15-850: Advanced Algorithms **CMU, Fall 2018**
Lecture #13: FPT Algorithms: Treewidth and Planarity September 26, 2018
Lecturer: Jason Li Scribe: Benjamin Lim and Yao Chong Lim

1 Introduction

In this lecture, we will see more examples of fixed-parameter tractable (FPT) algorithms, applied to the Maximum Independent Set problem (MaxIS). While the MaxIS problem is NP-hard in general, we will show an exact algorithm for MaxIS on graphs of treewidth-k that runs in polynomial time in the number of vertices, but exponential time in k. We also give a $(1-\varepsilon)$-approximation algorithm for MaxIS on planar graphs using Baker's technique that also runs in polynomial time in the number of vertices.

2 Maximum Independent Set

Given a graph $G = (V, E)$, an independent set of vertices is some subset of vertices of the graph $S \subseteq V$ such that no two vertices in the set are adjacent. The maximum independent set is the largest possible independent set in the graph.

In general, the problem is NP-hard. However, as before, we wish to find ways to solve the problem more quickly.

2.1 (Lack of) Approximation Algorithms

It turns out that there is no $\frac{1}{n^\varepsilon}$-approximation for MaxIS, for some $\varepsilon > 0$ [Has96]. In general, we wish to find approximation algorithms that give a constant-factor approximation, but this is not possible to do so efficiently for MaxIS, making the problem "inapproximable".

2.2 Input Restriction

As we did for previous FPT algorithms, we can try restricting the input somehow, and hopefully find efficient algorithms for solving MaxIS.

Theorem 13.1. *MaxIS can be solved exactly on treewidth-k graphs in time $2^{O(k)}poly(n)$.*

Theorem 13.2. *A $(1-\epsilon)$-approximation for MaxIS on a planar graph can be found in time $2^{O(1/\epsilon)}poly(n)$. (Finding an exact solution for planar graphs is still NP-hard in general).*

3 Pathwidth and Treewidth

Before giving the MaxIS algorithm for treewidth-k graphs, we will introduce the simpler but similar concept of pathwidth and give an exact MaxIS algorithm for pathwidth-k graphs. The algorithm for treewidth-k graphs will be similar.

3.1 Pathwidth

Before defining pathwidth, it is instructive to see how we can construct a graph with pathwidth-k.

We give an alternative representation of a graph: Each vertex is an interval on the real line, and edges can only exist between vertices whose intervals overlap (but those edges do not necessarily

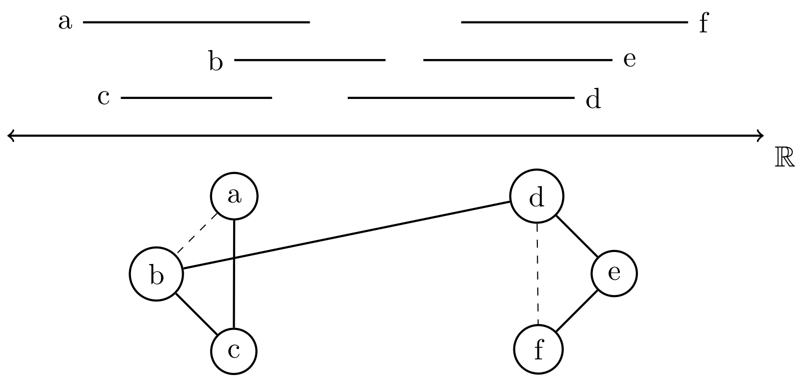


Figure 13.1: Constructing a graph with pathwidth 2.

have to exist). Furthermore, to construct a graph with pathwidth-k, every point on the real line must be contained in at most $k+1$ intervals.

For example, Figure 13.1 shows the construction of a pathwidth-2 graph as every point on the real line falls inside at most 3 intervals. Each edge in the graph also has endpoints whose intervals on the real line overlap. However, even though the intervals representing vertex a and b overlap, there is no edge between them in the actual graph, and similarly for d and f.

