FPT-Approximation: Upper Bounds

Recent Advances in Parameterized Complexity, 2017

Meirav Zehavi

Outline

1. Introduction
2. Partial Vertex Cover: $(1+\varepsilon)$-approx.
3. k-Path Transversal: $O(\log k)$-approx.

Introduction

Parameterized approximation algorithm.

Parameter. k.

Approximation factor. c.

Introduction

Parameterized approximation algorithm.

Parameter. k.

Approximation factor. c.

[Try 1.] Given an instance (I,k), return in time $f(k)n^{O(1)}$ a solution S such that

- $\mathrm{val}(S) \leq c \cdot opt$ (for minimization).
- $\mathrm{val}(S) \geq opt/c$ (for maximization).

Introduction

Parameterized approximation algorithm.

Parameter. k.

Approximation factor. c.

[Try 1.] Given an instance (I,k), return in time $f(k)n^{O(1)}$ a solution S such that

- $\mathrm{val}(S) \leq c \cdot opt$ (for minimization).
- $\mathrm{val}(S) \geq opt/c$ (for maximization).

Say k=1 …

If k is the value of the solution, does it play a role?

In case opt exceeds k, the task should become easier.

Introduction

Parameterized approximation algorithm.

Parameter. k.

Approximation factor. c.

[Try 1.] Given an instance (I,k), return in time $f(k)n^{O(1)}$ a solution S such that

- $\mathrm{val}(S) \leq c \cdot opt$ (for minimization).
- $\mathrm{val}(S) \geq opt/c$ (for maximization).

If k is the value of the solution, does it play a role?

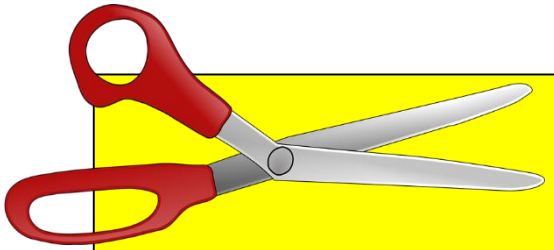
In case opt exceeds k, the task should become easier.

Parameterized approximation algorithm.
Parameter. k.
Approximation factor. c.

~~[Try 1.]~~ Given an instance (I,k), return in time $f(k)n^{O(1)}$ a solution S such that
- $\prod(S) \leq c \cdot opt$ (for minimization).
- $\prod(S) \geq opt/c$ (for maximization).

If k is the value of the solution, does it play a role?
In case opt exceeds k, the task should become trivial.

Introduction



$$\prod(S) = \min\{val(S), k+1\}$$

~~[Try 1.]~~ Given an instance (I,k), return in time $f(k)n^{O(1)}$ a solution S such that
- $\prod(S) \leq c \cdot opt$ (for minimization).
- $\prod(S) \geq opt/c$ (for maximization).

If k is the value of the solution, does it play a role?
In case opt exceeds k, the task should become easier.

Parameterized approximation algorithm.
Parameter. k.
Approximation factor. c.

~~[Try 1.]~~ Given an instance (I,k), return in time $f(k)n^{O(1)}$ a solution S such that
- $\prod(S) \le c \cdot opt$ (for minimization).
- $\prod(S) \ge opt/c$ (for maximization).

If k is the value of the solution, does it play a role?
In case opt exceeds k, the task should become trivial.

Introduction

$$\prod(S) = \min\{\mathrm{val}(S), k+1\}$$

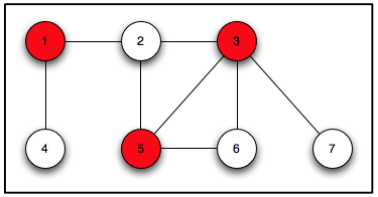
Our task is trivial (min) when $opt \ge (k+1)/c$.

~~[Try 1.]~~ Given an instance (I,k), return in time $f(k)n^{O(1)}$ a solution S such that
- $\prod(S) \le c \cdot opt$ (for minimization).
- $\prod(S) \ge opt/c$ (for maximization).

If k is the value of the solution, does it play a role?
In case opt exceeds k, the task should become easier.

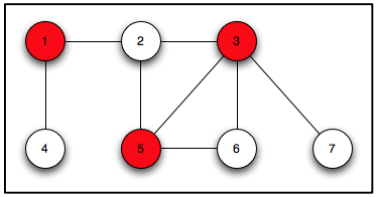
Outline

1. **Introduction**
2. **Partial Vertex Cover: (1+ε)-approx.**
3. ***k*-Path Transversal: *O*(log *k*)-approx.**



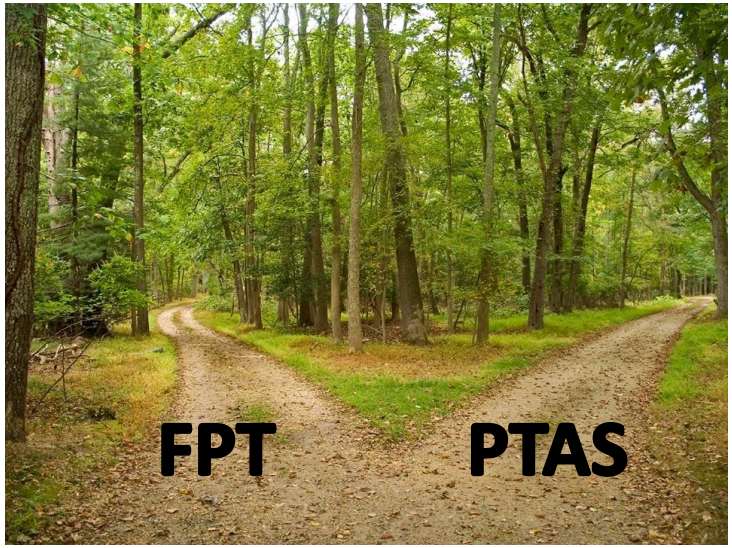
Partial Vertex Cover

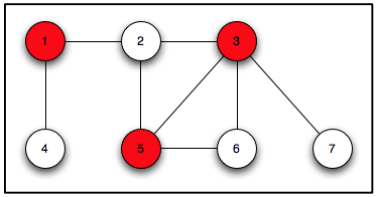
Partial Vertex Cover. Given a graph *G* and a parameter *k*, what is the maximum number of edges in *G* that can be covered by *k* vertices?



Partial Vertex Cover

Partial Vertex Cover. Given a graph *G* and a parameter *k*, what is the maximum number of edges in *G* that can be covered by *k* vertices?

