Lecture 13 (18.01.2013) Author: Tomasz Kociumaka

Exponential Time Hypothesis

1 Motivation

We have learned how to solve Feedback Vertex Set in $\mathcal{O}^\star(1.8899^n)$ time (see lecture 1) and Maximum Independent Set in $\mathcal{O}^\star(3^{n/3}) = \mathcal{O}^\star(1.4423^n)$ (lecture 4). On the other hand, we know that k-Clique cannot be solved in $\mathcal{O}^\star\left(f(k)n^{\mathcal{O}(1)}\right)$ time unless $FPT = W[1]$ (lecture 11). Today we are going to ask questions like the following ones.

- Can we hope for $2^{o(n)}$-time algorithms for Feedback Vertex Set or Maximum Independent Set?
- Can we hope for an $f(k)n^{o(k)}$-time algorithm for k-clique?

Note that until now, we were not able to argue (under widely believed assumptions, like $P \neq NP$ or $FPT \neq W[1]$) that e.g. there are no $2^{\mathcal{O}(\log^2 n)}$ or even $f(k)n^{\mathcal{O}(\log^* k)}$ algorithms respectively.

Observe that the questions we have stated disregard the (multiplicative) constants in the exponent of the time complexity. There are tools that allow to prove the optimality of such constants, but they require very strong assumptions (which are not so unlikely to fail) and the theory around them is less developed (see lecture 14).

In this lecture we are going to deal with subexponential algorithms, but before we begin, note that one needs to be very careful how the instance size is measured. For example the Maximum Clique problem can be solved in $2^{\mathcal{O}(\sqrt{m})}$ time (see exercise 110), which is subexponential in terms of the input size, while later today we shall see that an $2^{o(n)}$-time algorithm, i.e. an algorithm subexponential in terms of the number of vertices, is unlikely to exists.

2 Exponential Time Hypothesis

The following conjecture is the central assumption made to deduce further results. It comes in two versions, which are *not* known to be equivalent, but both are widely believed to be true.

Conjecture 1 (Exponential Time Hypothesis [1]).

(i) There is no $2^{o(n)}$-time algorithm for 3-CNF-SAT*, where n is the number of variables.*

(ii) There exists $\varepsilon > 0$ such that 3-CNF-SAT *cannot be solved in $\mathcal{O}(2^{\varepsilon n})$ time.*

Observe, that (i) is a consequence of (ii). In order to understand the difference, let us define $\mathcal{A}$ to be the family of all algorithms solving the 3-CNF-SAT problem. For each $A \in \mathcal{A}$ let

$$s(A) = \inf\{\varepsilon : A \text{ runs in } \mathcal{O}(2^{\varepsilon n}) \text{ time}\}.$$

Moreover, let $S_3 = \{s(A) : A \in \mathcal{A}\}$. Version (i) states that $0 \notin S_3$, while version (ii) that $s_3 = \inf S_3 > 0$. Hypothetically, there could exist an infinite sequence of algorithms (i.e. a *non-uniform* algorithm) such that the for every $\varepsilon > 0$ the algorithms far enough in the sequence would run in $2^{\varepsilon n}$ time, but none of them would run in $2^{o(n)}$ time.

One of the most significant consequence of ETH is the following theorem.

Theorem 2. *Under ETH version (ii), there exists $\varepsilon' > 0$ such that* 3-CNF-SAT *cannot be solved in $\mathcal{O}(2^{\varepsilon' m})$ time, where m is the number of clauses. In particular, no $2^{o(m)}$ algorithm exists.*

We shall prove this theorem later this lecture. Note that it allows to measure the instance size of 3-CNF-SAT using number of clauses (or the input size), while ETH relies only on hardness in terms of the number of variables. This makes designing reductions much easier, in particular the classic reductions can be used to prove hardness of many problems.

Corollary 3. *Under ETH, there is no $2^{o(m)}$-time algorithm for* MAXIMUM INDEPENDENT SET.

