

Plan Last time we have used the first moment method to construct good graphs and good codes.

This time we will see the second moment method and the Lovász local lemma.

- ① second moment method: random graphs, threshold phenomenon.
- ② Lovász local lemma: k-SAT, packet routing, algorithmic proof

Second Moment Method (MU 6.5)

The first moment method is to compute the expected value of a random variable, and conclude that there is an outcome with value at least $E[X]$ or at most $E[X]$.

For a non-negative random variable X , we can use the Markov inequality to prove that $\Pr(X \geq 1) \leq E[X]$, and thus conclude that $X=0$ with high probability if $E[X] \ll 1$ for an integral X .

Often we also want to prove that $\Pr(X \geq 1)$ is large. It is not enough to just show that $E[X]$ is large, as it could be the case that X is very large for a small fraction of the outputs, while $X=0$ for a large fraction of the outputs.

To exclude this case, we need some kind of concentration inequalities (e.g. variance of X is small), and the second moment method provides one way to establish this.

Theorem If X is an integral-valued random variable, then $\Pr(X=0) \leq \frac{\text{Var}[X]}{(E[X])^2}$

Proof By Chebyshev's inequality, $\Pr(X=0) \leq \Pr(|X-E[X]| \geq E[X]) \leq \text{Var}[X]/(E[X])^2$. \square

Corollary If $\text{Var}[X] = o((E[X])^2)$ or $E[X^2] = (1+o(1))(E[X])^2$, then $X > 0$ almost always.

Threshold Behavior in Random Graphs [1]


Let $G_{n,p}$ be a graph with n vertices where each edge appears with probability p .

A property has a threshold behavior if there is a function f such that:

- when $\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = 0$, then almost surely $G(n, g(n))$ does not have this property.
- when $\lim_{n \rightarrow \infty} \frac{h(n)}{f(n)} = \infty$, then almost surely $G(n, h(n))$ has this property.

A property has a sharp threshold behavior if there is a function f such that for any $\varepsilon > 0$,

- $G(n, (1-\varepsilon)f(n))$ almost surely does not have the property.
- $G(n, (1+\varepsilon)f(n))$ almost surely has the property.

For example, consider the property of having a clique of size 4, i.e. .

Let X be the number of 4-cliques in $G_{n,p}$. Then $E[X] = \binom{n}{4} p^6$.

When $p = o(n^{-\frac{2}{3}})$, then $E[X] \rightarrow 0$, and we can conclude that $\Pr(X=0) \rightarrow 1$ by the first moment method.

On the other hand, when $p = \omega(n^{-\frac{2}{3}})$, then $E[X] \rightarrow \infty$, and we can use the second moment method to conclude that $\Pr(X \geq 1) \rightarrow 1$, by showing that $\text{Var}[X] = o((E[X])^2)$. See L05 in 2011 or MU 6.S.1.

Theorem The property of having a clique of size 4 has a threshold function $f(n) = n^{-2/3}$.

Let's see a property with a sharp threshold.

Consider the property that a random graph has diameter less than or equal to two, i.e. for every pair of vertices there is a path of length at most two connecting them.

Theorem The property that $G(n,p)$ has diameter two has a sharp threshold at $p = \sqrt{\frac{2 \ln n}{n}}$.

Proof We say a pair of vertices i and j is a bad pair, if there is no edge between i and j , and no other vertex in G is adjacent to both i and j .

Note that a graph has diameter at most two if and only if there is no bad pair.

For every pair of vertices i and j with $i < j$, let X_{ij} be an indicator variable of (i,j) being a bad pair.

Let $X = \sum_{i < j} X_{ij}$ be the number of bad pairs.

Then $E[X_{ij}] = (1-p)(1-p^2)^{n-2}$, as each of the other $n-2$ vertices is not adjacent to both i and j .

So, $E[X] = \binom{n}{2} (1-p)(1-p^2)^{n-2} \sim \frac{n^2}{2} (1-p^2)^n \sim \frac{n^2}{2} e^{-p^2 n}$.

