TTIC 31290: Machine Learning for Algorithm Design (Fall 2025)

Avrim Blum and Dravyansh Sharma

Lecture 2: 10/02/25 Lecturer: Avrim Blum

Outline for today

- Fixed Parameter Tractability
- Case study: Vertex Cover
- Average-case Analysis
- Case study: Graph coloring

Today we are continuing our discussion of classic theoretical approaches for dealing with NP-hardness, focusing on Fixed parameter tractability and average-case analysis.

1 Fixed Parameter Tractability

Last time we looked at approximation algorithms. But what if you really want the optimal solution for some NP-hard problem. We know we can't expect to optimally solve arbitrary instances in time polynomial in the input size, but maybe we can identify a relevant parameter "k" such that we can achieve running time $f(k) \cdot poly(n)$ for some function f. If we can do that, we say the problem is *Fixed-parameter tractable*, in that parameter k.

One nice problem where you can do this is vertex cover, where k is the size of the optimal cover.

Theorem 1. *There is an algorithm for finding a minimum vertex cover of an n-node graph that runs in time $O(2^k \cdot poly(n))$ where k is the size of the minimum vertex cover.*

Algorithm VC-FPT

Input: graph G, value k (we'll try this with $k = 0, 1, 2, 3, ...$ until we solve it)

```
0. If $k = 0$ then if $G$ has no edges return $\{\}$, else return "fail".
1. Pick an arbitrary edge $(u, v)$.
2. Let $G_u$ be the graph $G$ with vertex $u$ and its incident edges removed.
   (a) Call VC-FPT($G_u, k-1$).
   (b) If this returns a cover $S$ of size $k-1$, return $S \cup \{u\}$.
3. Let $G_v$ be the graph $G$ with vertex $v$ and its incident edges removed.
   (a) Call VC-FPT($G_v, k-1$).
```

    (b) If this returns a cover S of size $k-1$, return $S \cup \{v\}$.

4. If we reach here, then return "fail".

Proof. (Theorem 1) There are two things to analyze: running time and correctness. The easiest is running time: given input value k, we make two recursive calls with value $k-1$, halting at $k=0$. So we immediately get the $O(2^k \cdot poly(n))$. Now, what about correctness? For this, let's argue by induction. The base case $k=0$ is trivial. For the general case, consider some Vertex Cover C of size k. We know C must contain at least one of u or v. If C contains u, then $C \setminus \{u\}$ is a Vertex Cover of G_u of size $k-1$, so Step 2(a) will return a cover of G_u of size $k-1$ by induction, and so $S \cup \{u\}$ is a cover of G of size k. If C doesn't contain u then it must contain v, in which case $C \setminus \{v\}$ is a Vertex Cover of G_v of size $k-1$, so by induction Step 3 will return an optimal vertex cover. □

2 Average-case analysis

If we can't solve our problem in polynomial time in the worst case, then another classic approach is to posit some simple probability distribution over instances, and ask whether we can get an algorithm that works well with high probability (or in expectation) over that distribution. This is called "average case analysis". For example, if we have a graph problem, and we use the $\mathcal{G}(n, 1/2)$ random graph model, which says that each edge is in the graph independently with probability $1/2$, then this is really doing an average case out of all graphs on n vertices.

An interesting case-study to look at for average-case analysis is Graph k-coloring. Given a graph G and an integer k, the goal is to assign each vertex of G to one of k colors so that no two adjacent vertices get the same color (or output "not possible" if it's not possible). This problem is easy to solve for $k=2$ (asking "is the graph bipartite") and NP-hard for any $k \geq 3$. Famously, any *planar* graph is 4-colorable (and there are efficient algorithms for 4-coloring planar graphs).

Unfortunately, the best approximation factors known for this problem are quite high. E.g., currently the best approximation known for coloring 3-colorable graphs is $n^{0.197}$ colors, and it gets worse for $k > 3$. On the hardness side, it's known to be NP-hard to achieve a $2k-1$ coloring. But what about *random* k-colorable graphs?

