Vertex Cover $S$ of $E'$

# 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:** Lower bound on OPT

Let $E'$ be the edges of $G$ between distinct components ($K_i \neq K_j$).




$E(T^*) \cap E' \geq k$ for an optimal spanning tree $T^*$

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

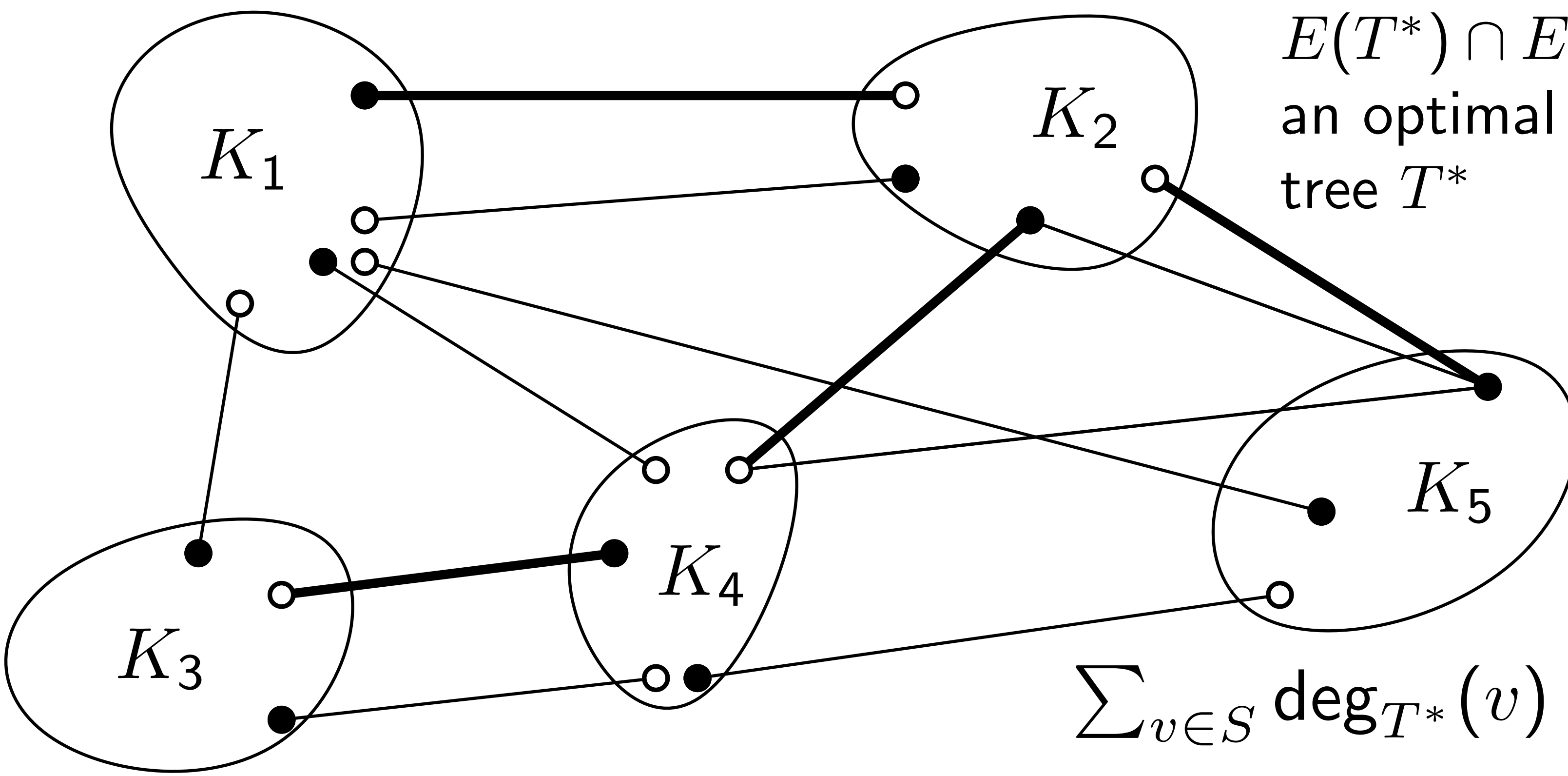
Vertex Cover $S$ of $E'$

# 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:** Lower bound on OPT $\boxed{\mathrm{OPT} \geq k/|S|}$

Let $E'$ be the edges of $G$ between distinct components ($K_i \neq K_j$).




$E(T^*) \cap E' \geq k$ for an optimal spanning tree $T^*$

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

Vertex Cover $S$ of $E'$

# 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:** $\mathrm{OPT} \geq k/|S|$
**Part 2:** Applying the bound.

# Approximation Factor

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

**Proof.** **Part 1:** $\mathsf{OPT} \geq k/|S|$
**Part 2:** Applying the bound.

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

# Approximation Factor

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

**Proof.** **Part 1:** $\text{OPT} \geq k/|S|$
**Part 2:** Applying the bound.

Let $S_i$ be the nodes in $T$ with $d_T(v) \geq i$.
Let $E_i$ be the edges of $T$ incident to $S_i$.
**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:** $\mathrm{OPT} \geq k/|S|$
**Part 2:** Applying the bound.

Let $S_i$ be the nodes in $T$ with $d_T(v) \geq i$.
Let $E_i$ be the edges of $T$ incident to $S_i$.
**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 $i$ 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:** $\mathrm{OPT} \geq k/|S|$
**Part 2:** Applying the bound.

