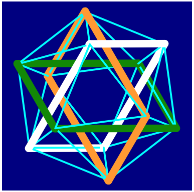
Planar k-Path in Subexponential Time and Polynomial Space

SAKET SAURABH

The Institute of Mathematical Sciences, India



(JOINT WORK WITH DANIEL LOKSHTANOV AND MATTHIAS MNICH)

WG, 21st June 2011

Problem

k-PATH
Input: *A graph G and an integer k.*
Question: *Does there exist a path of length at least k?*

Objective: To obtain an algorithm with running time $2^{o(k)}n^{O(1)}$ and space polynomial in n on planar graphs.
[Disclaimer:] Throughout the talk we will focus on exponential dependence on k and ignore polynomial time.

Problem

k-PATH
Input: *A graph* G *and an integer* k*.*
Question: *Does there exist a path of length at least* k*?*

Objective: To obtain an algorithm with running time $2^{o(k)}n^{O(1)}$ and space polynomial in n on planar graphs.
[Disclaimer:] Throughout the talk we will focus on exponential dependence on k and ignore polynomial time.

Problem

k-PATH
Input: *A graph G and an integer k.*
Question: *Does there exist a path of length at least k?*

Objective: To obtain an algorithm with running time $2^{o(k)}n^{O(1)}$ and space polynomial in n on planar graphs.
[Disclaimer:] Throughout the talk we will focus on exponential dependence on k and ignore polynomial time.

Motivation or Why?

- Time and Space are two main resources in Algorithm Design.
- algorithms that use exponential time and space tend to run out of space long before they run out of time... so it make sense to even settle for slightly slower algorithms if the space uses is reduced drastically
- Quote from Woeginger survey paper on exponential time algorithms: "algorithms with exponential space complexities are absolutely useless for real life applications"..
- Combinatorically Interesting problem

Motivation or Why?

- Time and Space are two main resources in Algorithm Design.
- algorithms that use exponential time and space tend to run out of space long before they run out of time... so it make sense to even settle for slightly slower algorithms if the space uses is reduced drastically
- Quote from Woeginger survey paper on exponential time algorithms: "algorithms with exponential space complexities are absolutely useless for real life applications"..
- Combinatorically Interesting problem

Motivation or Why?

- Time and Space are two main resources in Algorithm Design.
- algorithms that use exponential time and space tend to run out of space long before they run out of time... so it make sense to even settle for slightly slower algorithms if the space uses is reduced drastically
- Quote from Woeginger survey paper on exponential time algorithms: "algorithms with exponential space complexities are absolutely useless for real life applications"..
- Combinatorically Interesting problem

Motivation or Why?

- Time and Space are two main resources in Algorithm Design.
- algorithms that use exponential time and space tend to run out of space long before they run out of time... so it make sense to even settle for slightly slower algorithms if the space uses is reduced drastically
- Quote from Woeginger survey paper on exponential time algorithms: "algorithms with exponential space complexities are absolutely useless for real life applications"..
- Combinatorically Interesting problem

Issue of Space In Parameterized Algorithms

- It has been largely ignored
- Lately some papers have been written on this front:
  - Fomin, Grandoni and Kratsch gave a $6^k n^{O(\log k)}$ time, polynomial space algorithm for the STEINER TREE problem
  - In a breakthrough paper, Nederlof gave a $2^k n^{O(1)}$ time polynomial space algorithm for STEINER TREE
  - Finally Lokshtanov and Nederlof devised general sufficient conditions for turning exponential space dynamic programming algorithms into polynomial space algorithms based on algebraic transforms.
  - Also inclusion-exclusion or randomized algorithms for k-PATH on general graphs runs in polynomial space.

Issue of Space In Parameterized Algorithms

- It has been largely ignored
- Lately some papers have been written on this front:
  - Fomin, Grandoni and Kratsch gave a $6^k n^{O(\log k)}$ time, polynomial space algorithm for the STEINER TREE problem
  - In a breakthrough paper, Nederlof gave a $2^k n^{O(1)}$ time polynomial space algorithm for STEINER TREE
  - Finally Lokshtanov and Nederlof devised general sufficient conditions for turning exponential space dynamic programming algorithms into polynomial space algorithms based on algebraic transforms.
  - Also inclusion-exclusion or randomized algorithms for k-PATH on general graphs runs in polynomial space.