Set $p = \sqrt{\frac{c \ln n}{n}}$. Then $E[X] \sim \frac{1}{2} n^2 e^{-c \ln n} = \frac{1}{2} n^{2-c}$.

Therefore, if $c > 2$, then $E[X] \rightarrow 0$, and so diameter is two almost surely.

Now, consider the case when $c < 2$, then $E[X] \rightarrow \infty$.

We use the second moment method to show that $X \geq 1$ almost surely.

$$E[X^2] = E\left[\left(\sum_{i < j} X_{ij}\right)^2\right] = E\left(\sum_{\substack{i < j \\ k < l}} X_{ij} X_{kl}\right) = \sum_{\substack{i < j \\ k < l}} E[X_{ij} X_{kl}].$$

We split the sum into three cases: ① all i, j, k, l are different.

② three distinct vertices out of i, j, k, l .

③ two distinct vertices out of i, j, k, l (i.e. the same pair).

For ①, the two variables are independent, and thus $\sum_{i < j, k < l} E[X_{ij} X_{kl}] = \sum_{i < j, k < l} E[X_{ij}] E[X_{kl}]$

$$\leq \sum_{i < j} E[X_{ij}] \sum_{k < l} E[X_{kl}] = (E[X])^2.$$

For ③, the two variables are the same, and thus $\sum_{i < j, k < l} E[X_{ij} X_{kl}] = \sum_{i < j} E[X_{ij}^2]$.

Since X_{ij} is an indicator variable, $X_{ij}^2 = X_{ij}$, and thus this is just $\sum_{i < j} E[X_{ij}] = E[X]$.

For ②, let (i, j) and (i, k) be the two bad pairs. For any other vertex u , either it is not adjacent to i or it is not adjacent to both j and k . This happens with probability

$$(1-p) + p(1-p^2) = 1 - 2p^2 + p^3 \approx 1 - 2p^2.$$

So, $E[X_{ij} X_{ik}] \approx (1-p)^2 \cdot (1-2p^2)^{n-3} \approx (1-2p^2)^n \approx e^{-2p^2 n}$.

Since there are at most $3 \binom{n}{3}$ such triples, the sum in ② is at most $\frac{n^3}{2} e^{-2p^2 n}$.

Now, recall that $p = \sqrt{\frac{c \ln n}{n}}$, this is at most $\frac{1}{2} n^3 e^{-2c \ln n} = \frac{1}{2} n^{3-2c} = o(n^{4-2c}) = o((E[X])^2)$.

Therefore, $E[X^2] \leq (E[X])^2 + o((E[X])^2) + E[X] = (1+o(1))(E[X])^2$.

Hence, by the corollary of the second moment method, we conclude that $X \geq 1$ almost surely.

This implies that the graph has diameter at least three when $p = \sqrt{\frac{c \ln n}{n}}$ for $c < 2$. \square

Evolution of Random Graphs [1]

Let me mention some known results about random graphs, without proofs.

- When $p = o(1/n)$, the graph is a forest of trees, i.e. no cycles.
- When $p = \Omega(1/n)$, cycles appear, but each component has at most one cycle.

And no component has more than $\log n$ vertices.

- When $p = 1/n$, components of $n^{2/3}$ vertices emerge, which are almost surely trees.
- When $p \geq (1+\varepsilon)/n$, then there is a unique ^{giant} component of size $\Omega(n)$, while all other components are of size $O(\log n)$.
- When $p = \frac{1}{2} \frac{\ln n}{n}$, the graph consists only of isolated vertices plus a giant component.
- When $p = \ln n/n$, the graph is connected, and furthermore has a Hamiltonian cycle.

If there are enough interest, I can talk about some proofs in a later lecture along with web graphs.

Random Structures

Random Structures

Other random structures also have this phenomenon of "phase transition".

For random 3-SAT formula where each clause has three random variables (or their negations).

It is conjectured and generally believed that when the clause-to-variable ratio is less than 4.2, then the formula is almost surely satisfiable, and when this ratio is greater than 4.2, then the formula is almost surely unsatisfiable.