Definition 13.3 (Class of pathwidth-k)**.** The class of pathwidth-k graphs is all graphs that can be constructed in this way.

Definition 13.4 (Pathwidth)**.** A graph G has pathwidth-k if k is the smallest integer such that G is in the class of pathwidth-k graphs.

As another example, note that all path/line graphs and star graphs have pathwidth 1, while cycle graphs have pathwidth 2.

To clarify our understanding of pathwidth, we note the following proposition.

Proposition 13.5. *A pathwidth-k graph does not contain a $(k+2)$-clique.*

Proof. Let some graph G have a clique of size $k+2$. Suppose for contradiction that G can be constructed as a pathwidth-k graph.

Let the vertices in the clique be $v_1, v_2, ..., v_{k+2}$. Note that in any construction of G in the real line representation, every pair of intervals representing the $k+2$ vertices must intersect. However, this means that all the intervals must share a common point. Therefore the graph cannot be of pathwidth-k, as there is a point that is contained in all $k+2$ intervals. □

3.2 A 'discrete' view of pathwidth

In the 'real line' view of a graph, we are only concerned with how many, and which intervals each point on the real line is in. Going from left to right on the line, this only changes each time we reach a new endpoint of an interval. This motivates a 'discretization' of the construction above.

For example, in Figure 13.2, we discretize the representation of the path graph on the real line (above), into a sequence of bags of vertices (below). Whenever an interval begins or ends, we create a bag which contains the bags of vertices which have intervals that overlap that point (we tacitly assume that intervals are half-open on the right).

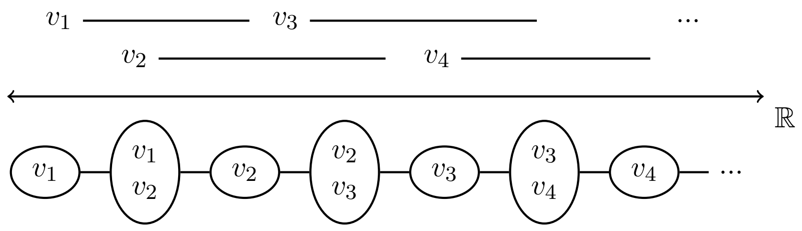


Figure 13.2: A 'discrete' view of the construction of a path graph

Observe that for a pathwidth-k graph, the biggest bag is of size at most $k+1$. Also note that the number of vertices in the graph is polynomial in the number of original vertices (specifically, there are at most $2n-1$ bags), since we only add a new vertex for each endpoint of an interval, and every interval is defined by two endpoints.

We can now define the **path decomposition** of a graph.

Definition 13.6 (Path decomposition)**.** A path decomposition of a graph $G = (V, E)$ is a sequence of subsets of vertices (call them "bags") $(X_1, X_2, ...)$ such that

- Every vertex of G is inside some bag.
- For each edge in G, there is a bag containing both of its endpoints.
- For each vertex, the set of bags containing v must form a connected interval in the sequence. Equivalently, if a vertex u is in bag X_i and X_k, then u must be in all bags X_j where $j \in [i, k]$.

The condition that every vertex of G is inside some bag is primarily to ensure that we do not have graphs with negative pathwidth. Under this definition, a graph containing only isolated vertices is pathwidth-0.

Definition 13.7 (Path decomposition width)**.** The width of a path decomposition is the maximum size of a bag in the decomposition.

Definition 13.8. Let the smallest width over all path decompositions of a graph G be w. Then the pathwidth of G is $w-1$.

3.3 Pathwidth and Maximum Independent Set

Using our understanding of path decompositions, we can now prove the following theorem.