Issue of Space In Parameterized Algorithms

- It has been largely ignored
- Lately some papers have been written on this front:
  - Fomin, Grandoni and Kratsch gave a $6^k n^{O(\log k)}$ time, polynomial space algorithm for the STEINER TREE problem
  - In a breakthrough paper, Nederlof gave a $2^k n^{O(1)}$ time polynomial space algorithm for STEINER TREE
  - Finally Lokshtanov and Nederlof devised general sufficient conditions for turning exponential space dynamic programming algorithms into polynomial space algorithms based on algebraic transforms.
  - Also inclusion-exclusion or randomized algorithms for k-PATH on general graphs runs in polynomial space.

Issue of Space In Parameterized Algorithms

- It has been largely ignored
- Lately some papers have been written on this front:
  - Fomin, Grandoni and Kratsch gave a $6^k n^{O(\log k)}$ time, polynomial space algorithm for the STEINER TREE problem
  - In a breakthrough paper, Nederlof gave a $2^k n^{O(1)}$ time polynomial space algorithm for STEINER TREE
  - Finally Lokshtanov and Nederlof devised general sufficient conditions for turning exponential space dynamic programming algorithms into polynomial space algorithms based on algebraic transforms.
  - Also inclusion-exclusion or randomized algorithms for k-PATH on general graphs runs in polynomial space.

Issue of Space In Parameterized Algorithms

- It has been largely ignored
- Lately some papers have been written on this front:
    - Fomin, Grandoni and Kratsch gave a $6^k n^{O(\log k)}$ time, polynomial space algorithm for the Steiner Tree problem
    - In a breakthrough paper, Nederlof gave a $2^k n^{O(1)}$ time polynomial space algorithm for Steiner Tree
    - Finally Lokshtanov and Nederlof devised general sufficient conditions for turning exponential space dynamic programming algorithms into polynomial space algorithms based on algebraic transforms.
    - Also inclusion-exclusion or randomized algorithms for k-Path on general graphs runs in polynomial space.

Our Objective

- Look at the parameterized problems that take exponential space and see if we can make it run in polynomial space.
- A large chunk of these parameterized problems that utilize exponential space has an algorithm on graphs of bounded treewidth as subroutine.
- And most notable ones here are the so called bidimensional problems

Our Objective

- Look at the parameterized problems that take exponential space and see if we can make it run in polynomial space.
- A large chunk of these parameterized problems that utilize exponential space has an algorithm on graphs of bounded treewidth as subroutine.
- And most notable ones here are the so called bidimensional problems

Our Objective

- Look at the parameterized problems that take exponential space and see if we can make it run in polynomial space.
- A large chunk of these parameterized problems that utilize exponential space has an algorithm on graphs of bounded treewidth as subroutine.
- And most notable ones here are the so called bidimensional problems

Minors and contractions

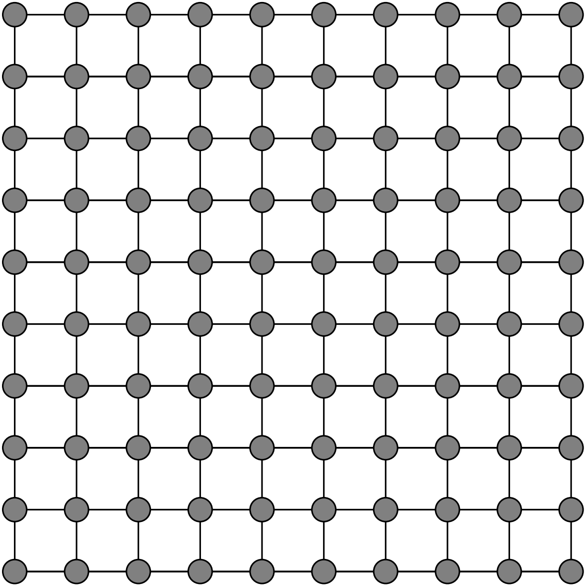
H is a contraction of G ($H \leq_c G$) if H occurs from G after applying a series of edge contractions.