The same happens for k -SAT, with a sharp threshold at around $2^k \ln 2 - \frac{3}{2} \ln 2$.

These behaviors are similar to some physical phenomenon.

Algorithmic Issues

The second moment method can be used to show that $G_{n,1/2}$ has a clique of size $2\log_2 n$ almost surely.

While it is easy to find a clique of size $\log_2 n$ (a simple greedy algorithm would work), it is not known how to find a clique of size $(1+\epsilon)\log_2 n$ in polynomial time, and in fact there is some evidence suggesting it may be computationally hard.

Similarly, there is no known polynomial time algorithms to determine whether a random 3-SAT formula with clause-to-variable ratio 4.2 is satisfiable or not. In fact, these are some hardest instances for 3-SAT that we know how to generate efficiently.

Lovász Local Lemma (MU 6.7)

Let E_1, E_2, \dots, E_n be a set of "bad" events.

A typical goal is to show that there exists an output with no bad events occur.

For example, in k -SAT, we want to find an assignment with no clauses violated (bad events).

That is, we want to show that $\Pr(\bigcap_{i=1}^n \bar{E}_i) > 0$.

There are two situations when this is easy to show:

- when the events E_1, \dots, E_n are mutually independent
- when $\sum_{i=1}^n \Pr(E_i) < 1$, in other words, when the union bound applies.

Lovász local lemma can be seen as a clever combination of them.

We say that an event E is mutually independent of the events E_1, E_2, \dots, E_n

if for any subset $I \subseteq [n]$, $\Pr(E | \bigcap_{j \in I} E_j) = \Pr(E)$.

Definition A dependency graph for a set of events E_1, \dots, E_n is a graph $G=(V, E)$ s.t.
 $V=\{1, \dots, n\}$ and for $1 \leq i \leq n$ event E_i is mutually independent of the events $\{E_j \mid (i, j) \notin E\}$.

Theorem (Lovász local lemma) Let E_1, \dots, E_n be a set of events. Suppose the followings hold:

- ① $\Pr(E_i) \leq p$
- ② The max degree in the dependency is at most d
- ③ $4dp \leq 1$.

Then $\Pr(\bigcap_{i=1}^n \bar{E}_i) > 0$.

proof We prove by induction that $\Pr(\bigcap_{i \in S} \bar{E}_i) > 0$ on the size of S .

To prove this, there is an intermediate step showing that $\Pr(E_k \mid \bigcap_{i \in S} \bar{E}_i) \leq 2p$.

The proof structure is like this: $\Pr(\bigcap_{i \in S, |S|=1} \bar{E}_i) > 0 \Rightarrow \Pr(E_k \mid \bigcap_{i \in S, |S|=1} \bar{E}_i) \leq 2p$
 $\Rightarrow \Pr(\bigcap_{i \in S, |S|=2} \bar{E}_i) > 0 \Rightarrow \Pr(E_k \mid \bigcap_{i \in S, |S|=2} \bar{E}_i) \leq 2p$
 $\Rightarrow \dots$
 $\Rightarrow \Pr(E_k \mid \bigcap_{i \in S, |S|=n-1} \bar{E}_i) \leq 2p \Rightarrow \Pr(\bigcap_{i \in S, |S|=n} \bar{E}_i) > 0$

First we prove $\Pr(\bigcap_{i \in S} \bar{E}_i) > 0$ assuming the previous steps in the chain are proven.

The base case when $|S|=1$ is easy, since $\Pr(\bar{E}_i) = 1 - \Pr(E_i) = 1 - p > 0$.

For the inductive step, without loss of generality assume $S = \{1, 2, \dots, s\}$.

$$\Pr(\bigcap_{i=1}^s \bar{E}_i) = \prod_{i=1}^s \Pr(\bar{E}_i \mid \bigcap_{j=1}^{i-1} \bar{E}_j) = \prod_{i=1}^s (1 - \Pr(E_i \mid \bigcap_{j=1}^{i-1} \bar{E}_j)) \geq \prod_{i=1}^s (1 - 2p) > 0.$$

Next we prove $\Pr(E_k \mid \bigcap_{i \in S} \bar{E}_i) \leq 2p$ assuming the previous steps are proven.