Theorem 13.9. *Given a pathwidth-k decomposition for a graph G, we can find the maximum independent set in time $2^{O(k)}$poly(n).*

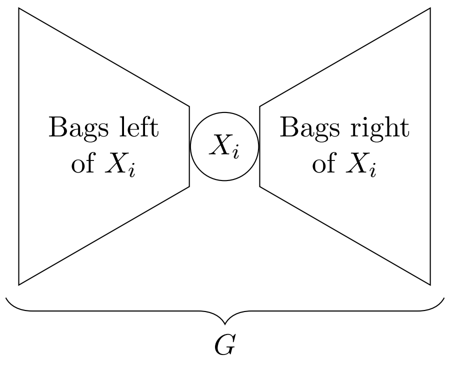


Figure 13.3: Splitting the path decomposition

Proof. For each bag X_i, we can split the path decomposition into three parts, as in Figure 13.3:

- $L = \bigcup_{j<i} X_j$ being the vertices in the bags left of X_i
- X_i itself
- $R = \bigcup_{j>i} X_j$ being the vertices in the bags to the right of X_i

Now note that if $u \in L$ and $v \in R$, then if there is an edge between u and v in the graph G then we must have both u and v in X_i, so every vertex v in R is either not adjacent to every vertex in L or v is in X_i along with some vertex u from L. Thus if S is a maximum independent set, it must be the case that $S \cap (L \cup X_i)$ is also a maximum independent set over the induced subgraph on vertices $L \cup X_i$, and when considering what vertices to add in from X_{i+1} we only need to consider the vertices from the maximum independent set that are in X_i.

Let $f(S, i)$ be the size of (any) maximum independent set containing the vertices $S \subseteq X_i$ (and 0 if no such set exists). We may thus form the recurrence: $f(S, 1) = |S|$ for independent $S \subseteq X_1$, and $f(S, i + 1) = |S| + \max_P |P|$ for independent subsets S of X_{i+1}, and P ranging over subsets of X_i such that $P \cup S$ independent and if $x \in P \cap X_{i+1}$ then $x \in S$. The size of the maximal independent set can just be formed by taking the max over all subsets S of the very last bag X_p of $f(S, p)$, and the actual independent set can be reconstructed easily.

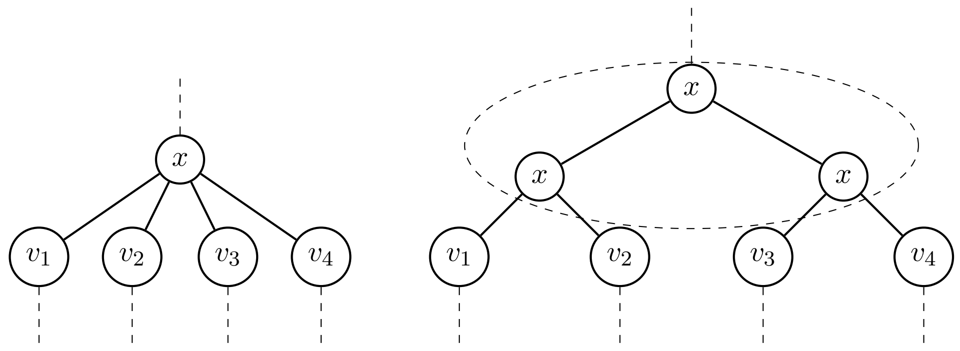
As the pathwidth is k there are 2^k possibilities to consider for each $f(S, i)$ and $2^k poly(n)$ subproblems to consider, giving a runtime of $2^{O(k)} poly(n)$. □

3.4 Treewidth

The tree decomposition of a graph is defined in a similar way as the path decomposition of a graph, except that instead of representing vertices by intervals over the real line, vertices are represented by connected subtrees of some tree T. In other words:

Definition 13.10 (Tree decomposition)**.** A tree decomposition of a graph $G = (V, E)$ is a tree T of bags of vertices $(X_1, X_2, ...)$ such that

- Every vertex of G is inside some bag.
- For each edge in G, there is a bag containing both of its endpoints.
- For each vertex, the set of bags containing v must form a subtree of T.