H is a minor of G ($H \leq_m G$) if H is the contraction of some subgraph of G.

Bidimensional Problems (BP)

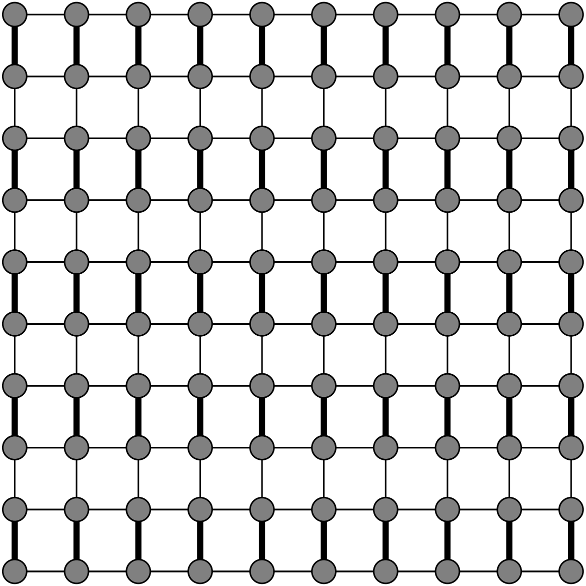
- These are either minor closed/contraction closed — essentially solution does not increase when we take the minor/contraction of the graph — example FEEDBACK VERTEX SET, VERTEX COVER, DOMINATING SET, ...
- In a nutshell these are the problems that have $\Omega(k^2)$ solution on $k \times k$ grid like graph.

Planar k-Vertex Cover



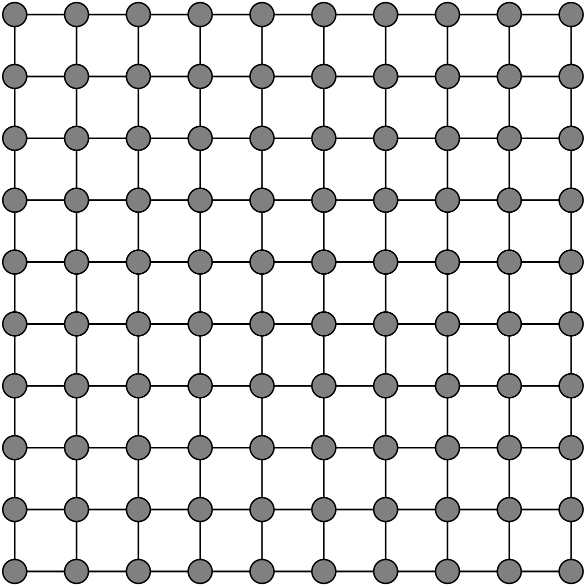
$H_{r,r}$ for $r = 10$

Planar k-Vertex Cover

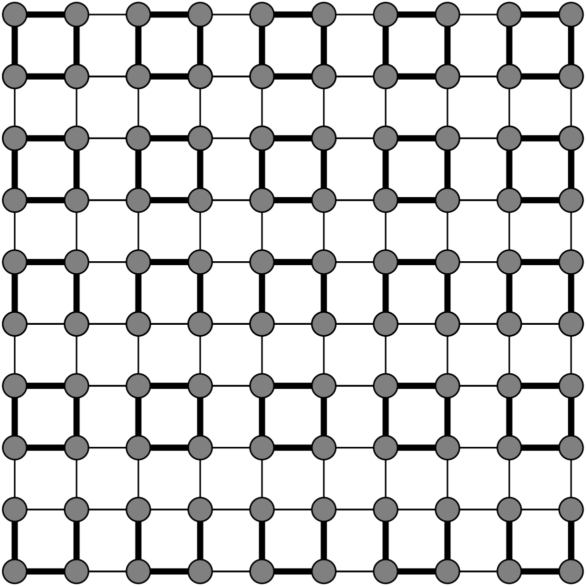


$\mathbf{vc}(H_{r,r}) \geq \frac{r^2}{2}$

k-Feedback Vertex Set



k-Feedback Vertex Set



$\mathbf{fvc}(H_{r,r}) \geq \frac{r^2}{4}$

How to Obtain Subexponential Algorithms for BP?