To do this we first divide the events into two types, based on its dependency to E_k :

$$S_1 = \{i \in S \mid (k, i) \in E\} \text{ and } S_2 = \{i \in S \mid (k, i) \notin E\}.$$

If $|S|=|S_2|$, then $\Pr(E_k \mid \bigcap_{i \in S} \bar{E}_i) = \Pr(E_k) \leq p$ and we're done.

Henceforth we assume $|S| > |S_2|$.

Let $F_S = \bigcap_{i \in S} \bar{E}_i$ and similarly define F_{S_1} and F_{S_2} . Note $F_S = F_{S_1} \cap F_{S_2}$.

$$\Pr(E_k \mid F_S) = \frac{\Pr(E_k \cap F_S)}{\Pr(F_S)} \quad \uparrow \quad = \frac{\Pr(E_k \cap F_{S_1} \mid F_{S_2}) \Pr(F_{S_2})}{\Pr(F_{S_1} \mid F_{S_2}) \Pr(F_{S_2})} = \frac{\Pr(E_k \cap F_{S_1} \mid F_{S_2})}{\Pr(F_{S_1} \mid F_{S_2})}$$

we do this because we want to take advantage of the independence of E_k and F_{S_2}
 \downarrow

The numerator is $\Pr(E_k \cap F_{S_1} \mid F_{S_2}) \leq \Pr(E_k \mid F_{S_2}) = \Pr(E_k) \leq p$.

The denominator is $\Pr(F_{S_1} \mid F_{S_2})$

$$\begin{aligned}
&= \Pr\left(\bigcap_{i \in S_1} \bar{E}_i \mid \bigcap_{j \in S_2} \bar{E}_j\right) = 1 - \Pr\left(\bigcup_{i \in S_1} E_i \mid \bigcap_{j \in S_2} \bar{E}_j\right) \\
&\geq 1 - \sum_{i \in S_1} \Pr(E_i \mid \bigcap_{j \in S_2} \bar{E}_j) \leftarrow (\text{union bound}) \\
&\geq 1 - \sum_{i \in S_1} 2p \leftarrow (\text{induction hypothesis, because } |S_2| < |S|) \\
&\geq 1 - 2dp \\
&\geq 1/2.
\end{aligned}$$

Plug it back, $\Pr(E_k | E_S) = \frac{\Pr(E_k \cap F_{S_1} | F_{S_2})}{\Pr(F_{S_1} | F_{S_2})} \leq \frac{p}{1/2} = 2p.$ \square

Applications We show one easy and one more advanced application.

① k-SAT (MU 6.7.2)

A boolean formula with exactly k variables in each clause. We would like to find an assignment of T/F to each variable s.t. all the clauses are satisfied.

This problem is NP-complete.

But we can prove that if each variable appears in not too many clauses, then there is always a satisfying assignment.

Theorem If no variable in a k -SAT formula appear in more than $T = 2^{k/4k}$ clauses, then the formula has a satisfying assignment.

Proof Consider a random assignment where each variable is true with prob. $1/2$ independently.

Let E_i be the bad event that clause i is violated by the random assignment.

Since each clause has k variables, $p = \Pr(E_i) = 2^{-k}$.

The event E_i is mutually independent of all other events corresponding to clauses that do not share variables with E_i .

So, $d \leq kT = 2^{k-2}$. Hence $4dp \leq 1$.

By local lemma, $\Pr(\bigcap_{i=1}^m \bar{E}_i) > 0$, so there is an assignment satisfying all clauses. \square

② packet routing [3,4]

We are given an undirected graph, N pairs, each pair has a source s_i , a destination t_i and a path P_i .

In each time step, at most one packet can traverse an edge.