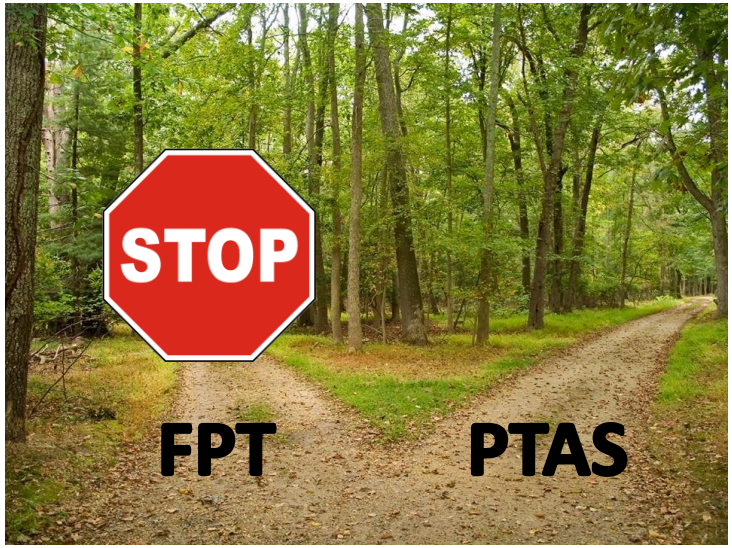


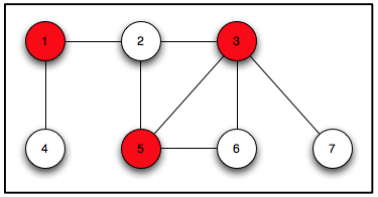


Partial Vertex Cover

Partial Vertex Cover. Given a graph *G* and a parameter *k*, what is the maximum number of edges in *G* that can be covered by *k* vertices?

W[1]-hard w.r.t. *k* !



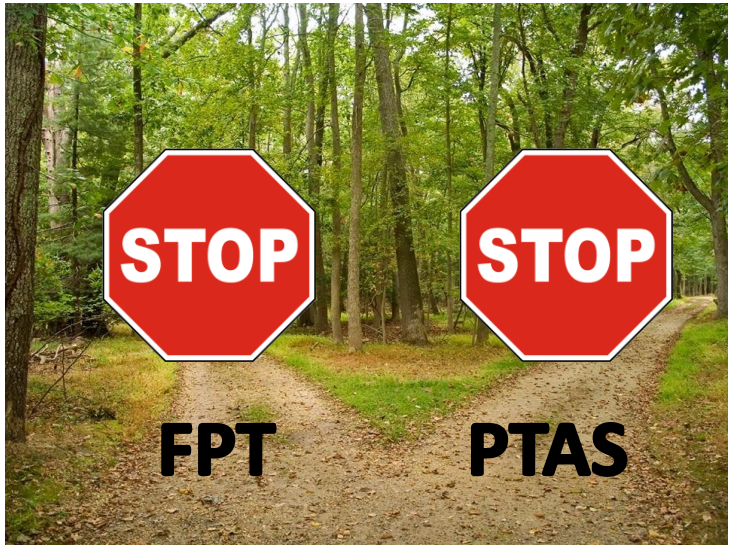


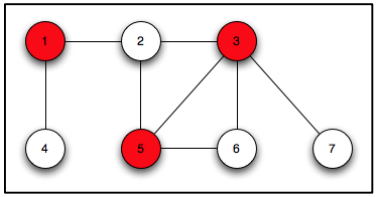
Partial Vertex Cover

Partial Vertex Cover. Given a graph *G* and a parameter *k*, what is the maximum number of edges in *G* that can be covered by *k* vertices?

W[1]-hard w.r.t. *k* !

No PTAS !





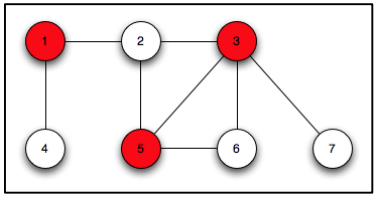
Partial Vertex Cover

Partial Vertex Cover. Given a graph *G* and a parameter *k*, what is the maximum number of edges in *G* that can be covered by *k* vertices?

W[1]-hard w.r.t. *k* !

No PTAS !

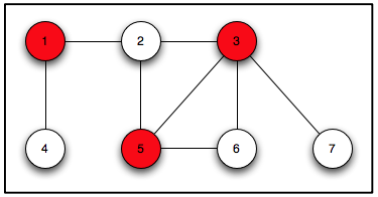
FPT-Approximation Scheme !



Partial Vertex Cover

Partial Vertex Cover. Given a graph G and a parameter k, what is the maximum number of edges in G that can be covered by k vertices?

$(1+\varepsilon)$-Approximation parameterized algorithm. Given (G,k), return $U \subseteq V(G)$ of size k that covers at least $opt/(1+\varepsilon)$ edges in G.



Partial Vertex Cover

Denote $D = 2\binom{k}{2}/\varepsilon$. Order $V(G) = \{v_1,\ldots,v_n\}$ s.t. $d(v_i) \geq d(v_{i+1})$ for all $i \in [n\text{-}1]$.

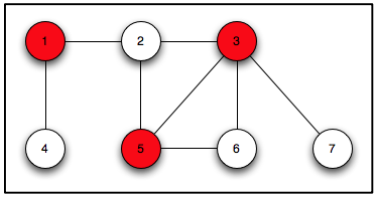
Denote $D = 2\binom{k}{2}/\varepsilon$. Order $V(G) = \{v_1,\ldots,v_n\}$ s.t. $d(v_i) \geq d(v_{i+1})$ for all $i \in [n\text{-}1]$.

Case 1. $d(v_1) \geq D$.

Partial Vertex Cover

Denote $D = 2\binom{k}{2}/\varepsilon$. Order $V(G) = \{v_1,\ldots,v_n\}$ s.t. $d(v_i) \geq d(v_{i+1})$ for all $i \in [n\text{-}1]$.

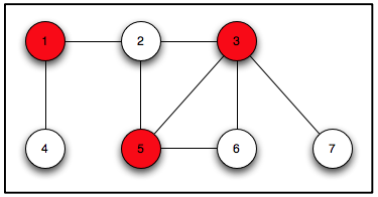
Case 1. $d(v_1) \geq D$.



Partial Vertex Cover

Denote $D = 2\binom{k}{2}/\varepsilon$. Order $V(G) = \{v_1,\ldots,v_n\}$ s.t. $d(v_i) \geq d(v_{i+1})$ for all $i \in [n\text{-}1]$.

Case 1. $d(v_1) \geq D$.
Output. $S = \{v_1,\ldots,v_k\}$.



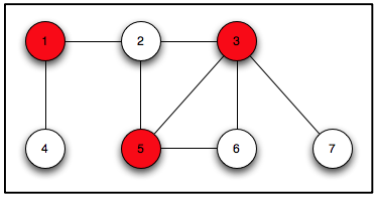
Partial Vertex Cover

Denote $D = 2\binom{k}{2}/\varepsilon$. Order $V(G) = \{v_1,\ldots,v_n\}$ s.t. $d(v_i) \geq d(v_{i+1})$ for all $i \in [n-1]$.

Case 1. $d(v_1) \geq D$.

Output. $S = \{v_1,\ldots,v_k\}$.

- $\mathrm{val}(S) \geq \sum_{i=1}^{k} d(v_i) - \binom{k}{2}$.
- $\mathrm{opt} \leq \sum_{i=1}^{k} d(v_i)$.