- First we must restrict ourselves to special graph classes like *planar or H minor free graphs.*
  - Show that if the graph has large treewidth $(> c\sqrt{k})$ then it has $\sqrt{k} \times \sqrt{k}$ grid like graph as a minor/contraction and hence answer is YES or NO immediately
  - Else treewidth is bounded and hence we can use the dynamic programming algorithm on graphs of bounded treewidth.
- If we have c^t or t^t treewidth algorithm then it implies $2^{O(\sqrt{k})}$ or $2^{o(k)}$ algorithm.

How to Obtain Subexponential Algorithms for BP?

- First we must restrict ourselves to special graph classes like *planar or H minor free graphs.*
  - Show that if the graph has large treewidth $(> c\sqrt{k})$ then it has $\sqrt{k} \times \sqrt{k}$ grid like graph as a minor/contraction and hence answer is YES or NO immediately
  - Else treewidth is bounded and hence we can use the dynamic programming algorithm on graphs of bounded treewidth.
- If we have c^t or t^t treewidth algorithm then it implies $2^{O(\sqrt{k})}$ or $2^{o(k)}$ algorithm.

How to Obtain Subexponential Algorithms for BP?

- First we must restrict ourselves to special graph classes like *planar or H minor free graphs.*
  - Show that if the graph has large treewidth $(> c\sqrt{k})$ then it has $\sqrt{k} \times \sqrt{k}$ grid like graph as a minor/contraction and hence answer is YES or NO immediately
  - Else treewidth is bounded and hence we can use the dynamic programming algorithm on graphs of bounded treewidth.
- If we have c^t or t^t treewidth algorithm then it implies $2^{O(\sqrt{k})}$ or $2^{o(k)}$ algorithm.

How to Obtain polyspace Subexponential Algorithms for BP?

- We need algorithms on graphs of bounded treewidth that runs in time $t^{O(t)}$ and in polynomial space
- We do not know how to make such algorithm even for VERTEX COVER parameterized by treewidth.
- [Digression] What about $f(t)poly(n)$ algorithm for VERTEX COVER parameterized by treewidth that runs in polynomial space?

How to Obtain polyspace Subexponential Algorithms for BP?

- We need algorithms on graphs of bounded treewidth that runs in time $t^{O(t)}$ and in polynomial space
- We do not know how to make such algorithm even for VERTEX COVER parameterized by treewidth.
- [Digression] What about $f(t)poly(n)$ algorithm for VERTEX COVER parameterized by treewidth that runs in polynomial space?

How to Obtain polyspace Subexponential Algorithms for BP?

- We need algorithms on graphs of bounded treewidth that runs in time $t^{O(t)}$ and in polynomial space
- We do not know how to make such algorithm even for VERTEX COVER parameterized by treewidth.
- [Digression] What about $f(t)poly(n)$ algorithm for VERTEX COVER parameterized by treewidth that runs in polynomial space?

Treewidth and balanced separator

- For a set $W \subseteq V(G)$ a set $S \subseteq V(G)$ is a *balanced separator* for W if $V(G)$ can be partitioned into L, S and R such that there is no edge from L to R and $|W \cap L| \leq \frac{2|W \setminus S|}{3}$ and $|W \cap R| \leq \frac{2|W \setminus S|}{3}$.
- In other words, W is evenly distributed between L and R.
- It is well-known that in any *tree* T, for every set $W \subseteq V(G)$ there is a balanced separator S for W with $|S| = 1$. This result has been generalized to graphs of bounded treewidth - in particular in a graph G of treewidth at most t, for any set W there is a balanced separator S of size at most $t+1$.

Treewidth and balanced separator

- For a set $W \subseteq V(G)$ a set $S \subseteq V(G)$ is a *balanced separator* for W if $V(G)$ can be partitioned into L, S and R such that there is no edge from L to R and $|W \cap L| \leq \frac{2|W \setminus S|}{3}$ and $|W \cap R| \leq \frac{2|W \setminus S|}{3}$.
- In other words, W is evenly distributed between L and R.
- It is well-known that in any *tree* T, for every set $W \subseteq V(G)$ there is a balanced separator S for W with $|S| = 1$. This result has been generalized to graphs of bounded treewidth - in particular in a graph G of treewidth at most t, for any set W there is a balanced separator S of size at most $t + 1$.