In each time step, at most one packet can traverse an edge.

A packet can wait at any node at any time step.

A schedule for a set of packets specifies the timing of the movement of packets along their respective paths (i.e. when to move and when to wait).

The goal is to find a schedule to minimize the total time to route all the packets.

Let d be the maximum distance traveled by any packet (i.e. maximum path length)

Let c be the max. number of packets that must traverse a single edge.

Then, it is clear that any schedule must require $\Omega(cd)$ steps to finish.

A surprising result by Leighton, Maggs, and Rao shows that there is always a schedule using $O(cd)$ steps.

This is a very strong result, as the bound is independent of n .

While usually we use Chernoff bound and union bound, the new idea here is to use Chernoff bound and local lemma.

For simplicity we only prove a weaker result, but is independent of N .

Theorem There is a schedule with length $O((cd)(1+\alpha)^{O(\ln^*(cd))})$, where α is a constant and $\ln^*(n)$ is the number of \ln it takes to bring n to ≤ 1 .

($\log^*(n)$ is growing extremely slowly, e.g. $\log_2^*(2^{65536}-1) = 5$.)

Proof Without loss of generality assume $c=d$.

For each packet, assign an initial delay from $[1, \alpha d]$ for some constant α .

We first consider a relaxed version of the problem, where the packet can go without any interruption (i.e. without waiting at a node) towards its destination.

So, the total time needed (in this relaxed version) is at most $(1+\alpha)d$.

Partition the time into periods, each period having $\ln d$ steps.

We will show that with positive probability that each edge has congestion at most $\ln d$

(equal to $\ln c$ by our assumption) in each period, i.e. $\leq \ln d$ packets using that edge in that period.

Then, for each period, we can think of it as a sub-problem with $c' = \ln d$ and $d' = \ln d$.

Then, for each subproblem, we apply the same argument recursively.

i.e. each subproblem can be partitioned into $\ln d' = \ln \ln d$ periods,

s.t. each edge has congestion $\leq \ln d = \ln \ln d$ in each period, and so on.

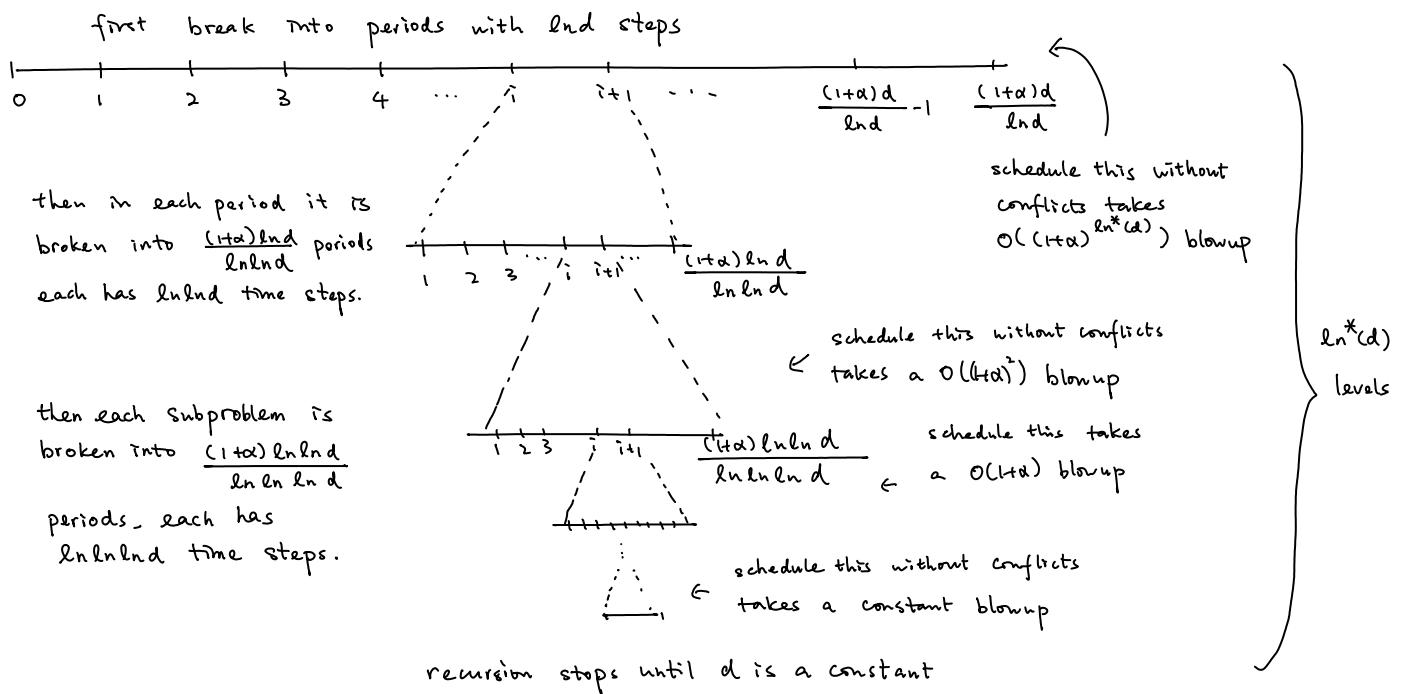
So, we recursively partition the problem until c and d become constant, and then we just find a naive $O(cd)$ solution (e.g. in arbitrary order). In these base cases this is also a $O(c+d)$ solution since c and d are constants.