Proof. Recall the standard reduction from 3-CNF-SAT to MAXIMUM INDEPENDENT SET. For each variable x a pair of adjacent vertices corresponding to x and $\neg x$ is made. For each clause a triangle is created and the vertex corresponding to a literal ℓ is made adjacent to the variable-vertex corresponding to $\neg\ell$. This way we introduce $3m+2n = \mathcal{O}(n+m)$ vertices and $6m+n = \mathcal{O}(m+n)$ edges. We claim that the formula can be satisfied iff the graph has an independent set of size $m+n$. As it can be partitioned into $m+n$ cliques (K_3 for clauses and K_2 for variables) clearly no larger independent set may exists and any independent set of size $m+n$ must hit each of this cliques. In particular for each variable x either the vertex corresponding to x or to $\neg x$ is chosen, which gives a interpretation of variables with the true value corresponding to the literal in the independent set. As each clause contains a member of the independent set, and the corresponding literal must be the true value of the variable (as the vertex is adjacent to the negation of that literal), each clause is satisfied, which makes the whole formula satisfied.

Similarly, if the formula can be satisfied, then one can create an independent set by adding a vertex corresponding to the true value of each variable in the variable-K_2 and any vertex corresponding to a satisfied literal in each clause-K_3. It is easy so see that no two adjacent vertices are chosen this way. □

Consequently, no $2^{o(n)}$-time algorithms exist for MAXIMUM INDEPENDENT SET and for MAXIMUM CLIQUE. (Recall that $2^{o(m)}$-time algorithm for MAXIMUM CLIQUE exists).

Before we proceed with the proof of Theorem 2, let us introduce the Sparsification Lemma and concepts required to formulate it.

d-HITTING SET
Input: A family $\mathcal{F} \subseteq 2^U$ ($|U| = n$) of sets of cardinality at most d, and an integer k
Question: Does there exist $X \subseteq U$, such that $|X| \le k$ and X intersects with each $F \in \mathcal{F}$?

Definition 4. We say that $\mathcal{F}'$ is a *restriction* of $\mathcal{F}$ if

(a) for each $X \in \mathcal{F}'$ there exists $Y \in \mathcal{F}$ such that $X \subseteq Y$,

(b) for each $Y \in \mathcal{F}$ there exists $X \in \mathcal{F}'$ such that $X \subseteq Y$.

Observe, that if $(\mathcal{F}, k)$ is a valid instance of d-HITTING SET and $\mathcal{F}'$ is a restriction of $\mathcal{F}$, then $(\mathcal{F}', k)$ is also a valid instance of d-HITTING SET. Moreover, if the latter is a YES-instance, so is the former. It is also easy to see, that being a restriction is a transitive relation.

Theorem 5 (Sparsification Lemma). *For each $d \in \mathbb{N}$ and $\varepsilon > 0$ there exists a constant C (dependent on d and ε only) such that in $\mathcal{O}^\star(2^{\varepsilon n})$ time a given instance $(\mathcal{F}, k)$ of d-*HITTING SET *can be reduced to instances $(\mathcal{F}_1, k), \ldots, (\mathcal{F}_\ell, k)$ of d-*HITTING SET*, such that $\ell = \mathcal{O}(2^{\varepsilon n})$, for each $i \in \{1, \ldots, k\}$, $|\mathcal{F}_i| \le Cn$ and the family $\mathcal{F}_i$ is a restriction of $\mathcal{F}$, and the input instance is a YES-instance iff at least one of the output instances is a YES-instance.*

Remark 6. The proof of the Sparsification Lemma implies, that additionally there exists a constant $C'(d, \varepsilon)$ such that for each i and $x \in U$, at most C' sets from $\mathcal{F}_i$ contain x.

We shall give the technical proof of the Sparsification Lemma later on. First, let us see how Theorem 2 can be deduced.

Let us begin with a (linearly-bounded) reduction of 3-CNF-SAT to 3-Hitting Set. Let

$$\begin{aligned} U &= \{x_i, \neg x_i : i \in \{1, \ldots, n\}\}, \\ \mathcal{F} &= \{\{x_i, \neg x_i\} : i \in \{1, \ldots, n\}\} \cup \{\{l_1, l_2, l_3\} : l_1 \vee l_2 \vee l_3 \in \phi\}, \\ k &= n. \end{aligned}$$

Clearly this is an equivalent instance, the hitting set must contain either x_i or $\neg x_i$, which can be interpreted as the true value of x_i.

