

Chapter 8

Vertex Cover and Friends

Lots of subsections missing...

8.1 A special case of Independent Set

In this section, we show that Independent Set has a PTAS in a special type of graph called *hexagonal grid graph*. (We will later see why these are useful.)

Definition 4 *The (infinite) hexagonal grid consists of vertices at all 2-dimensional points with integer coordinates, and two vertices are adjacent if the corresponding points have distance 1, or if they have distance $\sqrt{2}$ along a line of slope 1.*

A hexagonal grid graph is any finite subgraph of the hexagonal grid. It is said to have width w and height h if all its vertices correspond to points with x -coordinates in $\{1, \dots, w\}$ and y -coordinate in $\{1, \dots, h\}$.

8.1.1 Grid graphs with small height

The first intermediate goal is to show that independent set is polynomial in any hexagonal grid graph of height h (presuming that h is a constant - the run-time will be exponential in h .)

The algorithm for $h = 1$

It is very easy to prove this for $h = 1$. In a hexagonal grid graph of height 1, all vertices are on one row of the grid. In particular therefore, the graph consists of the union of paths. A simple greedy algorithm will then find a maximum independent set.

An exact (slow) algorithm for independent set

Before explaining the algorithm for $h = 2$, let us first consider an algorithm for independent set that works in any graph, but that is exponential unless the graph has a special structure.

Consider one vertex v of the graph. Either there exists a maximum independent set I that contains v , or there doesn't. If no maximum independent set contains v , then we can find a maximum independent set by deleting v and recursing in the remaining graph. If some maximum independent set contains v , then we can find it by deleting v and all its neighbours (we denote them by $N(v)$), computing the maximum independent set in the remaining graph, and adding v to it.

In other words, the following algorithm works for independent set:

```

if  $G$  has no vertices, return  $\emptyset$ 
let  $v$  be a vertex of  $G$ .
compute a maximum independent set  $I_1$  in  $G - v$ 
compute a maximum independent set  $I_2$  in  $G - v - N(v)$  and add  $v$  to it
return the larger of  $I_1$  and  $I_2$ 

```

The problem with this algorithm is that in each recursion, we examine two different subgraphs recursively. Therefore, the run-time of the algorithm is potentially $O(2^n)$ (and in fact, it reaches this on some graphs.) But for hexagonal grid graphs, not all these subgraphs are different if we choose v carefully (and eliminate multiple vertices at once, not just v), which therefore leads to a polynomial algorithm if h is a constant.

The case $h = 2$

So now assume we have a hexagonal grid graph of width w and height 2. We denote the vertices by (i, j) , with $i \in \{1, \dots, w\}$ and $j \in \{1, 2\}$. Vertices $a := (w, 1)$ and $b := (w, 2)$ will also be called the *bottom right corner* and *top right corner*.

We assume in the following assume that G is connected (if it is not, then we can find the maximum independent set by finding the maximum independent set in each component.) Now apply the idea of the above algorithm, except eliminate both top right and bottom right corner at once. The algorithm is hence as follows:

```

if  $G$  has width 0, return  $\emptyset$ 
let  $a$  and  $b$  be the the vertices  $(w, 1)$  and  $(w, 2)$ .
// Note that  $a$  and  $b$  don't necessarily exist in  $G$ 
let  $I_1$  be the maximum independent set in  $G - \{a, b\}$ .
if  $a$  doesn't exist, let  $I_2 = \emptyset$ 
else let  $I_2$  be the maximum independent set in  $G - \{a, b\} - N(a)$  and add  $a$  to it.
if  $b$  doesn't exist, let  $I_3 = \emptyset$ 
else let  $I_3$  be the maximum independent set in  $G - \{a, b\} - N(a)$  and add  $a$  to it.
if  $a$  or  $b$  doesn't exist, or if  $(a, b)$  is an edge of  $G$  let  $I_4 = \emptyset$ 
else let  $I_4$  be the maximum independent set in  $G - \{a, b\} - N(a) - N(b)$ 
    and add  $a$  and  $b$  to it.
return the largest of  $I_1, I_2, I_3, I_4$ 

```

The crucial idea is that because we choose a and b to be the right corners always, and because their neighbours are all only in the same or the previous column, there are not that

many different subgraphs that can result. In other words, many of the subproblems overlap, and we can hence use dynamic programming to solve the problem. To explain this precisely, we need a bit more notation.