To reduce the problem into constant congestion and maximum path length, we just need $O(\ln^* d) = O(\ln^*(c+d))$ recursions.

In each recursion we blow up the schedule by a factor of $(1+\alpha)$.

So the total time in the schedule is $O((c+d)(1+\alpha)^{O(\ln^*(c+d))})$ steps, and each edge is used by at most one packet in each time step.

Pictorially, the proof goes like this



Okay, it remains to prove the following lemma using local lemma.

Lemma When α is a large constant, each period has congestion $\leq \ln d$ with positive probability.

Let A_f be the bad event that edge f has congestion $> \ln d$ in some period.

First we bound the maximum degree in the dependency graph.

Note that whether A_e happens depends only on the at most c packets that use e .

Two events A_e and A_f are independent unless there is some packet use both e and f .

Since there are at most c packets and each such packet passes through at most d edges,

A_e is dependent on at most cd events. Hence the max. degree $\leq cd = d^2$.

Now we bound the probability that A_e happens. This is just a typical application of the Chernoff bound.

Each period is of length $\ln d$. Since there are at most $c=d$ packets that use e , $E[\# \text{ of packet using } e \text{ at a specific period}] \leq \frac{\ln d}{\alpha d} \cdot d = \frac{\ln d}{\alpha}$. Let this number be μ .

$$\begin{aligned} \text{By Chernoff bound, } & \Pr[\text{congestion of } e \text{ at a specific period} > \ln d] \\ &= \Pr[\text{congestion of } e \text{ at a specific period} > (1+\alpha-1)\mu] \\ &\leq \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}} \right)^\mu \leq \left(\frac{e}{1+\delta} \right)^{(1+\delta)\mu} \\ &= \left(\frac{e}{\alpha} \right)^{\alpha \cdot \frac{\ln d}{\alpha}} \quad \text{since } \mu = \ln d / d \text{ and } \delta = \alpha - 1 \\ &= \left(\frac{e}{\alpha} \right)^{\ln d} < \frac{d^{-4}}{\alpha} \quad \text{for say } \alpha \geq e^{11} \text{ (a large constant)} \end{aligned}$$

Since there are at most αd time frames, $\Pr(A_e) \leq \alpha d \cdot d^{-4} / \alpha = d^{-3}$.

So, $4 \cdot d^{-3} \cdot cd = 4d^{-1} \leq 1$ for $d \geq 4$. Hence, by local lemma, there is a choice of the initial delays such that no period has congestion more than $\ln d$.