Partial Vertex Cover

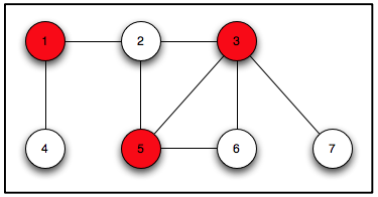
Denote $D = 2\binom{k}{2}/\varepsilon$. Order $V(G) = \{v_1,\ldots,v_n\}$ s.t. $d(v_i) \geq d(v_{i+1})$ for all $i \in [n-1]$.

Case 1. $d(v_1) \geq D$.

Output. $S = \{v_1,\ldots,v_k\}$.

- $\mathrm{val}(S) \geq \sum_{i=1}^{k} d(v_i) - \binom{k}{2}$.
- $\mathrm{opt} \leq \sum_{i=1}^{k} d(v_i)$.

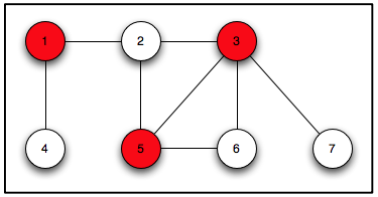
$$\frac{val(S)}{opt} \geq \frac{\sum_{i=1}^{k} d(v_i) - \binom{k}{2}}{\sum_{i=1}^{k} d(v_i)} \geq 1 - \frac{\binom{k}{2}}{D} = 1 - \varepsilon/2 \geq 1/(1+\varepsilon).$$



Partial Vertex Cover

Denote $D = 2\binom{k}{2}/\varepsilon$. Order $V(G) = \{v_1,\ldots,v_n\}$ s.t. $d(v_i) \geq d(v_{i+1})$ for all $i \in [n\text{-}1]$.

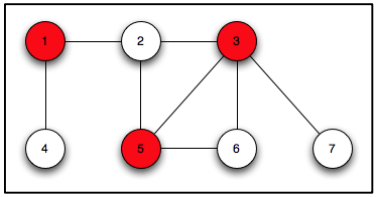
Case 2. $d(v_1) \leq D$



Partial Vertex Cover

Denote $D = 2\binom{k}{2}/\varepsilon$. Order $V(G) = \{v_1,\ldots,v_n\}$ s.t. $d(v_i) \geq d(v_{i+1})$ for all $i \in [n\text{-}1]$.

Case 2. $d(v_1) \leq D \rightarrow$ Max degree $\leq D$.

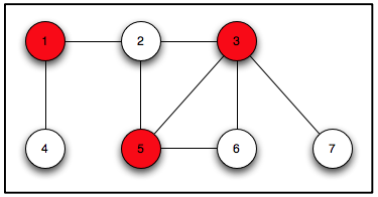


Partial Vertex Cover

Denote $D = 2\binom{k}{2}/\varepsilon$. Order $V(G) = \{v_1,\ldots,v_n\}$ s.t. $d(v_i) \geq d(v_{i+1})$ for all $i \in [n\text{-}1]$.

Case 2. $d(v_1) \leq D \rightarrow$ Max degree $\leq D$.

Can solve the problem optimally!

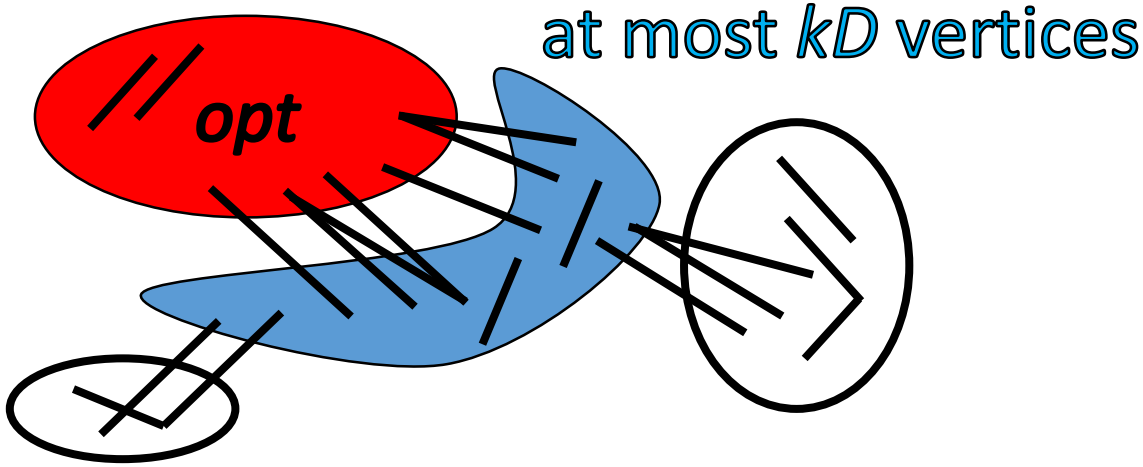


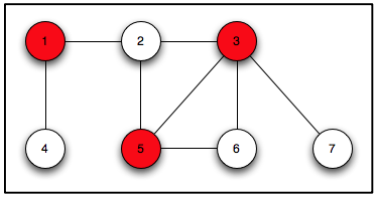
Partial Vertex Cover

Denote $D = 2\binom{k}{2}/\varepsilon$. Order $V(G) = \{v_1,\ldots,v_n\}$ s.t. $d(v_i) \geq d(v_{i+1})$ for all $i \in [n\text{-}1]$.

Case 2. $d(v_1) \leq D \rightarrow$ Max degree $\leq D$.

Can solve the problem optimally!



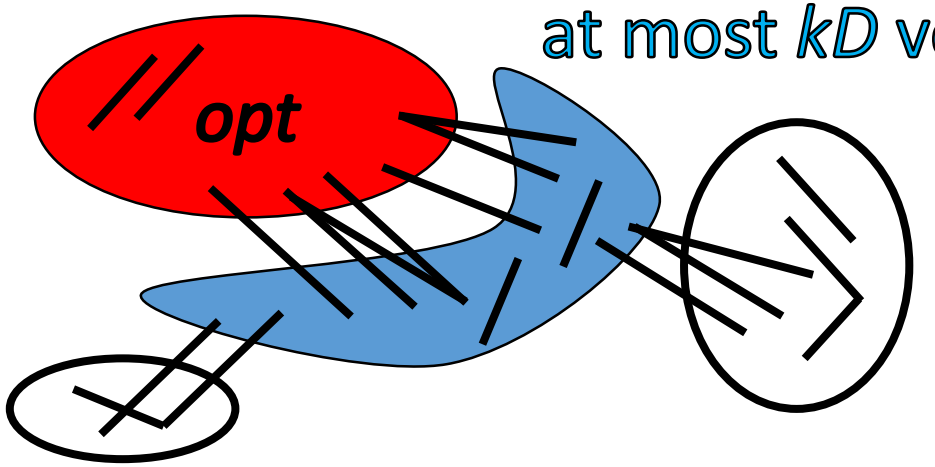


Partial Vertex Cover

Denote $D = 2\binom{k}{2}/\varepsilon$. Order $V(G) = \{v_1,\ldots,v_n\}$ s.t. $d(v_i) \geq d(v_{i+1})$ for all $i \in [n\text{-}1]$.

Case 2. $d(v_1) \leq D \rightarrow$ Max degree $\leq D$.

Can solve the problem optimally!



Solve using randomized separation in time $\binom{kD+k}{k} n^{O(1)}$.