(a) Before reduction (b) After reduction

Figure 13.4: Reducing the degree of tree decompositions to degree 3.

The relationship between treewidth and tree decomposition width is similar to the relationship between pathwidth and path decomposition width.

Definition 13.11 (Tree decomposition width)**.** The width of a tree decomposition is the maximum size of a bag in the decomposition.

Definition 13.12. Let the smallest width over all tree decompositions of a graph G be w. Then the treewidth of G is $w - 1$.

3.5 Treewidth and Maximum Independent Set

We can extend the DP for pathwidth-bounded graphs to treewidth-bounded graphs. First, we note that any tree decomposition can be converted to an equivalent tree of degree at most 3. This can be done by expanding vertices with degree greater than 3 into a binary tree, with the expanded vertex as internal vertices, and the neighbors as leaves (see Figure 13.4).

Since we do not change the size of any bags, the width of the tree decomposition remains the same. The size of the decomposition is still polynomial in n.

Then, we can extend the DP from 3.3 to tree decompositions with degree at most 3.

Given some tree decomposition with degree at most 3, we can view it as a binary tree rooted at an arbitrary root. Then, for each bag X_i, we can split the tree decomposition into three parts:

- T_i, the vertices in the subtree rooted at X_i excluding X_i
- X_i itself
- S_i, every other vertex not in the bags in the subtree rooted at X_i.

Like for the path decomposition, if $u \in T_i$ and $v \in S_i$, then if there is an edge between u and v in the original graph G then we must have both u and v in X_i, so every vertex in v in S_i is either not adjacent to every vertex in T_i or v is in X_i along with some vertex u from T_i. Thus, when considering what vertices to add to the independent set from some bag X_j, we only need to consider the vertices from the maximum independent set that are in both subtrees of X_j to ensure that we maintain an independent set over the whole subtree rooted at X_j. The recurrence is otherwise similar to the one in 3.3.

As before, when the treewidth is k, there are $2^k poly(n)$ subproblems, but we now need to consider up to 4^k combinations of independent sets in the child bags for each subproblem. The overall runtime is nevertheless still $2^{O(k)} poly(n)$.

4 Planar Graphs

A planar graph is a graph that can be drawn such that no edges cross each other. For example, K_4 is planar but K_5 is not. Eppstein [Epp99] showed that if G is planar, then the treewidth of G is bounded by the diameter of G.

Theorem 13.13. *If a graph G is planar, then treewidth$(G) \leq O($diameter$(G))$.*

Baker's technique [Bak94] gives a polynomial time approximation scheme for MaxIS on planar graphs.

Definition 13.14 (Polynomial time approximation scheme)**.** A polynomial time approximation scheme (PTAS) is an algorithm that takes a problem instance and any constant $\varepsilon > 0$, and computes a $(1 \pm \varepsilon)$-approximation to the problem in polynomial time.

4.1 Baker's technique

Baker's technique starts by computing a breadth-first-search tree on G from an arbitrary vertex s. In this tree, the vertices can be arranged in 'levels' based on their distance (in number of edges) from s.

Observation 13.15. There are no edges between vertices in levels that are more than 2 apart - otherwise the edge would have been explored, and the vertices would be in adjacent levels.

Given ε, fix $L := 1/\varepsilon$ (assume L is an integer). Then pick a random offset r from 0 to $L - 1$. uniformly at random, and delete all the vertices in that r-th level and every $L - th$ level after (that is, delete vertices from level r, $r + L$, $r + 2L$, etc.). We will not choose any vertices from these deleted levels in our final independent set.

How many vertices were deleted? Each vertex is deleted with probability $1/L$, since a layer is deleted with $1/L$ probability. Thus the expected number of deleted vertices is

$$\mathbf{E}\left[\text{number of deleted vertices}\right] = \frac{n}{L} = \varepsilon n \leq 4\varepsilon \cdot \text{OPT}. \tag{13.1}$$

where OPT is the size of the optimal maximum independent set.