Now, the Sparsification Lemma can be used for the instance of 3-Hitting Set we have just produced. We obtain a bunch of smaller instances of the 3-Hitting Set, but since the universe is preserved and the families $\mathcal{F}_i$ are restrictions of $\mathcal{F}$, these instances can be interpreted as 3-CNF-SAT instances with $\leq Cn$ clauses only.

Now, assume that for any ε' we can solve 3-CNF-SAT in $2^{\varepsilon' m}$ time. Let us take an instance of 3-CNF-SAT and go through the reduction using the Sparsification Lemma for $d = 3$ and ε to be fixed later. We have used $\mathcal{O}^\star(2^{\varepsilon n})$ time and now it suffices to solve $\leq 2^{\varepsilon n}$ instances with $m \leq C(3, \varepsilon)n$. By the assumption, this can be achieved in $2^{\varepsilon n} \cdot 2^{\varepsilon' C(3,\varepsilon) n} = 2^{(\varepsilon + \varepsilon' C(3,\varepsilon))n}$ time. Consequently, the 3-CNF-SAT can be solved in $\mathcal{O}^\star\left(2^{(\varepsilon+\varepsilon' C(3,\varepsilon))n}\right)$ time for any $\varepsilon, \varepsilon' > 0$. Fixing $\varepsilon < \frac{\delta}{2}$ and then $\varepsilon' < \frac{\delta}{2C(3,\varepsilon)}$ one gets an algorithm working in $\mathcal{O}^\star\left(2^{\delta n}\right)$ time for any $\delta > 0$, which contradicts the ETH version (ii).

3 Proof of Sparsification Lemma

From now on, we shall concentrate on the proof of the Sparsification Lemma. The proof is technical, but none of its ingredients is hard to understand. Neverheless, it seems rather unclear, why such a result has been obtained for the Hitting Set problem, and the analogues for different problems are not known.

3.1 Preliminaries

Definition 7. Let $S_1, \ldots, S_c$ be a collection of sets of size j. We call it a *j-sunflower*, if $H = \bigcap_{i=1}^{c} S_i \neq \emptyset$. Set H is called the *heart* of the sunflower and sets $S_i \setminus H$ are referred to as *petals*.

For $\mathcal{F} \subseteq 2^U$, we denote the family of inclusion-wise minimal elements of $\mathcal{F}$ by $\pi(\mathcal{F})$. Observe, that $X \subseteq U$ is a hitting set of $\mathcal{F}$ iff it is a hitting set of $\pi(\mathcal{F})$, since $\pi(\mathcal{F})$ is both a restriction and a subset of $\mathcal{F}$.

Let us define a couple of constants as well. Let α be a positive integer constant, whose value, dependent on d and ε, is going to be fixed later. Let $\beta_i = (4\alpha)^{2^{i-1}-1}$ and $\theta_i = \alpha\beta_i$ for $i \in \{1, \ldots, d\}$ and let $\theta_0 = 2$. Observe that θ is a strictly increasing integer sequence.

3.2 Algorithm

```
1  Procedure Reduce($\mathcal{F}$)
2      $\mathcal{F} := \pi(\mathcal{F})$;
3      for $j := 2$ to $d$ do
4          for $i := 1$ to $j - 1$ do
5              if there exists a $j$-sunflower $S_1, \ldots, S_c$ with petals of size $i$ and $c \geq \theta_i$ then
6                  $H := \bigcap S_i$;
7                  Reduce($\mathcal{F} \cup \{H\}$);
8                  Reduce($\mathcal{F} \cup \{S_h \setminus H : h \in \{1, \ldots, c\}$);
9                  return;
10     output $\mathcal{F}$;
```

Observe that, since $d = \mathcal{O}(1)$, the condition of line 5 can be checked by guessing H (there are at most $|\mathcal{F}|2^d$ choices) and then it suffices to find all sets in $\mathcal{F}$, which are of size j and include H. Thus, except for the recursive call, the procedure works in polynomial time. In other words, the total time complexity is up to polynomial factors, bounded by the size of the recursion tree.

3.3 Proof of correctness

In the following series of lemmas, we are going to prove that the procedure *Reduce* satisfies the conditions required by the Sparsification Lemma.