Treewidth and balanced separator

- For a set $W \subseteq V(G)$ a set $S \subseteq V(G)$ is a *balanced separator* for W if $V(G)$ can be partitioned into L, S and R such that there is no edge from L to R and $|W \cap L| \leq \frac{2|W \setminus S|}{3}$ and $|W \cap R| \leq \frac{2|W \setminus S|}{3}$.
- In other words, W is evenly distributed between L and R.
- It is well-known that in any *tree* T, for every set $W \subseteq V(G)$ there is a balanced separator S for W with $|S| = 1$. This result has been generalized to graphs of bounded treewidth - in particular in a graph G of treewidth at most t, for any set W there is a balanced separator S of size at most $t + 1$.

Poly Space Time $2^{O(t^2)} poly(n)$ algorithm for Vertex Cover parameterized by treewidth

- Let L, S and R be the balanced separator corresponding to $W = V(G)$.
- Guess all possible ways a vertex cover can interact with S. Then for each possibility solve the problem recursively in both sides. This gives us:

$$T(n) \leq 2^t 2T(2n/3) \sim 2^{O(t \log n)} \sim n^{O(t)}$$

- We also have 2^t time and space algorithm.

Poly Space Time $2^{O(t^2)}poly(n)$ algorithm for Vertex Cover parameterized by treewidth

- Let L, S and R be the balanced separator corresponding to $W = V(G)$.
- Guess all possible ways a vertex cover can interact with S. Then for each possibility solve the problem recursively in both sides. This gives us:

$$T(n) \leq 2^t 2T(2n/3) \sim 2^{O(t \log n)} \sim n^{O(t)}$$

- We also have 2^t time and space algorithm.

Poly Space Time $2^{O(t^2)} poly(n)$ algorithm for VERTEX COVER parameterized by treewidth

- Let L, S and R be the balanced separator corresponding to $W = V(G)$.
- Guess all possible ways a vertex cover can interact with S. Then for each possibility solve the problem recursively in both sides. This gives us:

$$T(n) \leq 2^t 2T(2n/3) \sim 2^{O(t \log n)} \sim n^{O(t)}$$

- We also have 2^t time and space algorithm.

Poly Space Time $2^{O(t^2)} poly(n)$ algorithm for Vertex Cover parameterized by treewidth

So we have:

- $\mathcal{A}$ — $n^{O(t)}$ polynomial space algorithm;
- $\mathcal{B}$ — 2^t time and space algorithm.
- Now if $n \leq 2^t$ then $\mathcal{B}$ runs in polynomial time and space and when $n > 2^t$ then $\mathcal{A}$ runs in time $2^{O(t^2)}$ and space polynomial in n.

Open Question: Does there exist a polynomial space algorithm for Vertex Cover parameterized by treewidth that runs in time $O^*(2^{O(t^{2-\varepsilon})})$ for some fixed $\varepsilon > 0$?

Poly Space Time $2^{O(t^2)} poly(n)$ algorithm for Vertex Cover parameterized by treewidth

So we have:

- $\mathcal{A}$ — $n^{O(t)}$ polynomial space algorithm;
- $\mathcal{B}$ — 2^t time and space algorithm.
- Now if $n \leq 2^t$ then $\mathcal{B}$ runs in polynomial time and space and when $n > 2^t$ then $\mathcal{A}$ runs in time $2^{O(t^2)}$ and space polynomial in n.

Open Question: Does there exist a polynomial space algorithm for Vertex Cover parameterized by treewidth that runs in time $O^*(2^{O(t^{2-\varepsilon})})$ for some fixed $\varepsilon > 0$?