Let $S_i$ be the nodes in $T$ with $d_T(v) \geq i$.
Let $E_i$ be the edges of $T$ incident to $S_i$.
**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 $i$ 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:** $\mathrm{OPT} \geq k/|S|$
**Part 2:** Applying the bound.

Let $S_i$ be the nodes in $T$ with $d_T(v) \geq i$.
Let $E_i$ be the edges of $T$ incident to $S_i$.
**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 $i$ such that $|S_{i-1}| \leq 2|S_i|$.

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

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

# Approximation Factor

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

**Proof.** **Part 1:** $\text{OPT} \geq k/|S|$
**Part 2:** Applying the bound.

Let $S_i$ be the nodes in $T$ with $d_T(v) \geq i$.
Let $E_i$ be the edges of $T$ incident to $S_i$.
**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 $i$ such that $|S_{i-1}| \leq 2|S_i|$.

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

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

$\square$

# Runtime

**Thm.** The algorithm finds a local optimal in polynomial time.

**Proof.**

# Runtime

**Thm.** The algorithm finds a local optimal in polynomial time.

**Proof.**

Via potenial function $\Phi(G, T)$. $\rightsquigarrow$ a function measuring the value of a solution where, e.g., :

- each iteration decreases the potential of a solution.

# Runtime

**Thm.** The algorithm finds a local optimal in polynomial time.

**Proof.**

Via potenial function $\Phi(G, T)$. $\rightsquigarrow$ a function measuring the value of a solution where, e.g., :

- each iteration decreases the potential of a solution.

- the function is bounded both from above and below.

# Runtime

**Thm.** The algorithm finds a local optimal in polynomial time.

**Proof.**

Via potenial function $\Phi(G, T)$. $\rightsquigarrow$ a function measuring the value of a solution where, e.g., :

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

# Runtime

**Thm.** The algorithm finds a local optimal in polynomial time.

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

Via potenial function $\Phi(G, T)$. $\rightsquigarrow$ a function measuring the value of a solution where, e.g., :

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

# Runtime

**Thm.** The algorithm finds a local optimal in polynomial time.

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

Via potenial function $\Phi(G, T)$. $\rightsquigarrow$ a function measuring the value of a solution where, e.g., :

- each iteration decreases the potential of a solution.

**Lemma:** each iteration $\Phi(T') \leq (1 - \frac{2}{27n^3})\Phi(T)$.

- the function is bounded both from above and below.

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

# Runtime

**Thm.** The algorithm finds a local optimal in polynomial time.

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

Via potenial function $\Phi(G, T)$. $\rightsquigarrow$ a function measuring the value of a solution where, e.g., :

- each iteration decreases the potential of a solution.

**Lemma:** each iteration $\Phi(T') \leq (1 - \frac{2}{27n^3})\Phi(T)$.

- 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 execeed this lower bound.

# Runtime

**Thm.** The algorithm finds a local optimal in polynomial time.

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

Via potenial function $\Phi(G, T)$. $\rightsquigarrow$ a function measuring the value of a solution where, e.g., :

- each iteration decreases the potential of a solution.

**Lemma:** each iteration $\Phi(T') \leq (1 - \frac{2}{27n^3})\Phi(T)$.

- 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 execeed this lower bound.

How does $\Phi(T)$ change?

# Runtime

**Thm.** The algorithm finds a local optimal in polynomial time.

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

Via potenial function $\Phi(G,T)$. $\rightsquigarrow$ a function measuring the value of a solution where, e.g., :

- each iteration decreases the potential of a solution.

**Lemma:** each iteration $\Phi(T') \leq (1 - \frac{2}{27n^3})\Phi(T)$.

- 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 execeed 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 local optimal in polynomial time.

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

Via potenial function $\Phi(G, T)$. $\rightsquigarrow$ a function measuring the value of a solution where, e.g., :

- each iteration decreases the potential of a solution.

**Lemma:** each iteration $\Phi(T') \leq (1 - \frac{2}{27n^3})\Phi(T)$.

- 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 execeed 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 $\rightsquigarrow$ after $f(n)$ iterations $\Phi(T) = n < 3n$

# Runtime

**Thm.** The algorithm finds a local optimal in polynomial time.

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

Via potenial function $\Phi(G, T)$. $\rightsquigarrow$ a function measuring the value of a solution where, e.g., :

- each iteration decreases the potential of a solution.

**Lemma:** each iteration $\Phi(T') \leq (1 - \frac{2}{27n^3})\Phi(T)$.

- 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 execeed this lower bound.

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

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

Goal $\rightsquigarrow$ after $f(n)$ iterations $\Phi(T) = n < 3n$

# Runtime

**Thm.** The algorithm finds a local optimal in polynomial time.

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

Via potenial function $\Phi(G, T)$. $\rightsquigarrow$ a function measuring the value of a solution where, e.g., :

- each iteration decreases the potential of a solution.

**Lemma:** each iteration $\Phi(T') \leq (1 - \frac{2}{27n^3})\Phi(T)$.

- 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 execeed this lower bound.

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

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

Goal $\rightsquigarrow$ after $f(n)$ iterations $\Phi(T) = n < 3n$ $\square$

# Extensions

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

**Proof.** Similar to before. $\square$

# Extensions

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

**Proof.** Similar to before. $\square$

**Next Class:**
**Approximation Schemes:**
**$(1 + \epsilon)$-approximation**