This proves the lemma and hence the theorem. \square

Efficient Algorithms for Local Lemma

How to find an outcome (e.g. a satisfying assignment) whose existence is guaranteed by the local lemma? In fact, since the probability could be very small, we don't expect that a random outcome will do, and it seems to be a very difficult algorithmic task like "finding a needle in a haystack".

There is a long history about finding efficient algorithms for local lemma, with a recent breakthrough. To illustrate the ideas, we just focus on the k -SAT problem.

Original proof : It is nonconstructive, giving no idea how to find such an outcome.

Early results (MU 6.8) There is a framework developed by Beck.

Let me just try to give a very brief idea here. See MU 6.8 for details.

For this framework to work, a stronger condition is assumed: each variable appears in at most

$$T = 2^{\alpha k} \text{ clauses for some } 0 < \alpha < 1 \text{ (instead of } T = 2^k / 4k \text{)}.$$

The algorithm has two phases:

① Find a random "partial" assignment (each clause with at least $k/2$ variables remain unassigned).

Using the local lemma itself with the stronger assumption ($T = 2^{\alpha k}$), it can be proved that the partial solution can be extended to a full solution. This step is easy if α is small enough.

② After the initial partial assignment, prove that the dependency graph is broken into small pieces, where each piece has at most $O(\log m)$ events. Since each clause has k variables, we can do exhaustive search in each piece in polynomial time to find a satisfying assignment whose existence is guaranteed by the local lemma in phase ①.

The difficult part is to show that each piece is of size $O(\log m)$, by a careful counting argument.

(See MU 6.8 or LOS in 2011 for a slightly more detailed outline).

Recent Breakthrough by a Ph.D. student.

The algorithm is surprisingly simple.

First fix an ordering of the clauses, say C_1, C_2, \dots, C_m

Solve-SAT

Find a random assignment of the variables.

For $1 \leq i \leq m$

if C_i is not satisfied

FIX(C_i)

FIX(C)

Replace the variables in C_i by new random values.

While there is a clause D that share variables with C and D is not satisfied

Choose such D with the smallest index

FIX(D)

Note that once we called FIX(C_i), C_i will remain satisfied after each

FIX(C_j) for $j > i$, by the definition of FIX(C_j).

So, we just need to prove that FIX(C_i) terminates in a reasonable amount of time, although at first glance it seems that it could have gone into infinite loop,

Moser came up with a remarkable proof, proving that this algorithm terminates very quickly, otherwise one can obtain an algorithm to compress l random bits

using fewer than l bits, which is impossible by (a strengthening of) the arguments in week 4. A surprising connection.

First, suppose the algorithm has run for t steps but not successful yet, how many random bits it has used?

Initially, we used n random bits for the initial assignment.

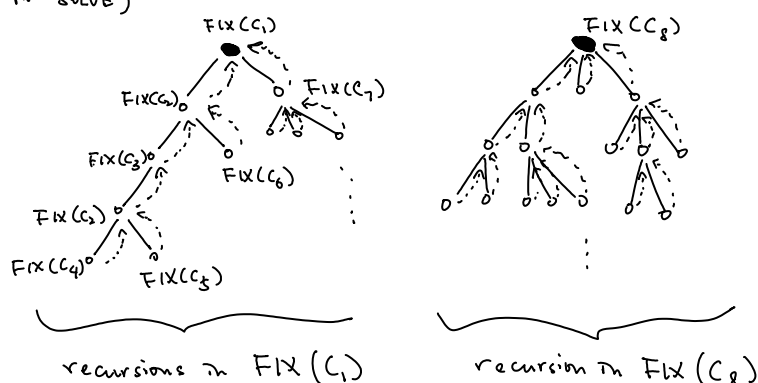
Then, for each $\text{Fix}(C)$, we used k random bits.

So, the total number of random bits used is $n + tk$.

Now, we show how to compress the random bits if t is large.

The idea is to trace the execution of the algorithm.