Lemma 8. *Procedure* Reduce *outputs a collection of families $\mathcal{F}_1, \ldots, \mathcal{F}_\ell$, which are restrictions of $\mathcal{F}$ and $(\mathcal{F}, k)$ is a YES-instance of d-*HITTING SET *iff any of $(\mathcal{F}_1, k), \ldots, (\mathcal{F}_\ell, k)$ is a YES-instance.*

Proof. First, let us prove that *Reduce* terminates. Observe, that when we call *Reduce* in lines 7 and 8, then the restriction to minimal set at line 2 immediately removes sets $S_1, \ldots, S_{\theta_i}$ from the family. Thus, the total size of sets in $\mathcal{F}$ after line 2 is strictly smaller then at the same moment of the parent call.

Now, the structure of recursive calls is proved to be a full binary tree and we can conduct inductive proofs on the depth of the tree. As we have seen, $\mathcal{F}$ and $\pi(\mathcal{F})$ are equivalent, so it suffices to prove that $\mathcal{F}_1 = \mathcal{F} \cup \{H\}$ and $\mathcal{F}_2 = \mathcal{F} \cup \{S_h \setminus H : h \in \{1, \ldots, c\}\}$ are restriction of $\mathcal{F}$ (which is clear) and that $(\mathcal{F}, k)$ is YES-instance iff $(\mathcal{F}_1, k)$ or $(\mathcal{F}_2, k)$ is. Properties of restrictions give the right-to-left implication. For a proof of the left-to-right one, it suffices to see that if X is a hitting set of $\mathcal{F}$, then X hits all sets S_i, so it hits *all* petals $S_i \setminus H$ (and is a hitting set of $\mathcal{F}_1$) or it hits H (and is a hitting set of $\mathcal{F}_2$). □

Lemma 9. *Any output family $\mathcal{F}$ contains at most $(\theta_{j-1} - 1)n$ sets of size j for each $j \in \{1, \ldots, d\}$.*

Proof. For a proof by contradiction assume that there are at least $1 + (\theta_{j-1} - 1)n$ sets of size j in an output family $\mathcal{F}$. Then, one of the elements of the universe would be, by the Pigeonhole Principle, included in at least θ_{j-1} sets of size j. But then, these sets would form a sunflower of size θ_{j-1} with petals of size $i \leq j - 1$. But $\theta_i \leq \theta_{j-1}$, so such a sunflower would be detected in line 5, a contradiction. □

Consequently, the size of the output families is $\mathcal{O}(n)$ as θ_j and d are constants here.

The most involved part of the proof is devoted to limit the number of output families. Before we proceed with formal proofs, let us sketch their structure. Having in mind the recursion tree, we say that the calls in line 7 are *left* edges, and the calls in line 8 – *right* edges. Each output family can be associated with a unique root-to-leaf path in the recursion tree. We shall prove that such path's length is $\mathcal{O}(n)$, and manipulating α, the number of right edges on a single path can be limited to δn for arbitrarily small δ. This way the number output families is bounded by $d \cdot \binom{\mathcal{O}(n)}{\delta n}$. Due to the careful choice of constants, as α grows, this value can ultimately be bounded by $2^{\varepsilon n}$.

Lemma 10. *Assume that $\mathcal{F}$ is a family obtained in the course of the algorithm and that a set of size i has been inserted by one of the ancestor calls. Then any $x \in U$ belongs to $\leq 2\theta_{i-1}$ sets of size i.*

Proof. In an inductive argument lets us assume, that a set of size i has been inserted by the parent call. Let us investigate the progress of this call. Observe, that the outer loop has advanced to $j > i$, so no appropriate i-sunflower has been found. A collection of θ_{i-1} sets of size i containing a fixed $x \in U$ of size θ_{i-1} would give rise to such a sunflower, so at most θ_{i-1} original sets of size i contain x. Now, let us consider the newly-added sets. In case of the *left* edge, this is clear as just $1 \leq \theta_{i-1}$ set has been added. The *right* edge might be problematic, as we add $\theta_i > \theta_{i-1}$ sets. Thus, assume the parent call has detected a j-sunflower with petal size i and that x belongs to at least θ_{i-1} petals. But then, the original sunflower restricted to sets from which these petals originate, is a sunflower that would be detected at the previous step of the inner loop, a contradiction.