Outline

1. **Introduction**
2. **Partial Vertex Cover: (1+ε)-approx.**
3. ***k*-Path Transversal: *O*(log *k*)-approx.**

k-Path Transversal

k-Path Transversal. Given a graph G and a parameter k, what is the minimum size of $S \subseteq V(G)$ such that $G \backslash S$ has no path on (at least) k vertices?

k-Path Transversal

k-Path Transversal. Given a graph G and a parameter k, what is the minimum size of $S \subseteq V(G)$ such that $G \backslash S$ has no path on (at least) k vertices?

Need FPT. Checking whether there is a k-Path is NP-hard.

k-Path Transversal

k-Path Transversal. Given a graph G and a parameter k, what is the minimum size of $S \subseteq V(G)$ such that $G \backslash S$ has no path on (at least) k vertices?

Need FPT. Checking whether there is a k-Path is NP-hard.

Need approximation. Hitting 2-paths is NP-hard.

k-Path Transversal

k-Path Transversal. Given a graph G and a parameter k, what is the minimum size of $S \subseteq V(G)$ such that $G \setminus S$ has no path on (at least) k vertices?

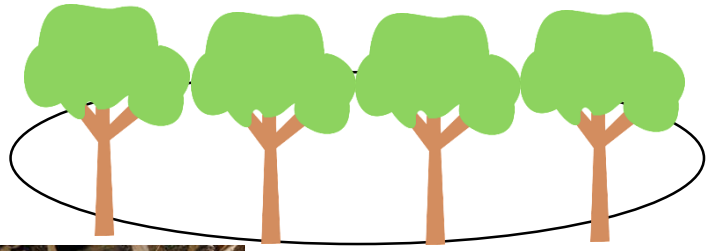
$O(\log k)$-Approximation parameterized algorithm. Given (G,k), return $S \subseteq V(G)$ of size $O(opt \cdot \log k)$ such that $G \setminus S$ has no path on (at least) k vertices.

k-Path Transversal

Theorem (Lee 2017). k-Path Transversal admits an $O(\log k)$-approximation algorithm that runs in time $2^{O(k^3 \log k)} n^{O(1)}$.

k-Path Transversal

$O(\log k \cdot opt)$

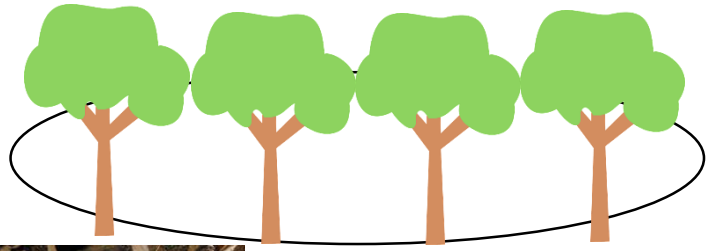


Our goal.

connected components

k-Path Transversal

$O(\log k \cdot opt)$

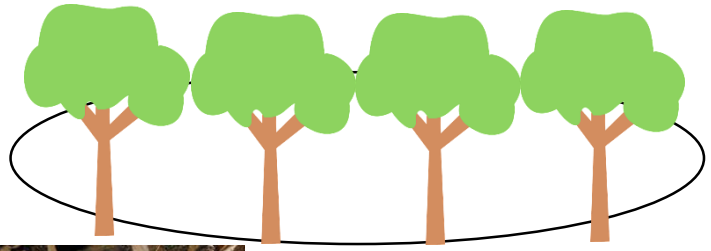


Different view.

connected components

k-Path Transversal

$O(\log k \cdot opt)$



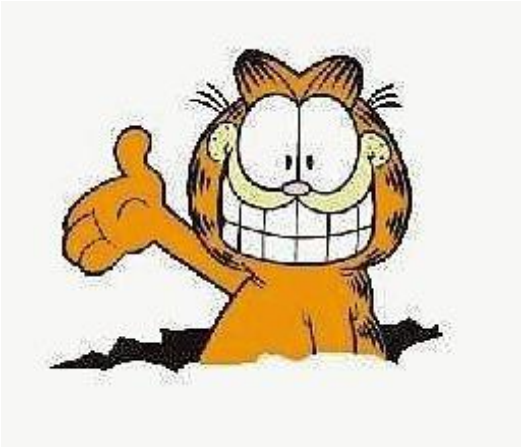
Easier task.

$f(k)$ $f(k)$ $f(k)$

connected components

k-Path Transversal

Easier task – sufficient!



k-Path Transversal

Easier task – sufficient!

Why? Let S be a set of size $c \cdot \log k \cdot opt$ s.t. for every connected component C of $G \backslash S$, the size of an optimal solution is $f(k) = k^3$.

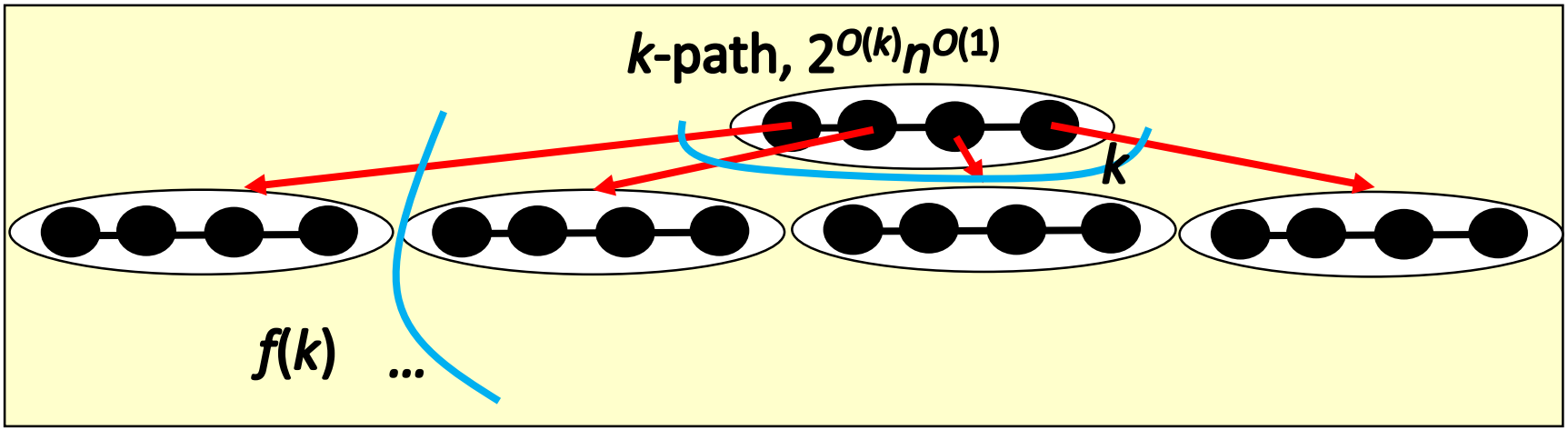
k-Path Transversal

Easier task – sufficient!

Why? Let S be a set of size $c \cdot \log k \cdot opt$ s.t. for every connected component C of $G \backslash S$, the size of an optimal solution is $f(k) = k^3$.