The last inequality in (13.1) follows from the four color theorem, which states that any planar graph is four-colorable. In any four-coloring of a planar graph, the vertices of a single color form an independent set, since no two adjacent vertices can have the same color. The color with the most vertices has at least $n/4$ vertices, so OPT is at least $n/4$.

After deleting some levels, we are left with 'blocks' of contiguous levels of vertices, as shown in Figure 13.5. We now make our key claim.

Claim 13.16. *Every block B has treewidth $O(1/\varepsilon)$.*

Proof. Start with the planar graph G and the BFS tree from s (Figure 13.5). Delete all levels below B, leaving us with B and the vertices above it $V_{\leq B}$.

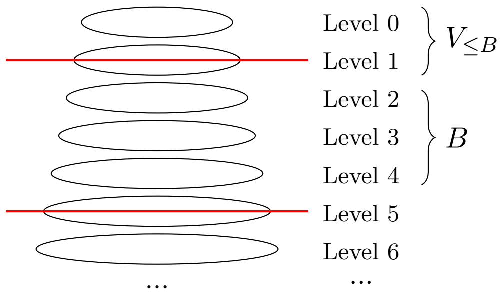


Figure 13.5: BFS tree after deleting levels with offset $r = 1$, $L = 4$ in Baker's technique. The start vertex s is in Level 0. B is an example of a block after this deletion. Level 0 is also a block. $V_{\leq B}$ are the vertices above B in the tree.

Since planar graphs are minor-closed[1], the resulting graph is still planar.

Contract $V_{\leq B}$ until we end up with a single vertex. The resulting graph is still planar, again since planar graphs are minor-closed.

This graph is still connected. Between any two vertices in B, one possible path is to go up to the contracted vertex, and then back down to the other vertex. Since the height of the block is $O(L)$, the length of this path is $O(L) = O(1/\varepsilon)$. So this graph has diameter less than $O(1/\varepsilon)$.

Using Theorem 13.13, this graph has treewidth $O(1/\varepsilon)$.

Now we delete the contracted vertex of $V_{\leq B}$. Since treewidth-$O(1/\varepsilon)$ graphs are minor-closed, we still have a treewidth-$O(1/\varepsilon)$ graph. □

Since every block has treewidth $O(1/\varepsilon)$, we can use our exact algorithm from earlier for bounded-treewidth graphs to solve MaxIS exactly for each block, each taking $2^{O(1/\varepsilon)}\text{poly}(n)$ time (and taking the same overall, since the number of blocks is $O(n)$). We then just return the union of these independent sets, since there can be no adjacent vertices among the independent sets, from our earlier observation.

To show that the independent set we return is a $(1-\varepsilon)$-approximation, first note that given any maximum independent set in G, we delete $\varepsilon \cdot \text{OPT}$ vertices from the set in expectation, since each vertex is deleted with $\varepsilon = 1/L$ probability.

Also, since we obtained the maximum independent set for each block B, this must be at least the number of vertices in the maximum independent set over G that lie in each B. This gives us

$$\begin{aligned}\sum_B \text{MaxIS}\,(G(B)) &\geq \sum_B |\,\text{OPT} \cap B| \\ &= \text{OPT} \,-\varepsilon \cdot \text{OPT} \qquad \text{(in expectation)} \\ &= (1-\varepsilon)\,\text{OPT}\end{aligned}$$

Hence the algorithm returns a $(1-\varepsilon)$-approximation for MaxIS in time polynomial in n.

References

[Bak94] B.S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. Assoc. Comput Mach.*, 41(1):153–180, 1994. 4

[1]The proof of this and other properties we use in this proof are given in the lecture's blogpost.

[Epp99] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms*, 3(3):1–27, 1999. 4

[Has96] J. Hastad. Clique is hard to approximate within $n^{1-\varepsilon}$. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96, pages 627–636, Washington, DC, USA, 1996. IEEE Computer Society. 2.1