Consequently a fixed $x \in U$ belongs to at most θ_{i-1} old and θ_{i-1} new sets of size i, which makes $2\theta_{i-1}$ in total. □

Recall that in line 2 some of the sets are removed from $\mathcal{F}$ as not minimal. We say that X *eliminated* Y if $X \subseteq Y$ and both $X, Y \in \mathcal{F}$ before line 2.

Lemma 11. *At most $2\theta_{i-1}$ added sets (i.e. not included in the original $\mathcal{F}$ at the root) of size i can be eliminated by a fixed set X through the whole 'life' of X in $\mathcal{F}$ along a fixed root-to-leaf path.*

Proof. Observe that X can eliminate anything only immediately after it has been inserted to $\mathcal{F}$. Let us consider this elimination and let us fix $x \in X$. Note that either no set of size i has been added in the ancestors (so none can be eliminated), or Lemma 10 holds, so x can be contained in at most $2\theta_{i-1}$ sets of size i. □

Lemma 12. *Along a fixed root-to-leaf path at most $\beta_i n$ sets of size $\leq i$ are added.*

Proof. We conduct an inductive proof wrt. i. For $i = 1$ this is clear. Now, let us consider sets of size i. By Lemma 9, at most $(\theta_{i-1} - 1)n$ of them remain in the leaf. All other sets must have been eliminated by one of the sets of size $\leq i - 1$, and by the inductive hypothesis the number of such sets is bounded by $\beta_{i-1}n$. However, by Lemma 11, each of them can eliminate $\leq 2\theta_{i-1}$ sets, so at most $2\theta_{i-1}\beta_{i-1}$ sets of size i can be added and then eliminated. In total this gives

$$\beta_{i-1}n + (\theta_{i-1} - 1)n + 2\beta_{i-1}\theta_{i-1}n \leq 4\beta_{i-1}\theta_{i-1}n = (4\alpha\beta_{i-1}^2)n = \beta_i n,$$

so the inductive step has been carried. □

Observe that in each node of the path we add at least one set, so the total length of the path is bounded by $\beta_d n$.

Lemma 13. *Each root-to-leaf path contains at most $\frac{dn}{\alpha}$ right edges.*

Proof. Clearly each right edge involves adding at least θ_i sets of size i for some $i \in \{1, \ldots, d\}$. By Lemma 12 at most $\beta_i n$ sets of size i are added, so sets of size i can be added at most $\frac{\beta_i n}{\theta_i} = \frac{n}{\alpha}$ times. Summing over all $i \in \{1, \ldots, d\}$, this gives $\frac{dn}{\alpha}$ as desired. □

Finally, we are now ready to make the final step, i.e. choose α so that the total number of leaves is $\leq 2^{\varepsilon n}$. The leaves correspond to root-to-leaf paths and such path can be described by a word over the binary alphabet. The number of right edges in a path is bounded by $\frac{dn}{\alpha}$ and the length of a path by $\beta_d n$, so the number of such path, i.e. the number of leaves, can be bounded by

$$\frac{dn}{\alpha}\binom{\beta_d n}{dn/\alpha} \leq \frac{dn}{\alpha} 2^{\beta_d n h(\frac{d}{\alpha\beta_d})},$$

where $h(\delta) = -(\delta \log \delta + (1-\delta)\log(1-\delta))$ is the zero-order entropy function. As $\frac{d}{\alpha\beta_d} < \frac{1}{2}$ for large α, we have $h(\delta) < -2\delta \log \delta$. This gives the following bound

$$\frac{dn}{\alpha} 2^{2\beta_d n \frac{d}{\alpha\beta_d} \log(\frac{\alpha\beta_d}{d})} = \frac{dn}{\alpha} 2^{2n\frac{d}{\alpha}\log(\frac{\alpha\beta_d}{d})} = 2^{o(n) + \mathcal{O}(n)\cdot\frac{\log(\alpha\beta_d)}{\alpha}}$$

Finally observe that

$$\frac{\log(\alpha\beta_d)}{\alpha} = \frac{\log\alpha + (2^{d-1}-1)(2+\log\alpha)}{\alpha} = \mathcal{O}\left(\frac{\alpha}{\log\alpha}\right),$$

so this value can be arbitrarily small as alpha rises, which means that for sufficiently large α the number of output instances is $\mathcal{O}(2^{\varepsilon n})$. □

References

[1] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.