Definition 5 Let \mathcal{G}_i be all graphs H that satisfy

- H contains all vertices of G that have x -coordinate $\leq i$.
- H contains some of the vertices of G that have x -coordinate $i + 1$.
- H contains no other vertices.
- H contains all edges between the above vertices that were edges in G .

Since G had at most 2 vertices with x -coordinate i , there are at most 4 different graphs in \mathcal{G}_i .

Now define $\alpha(i, H)$, for any graph $H \in \mathcal{G}_i$, to be the size of the maximum independent set in H . (Like in most dynamic programming algorithms, we will only compute the size of the optimum set and not the actual set, but one can retrieve the optimum set by storing which case was used to achieve the optimum size.)

We compute $\alpha(i, H)$ recursively using the value of $\alpha(i - 1, H')$. In the base case, $i = 0$. So H consists of some subset of the vertices with x -coordinate 1. This is a hexagonal grid graph of width 1 and we know how to find an independent set in it. (Or an even easier base case would be to use $i = -1$, in case of which H is empty and so any independent set has size 0.)

Now consider the case $i > 1$. Apply the above algorithm for computing the maximum independent set in H . The crucial observation is that all neighbours of a and b have x -coordinate $i - 1$. In particular, the graph H' that we obtain by deleting a and b (and some of their neighbours) belong to \mathcal{G}_{i-1} . So we have already computed the size of the maximum independent set in H' and just need to look it up and add the appropriate number to it. Here is the full code for computing $\alpha(i, H)$:

```

if  $i = -1$ , set  $\alpha(i, H) = 0$  and return
let  $a$  and  $b$  be the the vertices at  $(i + 1, 1)$  and  $(i + 1, 2)$ .
// Note that  $a$  and  $b$  don't necessarily exist in  $H$ 
let  $H_1 = G - \{a, b\}$  and  $\alpha_1 = \alpha(i - 1, H_1)$ 
if  $a$  doesn't exist, let  $\alpha_2 = 0$ 
else let  $H_2 = G - \{a, b\} - N(a)$  and  $\alpha_2 = \alpha(i - 1, H_2) + 1$ 
if  $b$  doesn't exist, let  $\alpha_3 = 0$ 
else  $H_3 = G - \{a, b\} - N(a)$  and  $\alpha_3 = \alpha(i - 1, H_3) + 1$ 
if  $a$  or  $b$  doesn't exist, or if  $(a, b)$  is an edge of  $G$  let  $\alpha_4 = 0$ 
else let  $H_4 = G - \{a, b\} - N(a) - N(b)$  and  $\alpha_4 = \alpha(i - 1, H_4) + 2$ 
set  $\alpha(i, H) = \max\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ 

```

Note that any $H \in \mathcal{G}_i$ can be characterized with 3 parameters: i and two boolean values to indicate whether $(i+1, 1)$ and $(i+1, 2)$ belong to H or not. In particular, we can therefore store $\alpha(i, H)$ in a $w \times 2$ -array and look it up in constant time. Therefore, the algorithm takes constant time per subgraph H . Since for each i there are at most 4 subgraphs to consider, and we need to do it for $i = -1, \dots, w$, the run-time is $O(4w)$. Since G is connected we have $w \leq n$ and hence the run-time is $O(n)$. At the end, we have compute $\alpha(w, H)$ for all $H \in \mathcal{G}_w$. But there is only one graph in \mathcal{G}_w , namely G itself (there are no vertices in the $(w+1)$ st column), and therefore we have computed $\alpha(w, G)$, the size of the maximum independent set in G . (And as hinted at before, we can retrieve the actual maximum independent set by retracing which recursion was used to obtained the value for $\alpha(w, G)$.)