→ For every connected component of $G \backslash S$, find an optimal solution in time $k^{f(k)} n^{O(1)}$.

k-Path Transversal



→ For every connected component of $G \backslash S$, find an optimal solution in time $k^{f(k)} n^{O(1)}$.

k-Path Transversal

Easier task – sufficient!

Why? Let S be a set of size $c \cdot \log k \cdot opt$ s.t. for every connected component C of $G \backslash S$, the size of an optimal solution is $f(k) = k^3$.

→ For every connected component of $G \backslash S$, find an optimal solution in time $k^{f(k)} n^{O(1)}$.

$c \cdot \log k \cdot opt + \sum_C opt_c \leq c \cdot \log k \cdot opt + opt = O(\log k \cdot opt)$.

k-Path Transversal

How to solve the easier task?

k-Path Transversal

How to solve the easier task?

Suppose we have a set $R \subseteq V(G)$ such that

1. **$G \setminus R$ has no k-path and $|R| \leq k \cdot opt$.** (Hint: ILP 😉)

k-Path Transversal

How to solve the easier task?

Suppose we have a set $R \subseteq V(G)$ such that

1. $G \setminus R$ has no k-path and $|R| \leq k \cdot opt$. (Hint: ILP)
2. There is a set $S \subseteq V(G)$ such that
   - $|S| = O(|R|/k)$.
   - Every connected component of $G \setminus S$ has at most k^3 vertices from R.

k-Path Transversal

How to solve the easier task?

Suppose we have a set $R \subseteq V(G)$ such that

1. **$G \setminus R$ has no k-path and $|R| \leq k \cdot opt$. (Hint: ILP)**
2. **There is a set $S \subseteq V(G)$ such that**
   - **$|S| = O(|R|/k)$. → $|S| = O(opt)$!**
   - **Every connected component of $G \setminus S$ has at most k^3 vertices from R.**

k-Path Transversal

How to solve the easier task?

Suppose we have a set $R \subseteq V(G)$ such that

1. $G \setminus R$ has no k-path and $|R| \leq k \cdot opt$. (Hint: ILP)
2. There is a set $S \subseteq V(G)$ such that
   - $|S| = O(|R|/k)$. → $|S| = O(opt)$!
   - Every connected component of $G \setminus S$ has at most k^3 vertices from R. → The size of its optimal solution is at most k^3 !

k-Path Transversal

How to solve the easier task?

Suppose we have a set $R \subseteq V(G)$ such that

1. **$G \setminus R$ has no k-path and $|R| \leq k \cdot opt$. (Hint: ILP)**
2. **There is a set $S \subseteq V(G)$ such that**
   - **$|S| = O(|R|/k)$. → $|S| = O(opt)$!**
   - **Every connected component of $G \setminus S$ has at most k^3 vertices from R. → The size of its optimal solution is at most k^3 !**

If we get S' of size $O(\log k \cdot |S|)$, we are done!

k-Path Transversal

How to solve the easier task?

Suppose we have a set $R \subseteq V(G)$ such that …

(If we get S' of size $O(\log k \cdot |S|)$, we are done!)

k-Path Transversal

How to solve the easier task?

Suppose we have a set $R \subseteq V(G)$ such that …
(If we get S' of size $O(\log k \cdot |S|)$, we are done!)

Given a graph G, a set $R \subseteq V(G)$ and an integer k', find S of minimum size such that each connected component of $G \setminus S$ has at most k' vertices of R.

k-Path Transversal

How to solve the easier task?

Suppose we have a set $R \subseteq V(G)$ such that …
(If we get S' of size $O(\log k \cdot |S|)$, we are done!)

Given a graph G, a set $R \subseteq V(G)$ and an integer k', find S of minimum size such that each connected component of $G \setminus S$ has at most k' vertices of R.

***k*-Subset Vertex Separator:** Has an $O(\log k)$-approx. algorithm with running time $2^{O(k)} n^{O(1)}$. (Spreading metrics, low-diameter decomposition.)

k-Path Transversal

How to find R?

$$
\begin{aligned}
\text{minimize} \quad & \sum_{v \in V} x_v \\
\text{subject to} \quad & \sum_{i=1}^{k} x_{v_i} \geq 1 \qquad \forall P = (v_1, \ldots, v_k) \in \mathcal{P}_k \\
& x \geq 0
\end{aligned}
$$

k-Path Transversal

How to find R?

$$
\begin{aligned}
\text{minimize} \quad & \sum_{v \in V} x_v \\
\text{subject to} \quad & \sum_{i=1}^{k} x_{v_i} \geq 1 \qquad \forall P = (v_1, \ldots, v_k) \in \mathcal{P}_k \\
& x \geq 0
\end{aligned}
$$

$R = \{v \in V(G) : x_v \geq 1/k\}$.

k-Path Transversal

How to find R?

$$
\begin{aligned}
\text{minimize} \quad & \sum_{v \in V} x_v \\
\text{subject to} \quad & \sum_{i=1}^{k} x_{v_i} \geq 1 \qquad \forall P = (v_1, \ldots, v_k) \in \mathcal{P}_k \\
& x \geq 0
\end{aligned}
$$

$R = \{v \in V(G) : x_v \geq 1/k\}$.

$G \setminus R$ has no k-path and $|R| \leq k \cdot frac \leq k \cdot opt$. ✓

k-Path Transversal

How to find R?

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V} x_v & \\ \text{subject to} & \sum_{i=1}^{k} x_{v_i} \geq 1 & \forall P = (v_1, \ldots, v_k) \in \mathcal{P}_k \\ & x \geq 0 & \end{array}$$

$R = \{v \in V(G) : x_v \geq 1/k\}$.

To solve the LP: Use an $2^{O(k)} n^{O(1)}$–time algorithm for Weighted k-Path.

k-Path Transversal

Recall:

We need to prove that $R=\{v\in V(G): x_v\geq 1/k\}$ satisfies

1. $G\setminus R$ has no k-path and $|R|\leq k\cdot opt$. ✓
2. There is a set $S\subseteq V(G)$ such that
   - $|S| = O(|R|/k)$.
   - Every connected component of $G\setminus S$ has at most k^3 vertices from R.

k-Path Transversal

Proof. Let S^* be an optimal solution.

k-Path Transversal

Proof. Let S^* be an optimal solution.

As long as $G \backslash S^*$ has a connected component C with more than k^3 vertices of R:

k-Path Transversal

Proof. Let S^* be an optimal solution.

As long as $G \backslash S^*$ has a connected component C with more than k^3 vertices of R: Add the vertices of some longest path P in C to S^*.

k-Path Transversal

Proof. Let S^* be an optimal solution.
As long as $G \backslash S^*$ has a connected component C with more than k^3 vertices of R: Add the vertices of some longest path P in C to S^*.

Eventually, $G \backslash S^*$ has no comps with $> k^3$ red vertices.

k-Path Transversal

Proof. Let S^* be an optimal solution.
As long as $G \backslash S^*$ has a connected component C with more than k^3 vertices of R: Add the vertices of some longest path P in C to S^*.

Eventually, $G \backslash S^*$ has no comps with $> k^3$ red vertices.
But why $|S^*| = O(|R|/k)$?

k-Path Transversal

Proof. Let S^* be an optimal solution.