2.1 Random k-colorable graph model $\mathcal{G}(n, p, k)$

Let k be a given constant (like 3) and let $p \in [0, 1]$. Here is one reasonable model for a random k-colorable n-vertex graph:

1. Assign each vertex randomly (iid) to one of k color classes.
2. For each pair of vertices (u, v) from different color classes, add the edge (u, v) into the graph with probability p.

It turns out we can easily k-color graphs from $\mathcal{G}(n, p, k)$ with high probability, and in fact we can recover the hidden coloring (think of it as a hidden "clustering" where edges represent noticeable significant differences) for any constant value of p (and even very sub-constant).

Simple k-coloring algorithm

1. For each pair of vertices u, v, calculate the number n_{uv} of neighbors that they have in common.
2. Sort the numbers n_{uv}. Create a brand-new "similarity graph" by connecting those pairs (u, v) whose values n_{uv} is highest until there are k components to this graph. Output them as the k color classes.

Analysis: If u and v are in the same color class, then the expected number of neighbors they have in common is $(n-2)(\frac{k-1}{k})p^2$. That's because there are $n-2$ other nodes, each of which has a $(k-1)/k$ chance of being from a different color, and then conditioned on it being a different color, there is a p^2 chance it is a neighbor of both u and v. On the other hand, if u and v are in different color classes, then the expected number of neighbors they have in common is $(n-2)(\frac{k-2}{k})p^2$. Notice that "$k-1$" has changed to "$k-2$" because the node must have color different from both u and v. So, the expected number of neighbors in common differs by a factor of $\frac{k-1}{k-2}$. E.g., for 3-colorable graphs, the expected number of neighbors in common differs by a factor of 2.

Now, to finish, we just need to apply a tail inequality to say that so long as the expectations are large enough, with high probability every pair of nodes will share a number of neighbors in common that's close to their expectation. This will then allow us to partition all the nodes by color.

To do this, there are several tail inequalities we can use. Here are some that I generally find most convenient, called Chernoff bounds and Hoeffding bounds.

Chernoff and Hoeffding bounds

Let $X_1, \ldots, X_n$ be a sequence of n independent $\{0,1\}$ random variables with $\Pr[X_i = 1] = p_i$ not necessarily the same. Let S be the sum of the RVs, and let $\mu = \mathbf{E}[S]$. Then, for $0 \leq \delta \leq 1$, the following inequalities hold.
Additive (Hoeffding) bounds:

- $\Pr[S \geq \mu + \delta n] \leq e^{-2n\delta^2}$.
- $\Pr[S \leq \mu - \delta n] \leq e^{-2n\delta^2}$.

Multiplicative (Chernoff) bounds:

- $\Pr[S > (1+\delta)\mu] \leq e^{-\delta^2\mu/3}$,
- $\Pr[S < (1-\delta)\mu] \leq e^{-\delta^2\mu/2}$.

Also, for any $k > 1$, we get:

- $\Pr[S > k\mu] < \left(\frac{e^{k-1}}{k^k}\right)^{\mu}$.

To finish the argument, we can use the multiplicative (Chernoff) bounds with $\delta = \frac{1}{2k}$ so that $(k-2)(1+\delta) < (k-1)(1-\delta)$, giving us a separation. We just need the failure probability to be small enough so that with high probability, *no* pairs deviate by more than their allowed $1 \pm \delta$. This just requires that the expectation μ be asymptotically larger than $k^2 \log(n)$. So, p just has to be asymptotically larger than $k \cdot \sqrt{\log(n)/n}$. $\square$

3 Additional Resources

Fixed-Parameter Tractability:

- Jason Li's lecture notes at CMU: `https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15850-f18/www/scribes/lecture12.pdf`
- David Karger's lecture notes at MIT:

  `https://ocw.mit.edu/courses/6-854j-advanced-algorithms-fall-2005/d8a8d8a91186a724e9d19875657288b8_lecture14.pdf`

Average-case analysis:

- Wojciech Szpankowski's book on average-case analysis on sequences: `https://www.cs.purdue.edu/homes/spa/courses/pg17/mybook.pdf`