Theorem 2 *We can find a maximum independent set in a hexagonal grid graph of height 2 in $O(n)$ time.*

8.1.2 Constant h

Now we do the algorithm again for larger (but constant) h . Really almost nothing changes, except that we replace the two vertices $(i, 1)$ and $(i, 2)$ by the h vertices $(i, 1), \dots, (i, h)$ everywhere.

Thus, define \mathcal{G}_i to be all those graphs that contain all vertices in the first i columns of G , and some of the vertices in the $(i+1)$ st column. For each graph H in \mathcal{G}_i , define $\alpha(i, H)$ to be the size of the maximum independent set in H . This is computed quite exactly as before:

if $i = -1$, then $\alpha(i, H) = 0$

let W_{i+1} be the the vertices of H with x -coordinate $i+1$.

for any subset $S \subseteq W_{i+1}$

if there exists an edge between two vertices in S

set $\alpha_S = 0$

else

set $H_S = G - S - \bigcup_{v \in S} N(v)$ and $\alpha_S = \alpha(i-1, H_S)$

set $\alpha(i, H)$ to be the maximum over all α_S

Note that there are up to 2^h possible graphs in \mathcal{G}_i , and for each of them, we explore 2^h subsets S . Hence for each index i , we spend $O((2^h)^2)$ time to compute all values $\alpha(i, \cdot)$. In total the run-time is hence $O(4^h w) = O(4^h n)$. Note that this is polynomial (even linear) as long as h is a constant.

Theorem 3 *for any hexagonal grid graph of height h , we can find the maximum independent set in $O(4^h n)$ time.*

Two remarks are in order. First, the run-time can actually be improved to $O(2^h n)$ time if we do the bookkeeping a bit more carefully (no details will be given.) Secondly, this algorithm proves that the problem is what's known as *fixed-parameter tractable* in h , a concept that we will encounter soon.

8.1.3 Arbitrary heights

Now let us consider a hexagonal grid graph of arbitrary height. Finding a maximum independent set then is not polynomial (in fact, this is NP-hard even for hexagonal grid graphs, though we won't show this.) But by splitting the graph into hexagonal grid graphs of some constant height k , we can create a PTAS for maximum independent set.

So presume that we are given an $\varepsilon > 0$, and we would like to create an $(1 - \varepsilon)$ -approximation algorithm for maximum independent set. Define $k = 1/\varepsilon$.

Let V_0 be all those vertices whose y -coordinate is a multiple of k , and let G_0 be the graph obtained from G by deleting all vertices in V_0 . Observe that G_0 has many connected components, and each of those connected components is a hexagonal grid graph of height $k - 1$ (since we deleted every k th row of G .) So we can find the maximum independent set I_0 of G_0 in time $O(4^{k-1}n)$.

Similarly for $i = 1, \dots, k - 1$, define V_i to be all those vertices whose y -coordinate is $i \pmod{k}$, i.e., the y -coordinate is in $\{i, i + k, i + 2k, \dots\}$. Let G_i be the graph obtained from G by deleting all vertices in V_i . Each connected component of G_i has height $k - 1$, and so we can find the maximum independent set I_i of G_i in $O(4^{k-1}n)$ time.

Finally let I be the largest among sets I_0, \dots, I_{k-1} . Clearly this is an independent set of G (each I_i is an independent set in an induced subgraph of G , so it's an independent in G .) Return this set as an approximation of the maximum independent set in G .

Before proving why this is a good independent set, let's analyze the run-time. Finding I_i for one i takes time $O(4^{k-1}n)$ time. We do this for k different values of i , so this takes $O(k4^{k-1}n)$ time, which is polynomial in n as long as ε (and hence k) is a constant.