As long as $G \backslash S^*$ has a connected component C with more than k^3 vertices of R: Add the vertices of some longest path P in C to S^*, **and charge the cost of P ($|V(P)|$) to all red vertices in C uniformly.**

k-Path Transversal

Proof. Let S^* be an optimal solution.

As long as $G \backslash S^*$ has a connected component C with more than k^3 vertices of R: Add the vertices of some longest path P in C to S^*, **and charge the cost of P ($|V(P)|$) to all red vertices in C uniformly.**

(As P has less than k vertices, each red vertex in C gets charged at most $1/k^2$.)

k-Path Transversal

Proof. Let S^* be an optimal solution.

As long as $G \backslash S^*$ has a connected component C with more than k^3 vertices of R: Add the vertices of some longest path P in C to S^*, **and charge the cost of P ($|V(P)|$) to all red vertices in C uniformly.**

(As P has less than k vertices, each red vertex in C gets charged at most $1/k^2$.)

<u>**If**</u> every red vertex is charged $\leq k$ times

k-Path Transversal

Proof. Let S^* be an optimal solution.

As long as $G \backslash S^*$ has a connected component C with more than k^3 vertices of R: Add the vertices of some longest path P in C to S^*, **and charge the cost of P ($|V(P)|$) to all red vertices in C uniformly.**

(As P has less than k vertices, each red vertex in C gets charged at most $1/k^2$.)

<u>If</u> every red vertex is charged $\leq k$ times $\rightarrow$ every red vertex gets charged $\leq 1/k$

k-Path Transversal

Proof. Let S^* be an optimal solution.
As long as $G \backslash S^*$ has a connected component C with more than k^3 vertices of R: Add the vertices of some longest path P in C to S^*, **and charge the cost of P ($|V(P)|$) to all red vertices in C uniformly.**
(As P has less than k vertices, each red vertex in C gets charged at most $1/k^2$.)

<u>**If**</u> every red vertex is charged $\leq k$ times $\rightarrow$ every red vertex gets charged $\leq 1/k$ $\rightarrow$ total charge $\leq |R|/k$.

k-Path Transversal

Proof. Let S^* be an optimal solution.

As long as $G \backslash S^*$ has a connected component C with more than k^3 vertices of R: Add the vertices of some longest path P in C to S^*, **and charge the cost of P ($|V(P)|$) to all red vertices in C uniformly.**

(As P has less than k vertices, each red vertex in C gets charged at most $1/k^2$.)

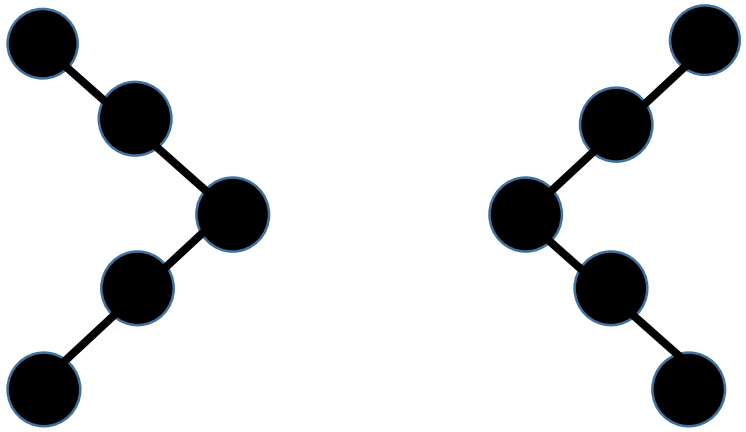
Remains to show: Each red vertex is charged $\leq k$ times.

k-Path Transversal

Observation. Let P be a longest path in a connected graph G. Then, the number of vertices on a longest path in $G \backslash P$ is at most $|V(P)|-1$.

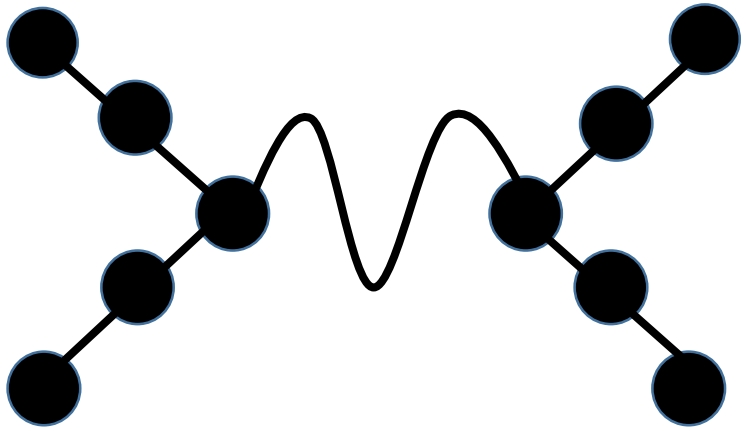
k-Path Transversal

Observation. Let P be a longest path in a connected graph G. Then, the number of vertices on a longest path in $G \backslash P$ is at most $|V(P)|-1$.



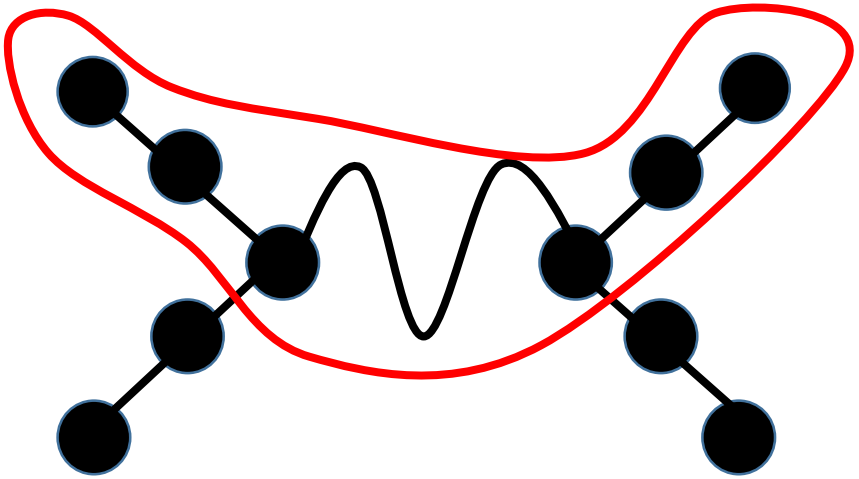
k-Path Transversal

Observation. Let P be a longest path in a connected graph G. Then, the number of vertices on a longest path in $G \backslash P$ is at most $|V(P)|-1$.