(in the main loop in `SOLVE`)



An encoding scheme is as follows:

- ① use $00 + \log(m)$ bits to represent the clauses in the root nodes
- ② use $01 + \log(d)$ bits to represent the next clause fixed in the recursion tree. why $\log(d)$ bits are enough? Because each clause shares variables with at most d other clauses
- ③ use 10 to represent the end of a recursive call, that is, to represent the back arrows in the figure.
- ④ When the algorithm ends, remember the n bits in the variables.

How many bits we used?

- There are at most m roots, since after calling $\text{Fix}(C_i)$, C_i will remain satisfied after calling $\text{Fix}(C_j)$ for $j > i$. So at most $m(\log m + 2)$ bits are used for ①.
- There are t steps in the algorithm. So at most $t(\log d + 2)$ bits for ② + ③.
- Finally n bits are needed for ④.

- Finally n bits are needed for ④.

Therefore, the total number of bits used is $m(\log m + 2) + t(\log d + 2) + n$.

Okay, if we can show that we can reconstruct the original random bits from the encoding, then since random bits cannot be compressed, we must have

$$m(\log m + 2) + t(\log d + 2) + n \geq tk + n$$

$$\Rightarrow m(\log m + 2) \geq t(k - \log d - 2)$$

So, if $d < 2^{k-2}$, then $t = O(m \log m)$. That is, the expected value of t is $O(m \log m)$, i.e. the algorithm terminates quickly.

Finally, we just need to show that with the encoding one can reconstruct the original random bits.

Let the original random bits be

$$v_1, v_2, \dots, v_n, r_1^{(1)} r_2^{(1)} \dots r_k^{(1)} r_1^{(2)} r_2^{(2)} \dots r_k^{(2)} \dots r_1^{(t)} r_2^{(t)} \dots r_k^{(t)}$$

where v_1, \dots, v_n are the initial random bits on the variables,

and $r_1^{(i)} \dots r_k^{(i)}$ are the random bits used in the i -th step.

So the algorithm will maintain n variables, initially it is v_1, \dots, v_n .

The algorithm does not know the values of these variables, and it tries to find out.

Now the algorithm follows the execution tree to figure out the variables.

If the algorithm fixes the clause i , say clause i has variables $\{x_1, x_2, \dots, x_k\}$,

then it must be the case that $\{v_1, v_2, \dots, v_k\}$ must violate the clause i .

The crucial observation is that there is only ONE setting (out of 2^k settings)

clause i unsatisfied. So we can recover the values of the variables v_1, v_2, \dots, v_k .

Then we know these variables will be replaced by $r_1^{(1)} r_2^{(1)} \dots r_k^{(1)}$.

The compression algorithm will maintain the variables $\{r_1^{(1)}, r_2^{(1)}, \dots, r_k^{(1)}, v_{k+1}, \dots, v_n\}$ in the next step.

Then, again, we know which clause is going to be fixed next, we learn

k bits of $\{r_1^{(1)}, \dots, r_k^{(1)}, v_{k+1}, \dots, v_n\}$ and can replace k variables by

$r_1^{(2)} \dots r_k^{(2)}$, and so on, until the algorithm stops. Since we have stored the

final n bits, we can recover the values of all original bits. \square

final n bits, we can recover the values of all original bits. \square

There are many follow-up work in this direction ; see the project page.

It is very interesting that probabilistic methods can be made efficient.,

e.g. now we can use this method to find a good schedule for the packet routing problem.

In other words, local lemma has become an algorithmic tool, i.e. if we can prove that it exists by the local lemma, then we can also find it efficiently.

References

- [1] Hopcroft, Kannan. Computer Science Theory for the Information Age. Book draft, chapter two.
- [2] Alon, Spencer. The probabilistic method. Chapter 4.5 in 2nd edition.
- [3] Leighton, Maggs, Rao. Packet Routing and Job-Shop Scheduling in $O(\text{congestion} + \text{dilation})$ steps.
- [4] Rothvoss. A simpler proof of $O(\text{congestion} + \text{dilation})$ packet routing
- [5] Moser. A constructive proof of the Lovász local lemma.