Lemma 8.1 *The independent set I computed with the algorithm satisfies $|I| \geq (1 - \varepsilon)|I^*|$, where I^* is a maximum independent set of G .*

Proof: For $i = 0, \dots, k - 1$, let $I_i^* = I^* - V_i$. Clearly I_i^* is an independent set of G_i , and since I_i was the maximum independent set of G_i , we have $|I_i^*| \leq |I_i|$.

Each vertex in I^* belongs to exactly one of the sets V_1, \dots, V_k . In consequence, each vertex in I^* belongs to exactly $k - 1$ of the k sets I_0^*, \dots, I_{k-1}^* . Or in other words, $|I_0^*| + \dots + |I_{k-1}^*| = (k - 1)|I^*|$.

Now we use the old trick about relating maxima and averages and get the result. Namely:

$$\begin{aligned}
 |I| &= \max\{|I_0|, \dots, |I_{k-1}|\} \\
 &\geq \frac{1}{k}(|I_0| + \dots + |I_{k-1}|) \\
 &\geq \frac{1}{k}(|I_0^*| + \dots + |I_{k-1}^*|) \\
 &= \frac{1}{k}(k - 1)|I^*| \\
 &= \frac{k - 1}{k}|I^*|
 \end{aligned}$$

$$\begin{aligned}
&= \left(1 - \frac{1}{k}\right)|I^*| \\
&\geq (1 - \varepsilon)|I^*| \text{ since } k = \lceil 1/\varepsilon \rceil \geq 1/\varepsilon.
\end{aligned}$$

□

So in summary, we have obtained an algorithm with run-time $O\left(\frac{1}{\varepsilon}4^{1/\varepsilon}n\right)$ to find a $(1 - \varepsilon)$ approximation of the maximum independent set in a hexagonal grid graph.

Theorem 4 *There exists a PTAS for maximum independent set in hexagonal grid graphs.*

Who cares?

A fair question to ask is: But who cares about hexagonal grid graphs? It is not exactly likely that such a graph will happen in practice, is it?

There are two reasons why hexagonal grid graphs are worth studying. Recall that a planar graph is a graph that can be drawn in the plane without crossing. Not every planar graph is a hexagonal grid graph, but one can show that after subdividing edges in a planar graph sufficiently often, it becomes a hexagonal grid graph. (No details are given.) Now as long as every edge is subdivided an even number of times, this affects the size of a maximum independent set in a deterministic way: for every two subdivisions, the maximum independent set size goes up by one. Therefore, there is a strong relationship between independent sets in planar graphs and independent sets in hexagonal grid graphs, which suggests that there might be a PTAS for independent set in all planar graphs.

In fact, there is such a PTAS for all planar graphs (and the algorithm in this section was a much-simplified version of the approach), given first by Baker in 1984. In a planar graph, we can define the outermost layer to be all vertices that are adjacent to the infinite face. We can define the next layer to be all vertices that appear on the infinite face after deleting the first layer, and so on. In this way, we can impose a layer-structure onto any planar graph, with the property that any vertex on layer i has neighbours only in layer $i - 1$ and $i + 1$. This somewhat resembles a grid-structure. So it may not be a surprise that if a planar graph has only h layers (for some constant h), we can find the maximum independent set in $O(8^h n)$ time.¹

Now we can do the same trick for planar graphs as we did for grid graphs: Delete every k th layer and compute the maximum independent set in the resulting graph. Do this k times (for the k possible shifts of layers) and return the maximum of the computed independent sets. This independent set then is a $(1 - \frac{1}{k})$ -approximation of the maximum independent set, and hence gives a PTAS for independent set in planar graphs.

Theorem 5 *Maximum independent set admits a PTAS in planar graph.*

¹The details of this are horribly complicated in Baker's paper, but were later much simplified noticing that a graph with h layers—also called an h -outerplanar graph—has treewidth at most $3h$, and in any graph of constant treewidth, we can find a maximum independent set in linear time. But this is *really* leading us too far from cs466 topics. The graduate course cs762 (graph-theoretic algorithms) covers a lot of this.