Poly Space Time $2^{O(t^2)} poly(n)$ algorithm for Vertex Cover parameterized by treewidth

So we have:

- $\mathcal{A}$ — $n^{O(t)}$ polynomial space algorithm;
- $\mathcal{B}$ — 2^t time and space algorithm.
- Now if $n \leq 2^t$ then $\mathcal{B}$ runs in polynomial time and space and when $n > 2^t$ then $\mathcal{A}$ runs in time $2^{O(t^2)}$ and space polynomial in n.

Open Question: Does there exist a polynomial space algorithm for Vertex Cover parameterized by treewidth that runs in time $O^*(2^{O(t^{2-\varepsilon})})$ for some fixed $\varepsilon > 0$?

Another Approach

- First obtain a linear (or polynomial kernel) in polynomial time – that is obtain an equivalent instance with $O(k)$ vertices (or $k^{O(1)}$ vertices) — Most bidimensioanl problems do have such kernel.
- Bound the treewidth as before with $O(\sqrt{k})$.
- Now the $n^{O(t)}$ algorithm runs in time $k^{O(\sqrt{k})} = 2^{O(\sqrt{k}\log k)}$.

Another Approach

- First obtain a linear (or polynomial kernel) in polynomial time – that is obtain an equivalent instance with $O(k)$ vertices (or $k^{O(1)}$ vertices) — Most bidimensioanl problems do have such kernel.
- Bound the treewidth as before with $O(\sqrt{k})$.
- Now the $n^{O(t)}$ algorithm runs in time $k^{O(\sqrt{k})} = 2^{O(\sqrt{k} \log k)}$.

Another Approach

- First obtain a linear (or polynomial kernel) in polynomial time – that is obtain an equivalent instance with $O(k)$ vertices (or $k^{O(1)}$ vertices) — Most bidimensioanl problems do have such kernel.
- Bound the treewidth as before with $O(\sqrt{k})$.
- Now the $n^{O(t)}$ algorithm runs in time $k^{O(\sqrt{k})} = 2^{O(\sqrt{k}\log k)}$.

Back to k-PATH

- Do not know how to make good ($t^{O(t)}$) polynomial space algorithm on graphs of bounded treewidth.
- Does not have polynomial kernel even on planar graphs unless polynomial hierarchy collapses to second level.

Back to k-Path

- Do not know how to make good ($t^{O(t)}$) polynomial space algorithm on graphs of bounded treewidth.
- Does not have polynomial kernel even on planar graphs unless polynomial hierarchy collapses to second level.

An Observation

For a graph G and a nice tree-decomposition $(T, \mathcal{B})$ of G and node $v \in V(T)$ let $X_v \in \mathcal{B}$ be the corresponding bag. Let T_v be the subtree of T rooted at v and let $A(v) = (\cup_{u \in V(T_v)} X_u) \setminus X_v$.

Let G be a graph, let $(T, \mathcal{B})$ be a nice tree-decomposition of G of width t and let $W \subseteq V$ be a vertex set of size at least 3.

Conclusion: Then there exists a vertex v such that X_v is a balanced separator for W and in the corresponding partition $V(G) = L \cup X_v \cup R$, $L = A(v)$.

An Observation

For a graph G and a nice tree-decomposition $(T, \mathcal{B})$ of G and node $v \in V(T)$ let $X_v \in \mathcal{B}$ be the corresponding bag. Let T_v be the subtree of T rooted at v and let $A(v) = (\cup_{u \in V(T_v)} X_u) \setminus X_v$.

Let G be a graph, let $(T, \mathcal{B})$ be a nice tree-decomposition of G of width t and let $W \subseteq V$ be a vertex set of size at least 3.

Conclusion: Then there exists a vertex v such that X_v is a balanced separator for W and in the corresponding partition $V(G) = L \cup X_v \cup R$, $L = A(v)$.

An Observation

For a graph G and a nice tree-decomposition $(T, \mathcal{B})$ of G and node $v \in V(T)$ let $X_v \in \mathcal{B}$ be the corresponding bag. Let T_v be the subtree of T rooted at v and let $A(v) = (\cup_{u \in V(T_v)} X_u) \setminus X_v$.