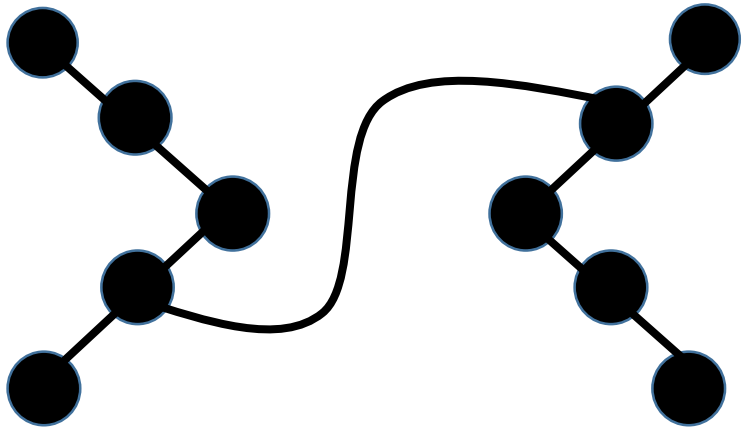
k-Path Transversal

Observation. Let P be a longest path in a connected graph G. Then, the number of vertices on a longest path in $G \backslash P$ is at most $|V(P)|-1$.



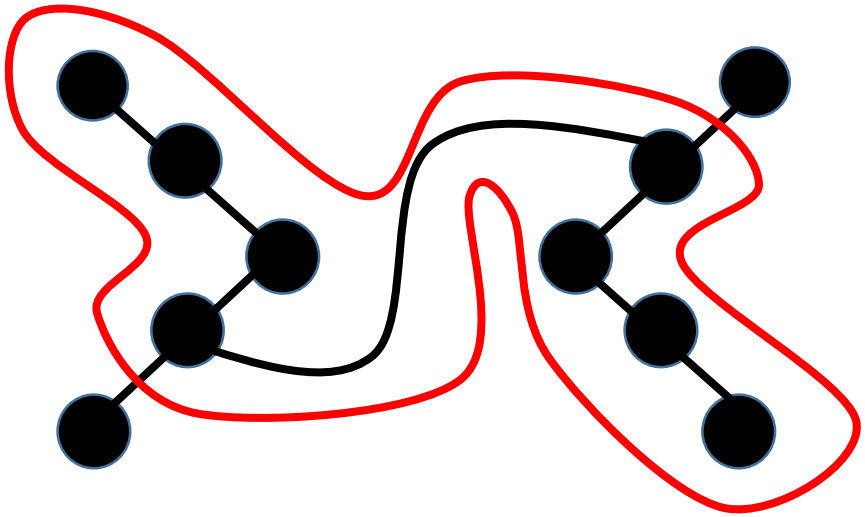
k-Path Transversal

Observation. Let P be a longest path in a connected graph G. Then, the number of vertices on a longest path in $G \backslash P$ is at most $|V(P)|-1$.