Let G be a graph, let $(T, \mathcal{B})$ be a nice tree-decomposition of G of width t and let $W \subseteq V$ be a vertex set of size at least 3.
Conclusion: Then there exists a vertex v such that X_v is a balanced separator for W and in the corresponding partition $V(G) = L \cup X_v \cup R$, $L = A(v)$.

Algorithm Outline for k-Path

- The lemma said that given a tree-decomposition of width t and any set W, there is a bag that separates it in balance way!
- [Oversimplified Idea] Set W as the set of vertices of the k path. Now at least one of the bag separates it in balance way. Guess the bag and guess the intersection. This gives us

$$\begin{aligned} T(n,k) &\leq \text{no. of bags} \times \text{guesses} \times \text{time for subproblem} \\ &\leq n \times t^t \times 2T(n, 2k/3) \sim (nt^t)^{O(\log k)} \end{aligned}$$

Algorithm Outline for k-PATH

- The lemma said that given a tree-decomposition of width t and any set W, there is a bag that separates it in balance way!
- [Oversimplified Idea] Set W as the set of vertices of the k path. Now at least one of the bag separates it in balance way. Guess the bag and guess the intersection. This gives us

$$\begin{aligned} T(n,k) &\leq \text{no. of bags} \times \text{guesses} \times \text{time for subproblem} \\ &\leq n \times t^t \times 2T(n, 2k/3) \sim (nt^t)^{O(\log k)} \end{aligned}$$

Subexponential Polynomial space algorithm for k-PATH

So we have:

- $\mathcal{A}$ — $(nt)^{O(t \log k)}$ polynomial space algorithm for k-PATH;
- since $t = O(\sqrt{k})$ we have that $\mathcal{A}$ takes $(n\sqrt{k})^{O(\sqrt{k} \log k)}$
- $\mathcal{B}$ — $2^{O(\sqrt{k})}$ time and space algorithm.
- Now if $n > 2^{\sqrt{k}}$ then $\mathcal{B}$ runs in polynomial time and space and when $n \leq 2^{\sqrt{k}}$ then $\mathcal{A}$ runs in time $2^{O(\sqrt{k} \log^2 k)}$ and space polynomial in n.

This is indeed subexponential polynomial space algorithm for k-PATH.

Subexponential Polynomial space algorithm for k-PATH

So we have:

- $\mathcal{A}$ — $(nt)^{O(t \log k)}$ polynomial space algorithm for k-PATH;
- since $t = O(\sqrt{k})$ we have that $\mathcal{A}$ takes $(n\sqrt{k})^{O(\sqrt{k} \log k)}$
- $\mathcal{B}$ — $2^{O(\sqrt{k})}$ time and space algorithm.
- Now if $n > 2^{\sqrt{k}}$ then $\mathcal{B}$ runs in polynomial time and space and when $n \leq 2^{\sqrt{k}}$ then $\mathcal{A}$ runs in time $2^{O(\sqrt{k} \log^2 k)}$ and space polynomial in n.

This is indeed subexponential polynomial space algorithm for k-PATH.

Subexponential Polynomial space algorithm for k-Path

So we have:

- $\mathcal{A}$ — $(nt)^{O(t \log k)}$ polynomial space algorithm for k-Path;
- since $t = O(\sqrt{k})$ we have that $\mathcal{A}$ takes $(n\sqrt{k})^{O(\sqrt{k} \log k)}$
- $\mathcal{B}$ — $2^{O(\sqrt{k})}$ time and space algorithm.
- Now if $n > 2^{\sqrt{k}}$ then $\mathcal{B}$ runs in polynomial time and space and when $n \leq 2^{\sqrt{k}}$ then $\mathcal{A}$ runs in time $2^{O(\sqrt{k} \log^2 k)}$ and space polynomial in n.

This is indeed subexponential polynomial space algorithm for k-Path.

Open Problems

- Is there a polynomial space parameterized algorithm for the k-PATH problem on planar graphs with running time $2^{O(\sqrt{k})}n^{O(1)}$?
- Does there exist a polynomial space algorithm for VERTEX COVER/INDEPENDENT SET parameterized by treewidth that runs in time $O^*(2^{O(t^{2-\varepsilon})})$ for some fixed $\varepsilon > 0$?

Final Slide

Thank You!

Any Questions?