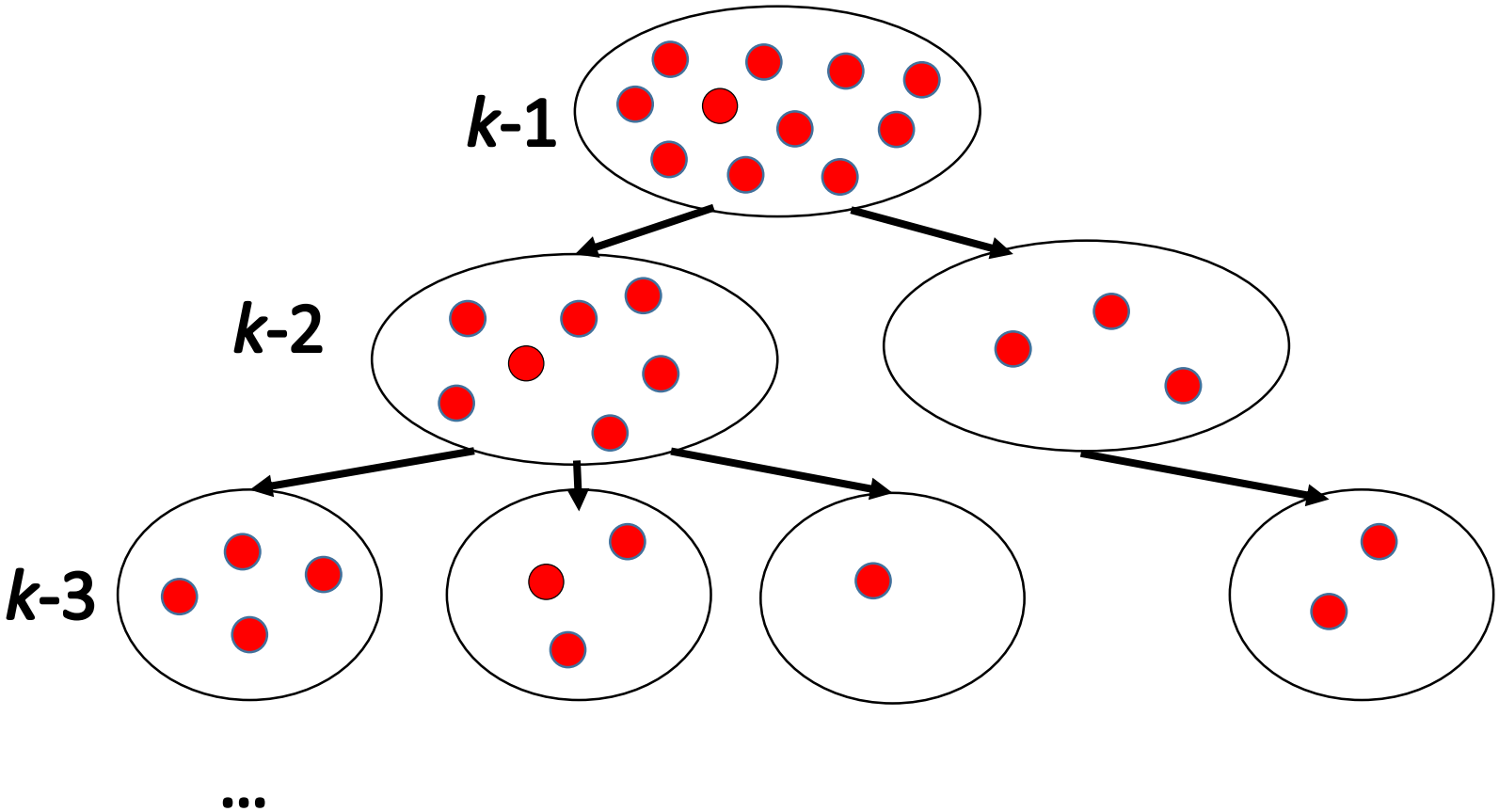


k-Path Transversal

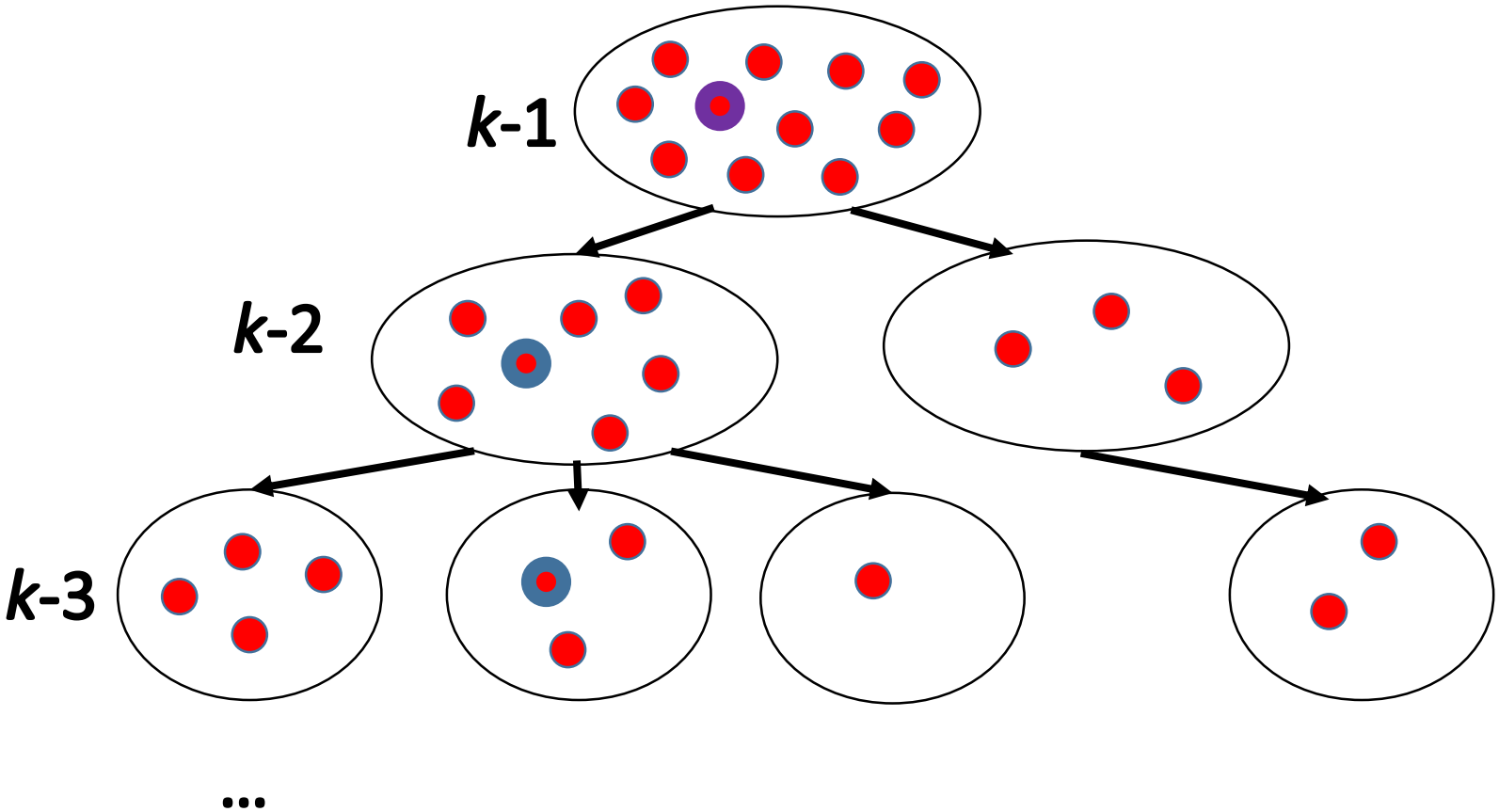
Observation. Let P be a longest path in a connected graph G. Then, the number of vertices on a longest path in $G \backslash P$ is at most $|V(P)|-1$.



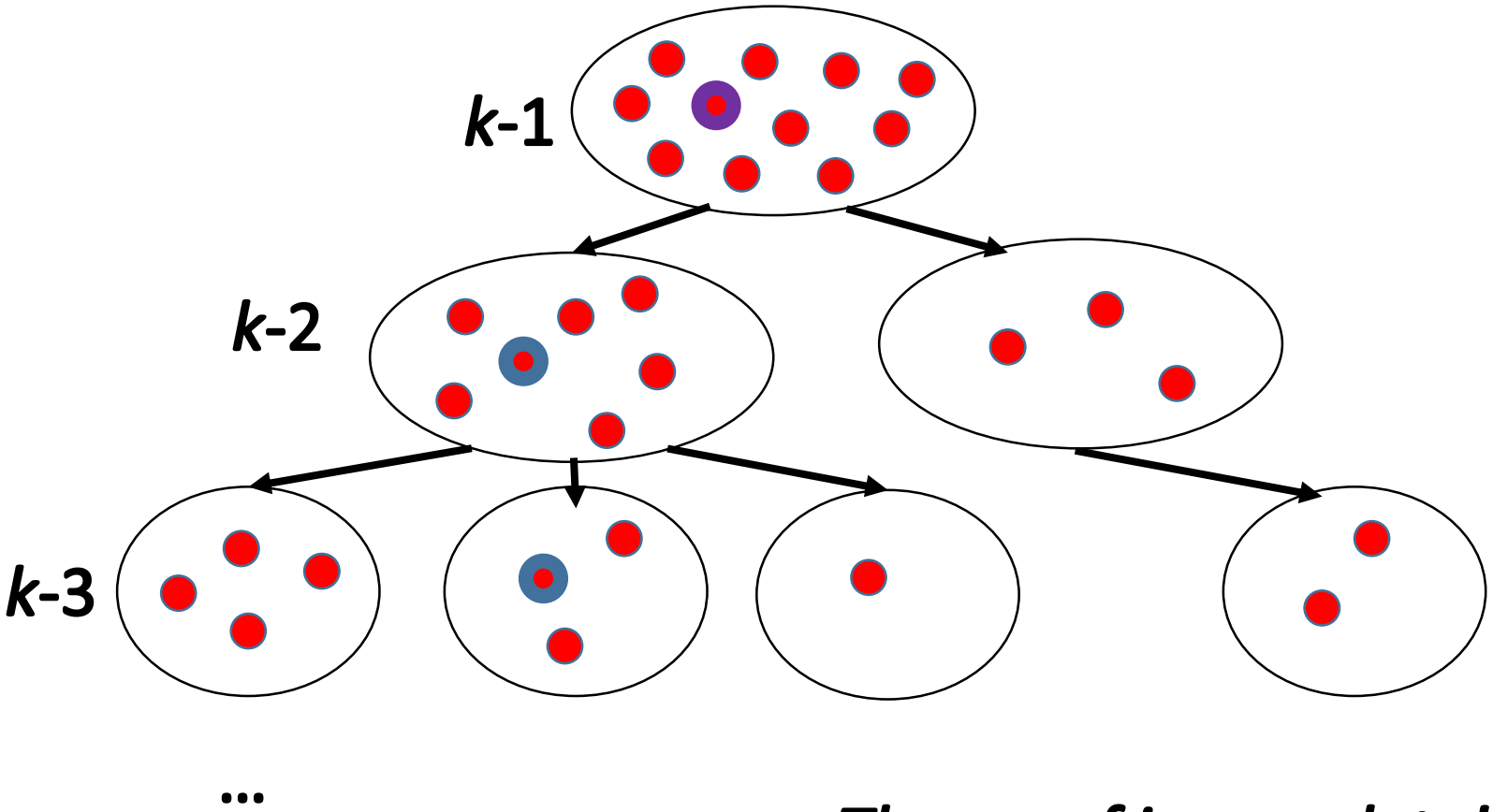
k-Path Transversal



k-Path Transversal



k-Path Transversal



The proof is complete!

Questions?