NEEDLES IN EXPONENTIAL HAYSTACKS II

Notes of Joel Spencer
Cornell Probability School 2012

1 The Lovász Local Lemma

We here describe the LLL in somewhat different, and somewhat more limited, format that done in other work, including our own. Still, this is a quite natural format and pretty much all applications of LLL fit it.

There is a finite universe Ω and for each $j \in \Omega$ there is a random variable $COLOR[j]$. These could be any variables but in practice, and the examples below, they are binary "coin flips," either $\{Red, Blue\}$ or $\{True, False\}$. The distributions can be different for different $j \in \Omega$ but – and this is critical for the entire result – we must assume that the values $COLOR[j]$ are mutually independent over $j \in \Omega$. That is, the $j \in \Omega$ may have different coins (or die, or whatever) but each one is "flipped" independently. In our algorithmic analysis we assume that any $COLOR[j]$ may be generated randomly in unit time. There is an index set I and for each $\alpha \in I$ a set $\Xi_\alpha \subset \Omega$ [1] and an event BAD_α which depends only on the $COLOR[j]$, $j \in \Xi_\alpha$. Let Σ denote the family of Ξ_i. Jumping to the end, the desired conclusion is

$$[\texttt{SIEVE}] : \wedge_{\alpha \in I} \overline{BAD_\alpha} \neq \emptyset \tag{1}$$

Of course, we don't always obtain `[SIEVE]`. Our object will be to find conditions that are relatively easy to check that can imply `[SIEVE]`. We shall say $\alpha, \beta \in I$ *overlap* if $\Xi_\alpha \cap \Xi_\beta \neq \emptyset$ and in this case we write $\alpha \sim \beta$. (In this presentation, unlike elsewhere, we write $\alpha \sim \alpha$.

The most used form of the LLL is when there is a symmetry in the bad events. Then we have the following result. (Note that because we are counting $\alpha \sim \alpha$ the value d is one off from other presentations.)

Theorem 1.1 `LLL - Symmetric Form` *Under the above circumstance suppose*

1. *All* $\Pr[BAD_\alpha] \leq p$.
2. *Each* $\alpha \in I$ *has* $\alpha \sim \beta$ *for at most* d *indices* $\beta \in I$

[1] Technically, there could be several $\alpha \in I$ on the same set Ξ_α

3.

$$p \leq \frac{(d-1)^{d-1}}{d^d} \tag{2}$$

Then `[SIEVE]`.

This will come out as a calculation from the basic approach.

2 Applications

Suppose we have n Boolean variables $x_1, \ldots, x_n$ and we consider an instance of $k - SAT$. Each clause is the disjunction of k variables, some possibly with negations. For example, $x_{13} \vee \overline{x}_{67} \vee x_{123}$ could be such a clause for $3 - SAT$. Set $\Omega = \{x_1, \ldots, x_n\}$. Let I index the clauses and for clause C_α let Ξ_α denote the underlying set of variables (without negations), here $\Xi = \{x_{13}, x_{67}, x_{123}\}$. As the underlying probability space we assign truth values t, f to the variables independently and unformly. The events BAD_α are that C_α is not satified. Clearly $\Pr[BAD_\alpha] = 2^{-k}$ for any such clause. Now `[SIEVE]` is precisely the event that

$$\wedge_\alpha \overline{C_\alpha} \text{ is satisfiable} \tag{3}$$

So in this case Theorem 1.1 states: An instance of $k - SAT$ in which every clause has a common variable (neglecting negation symbols) with at most d clauses (counting itself) will be satisfiable when

$$2^{-k} \leq \frac{(d-1)^{d-1}}{d^d} \tag{4}$$

It is usually more convenient (especially for d large) to estimate the RHS and use the criterion

$$e2^{-k}d \leq 1 \tag{5}$$

A quite similar application comes when we are given a family of subsets S_α, $\alpha \in I$, of a universe Ω and we want a 2-coloring of Ω such that no S_α. Here we assign $COLOR[j]$ to Red or Blue uniformly and randomly. The bad events BAD_α that S_α is monochromatic have $\Pr[BAD_\alpha] = 2^{1-k}$. Thus if each S_α overlaps at most d sets S_β and

$$e2^{1-k}d \leq 1 \tag{6}$$

there exists such a 2-coloring.

These examples indicate the algorithmic problem. Suppose we fix $k = 5$, $d = 10$ so that (5) holds and consider instances of $5 - SAT$ with no clause having common variable with more than 10 clauses. How do we find the assignment that satisfies $\wedge\overline{C_\alpha}$. There are 2^n assignments and one can give examples (remember that k, d are fixed here and $n \to \infty$) where a random assignment would only have an exponentially small probability of working. The method we give here, originating with the breakthrough results of Robin Moser, then a graduate student (!) at ETH (Zurich), will actually find one such assignment in polynomial (indeed, linear!!) time.

3 Moser's Fix-It Algorithm

3.1 FIXIT!

We write the Moser algorithm **FIXIT** as follows:

1. [**FIXIT1**] Set $COLOR[j]$, all $j \in \Omega$
2. [**FIXIT2**] WHILE *some* BAD_α
3. [**FIXIT2a**] Select a particular α with BAD_α
4. [**FIXIT3**] Reset $COLOR[j]$ for $j \in \Xi_\alpha$

Step [**FIXIT2a**] is a wild card. If there are several α which to we choose. Our analysis of the algorithm will, for the most part, assume *any* particular selection choice (e.g., Order I and take the first such α) but we shall return to this point later.

3.2 Time and Log

When we run **FIXIT** let $X_1, X_2, \ldots,, X_i \in \Sigma$, denote the Ξ_α, in order, called by **FIXIT3** and define the LOG [2] to be the ordered sequence

$$LOG = (T_1, T_2, \ldots, T_u, \ldots) \tag{7}$$

A priori the LOG could be empty (we got lucky with the first flipping at **FIXIT1** or it could be infinite, and $TLOG$ denotes its length. (Bear in mind that these values all have distributions as they depend on the various choices of $COLOR[j]$.) We will define the time, denoted $TLOG$, of the **FIXIT** algorithm to be the number of times step [**FIXIT3**] was employed, the length (possibly infinite) of the LOG. The relationship between $TLOG$ and the actual running time will be discussed later.

In our analysis we shall deal with finite initial strings $T_1 \cdots T_u$ of LOG. (We call these *prefixes* of LOG. These are strings from the alphabet Σ. We let, following standard usage, Σ^* denote the family of finite strings from the alphabet Σ.)

The **FIXIT** algorithm is simplicity itself and those of us that struggled for decades to find algorithmic implementations of LLL were certainly surprised. The subtlety is in the analysis of the algorithm. Observe that the WHILE loop can end only $\wedge \overline{BAD_\alpha} \neq \emptyset$. Thus: If $TLOG$ is ever finite then, via Erdős Magic, `[SIEVE]`. We actually will deal with something more:

$$[\texttt{ENDTIME}] \qquad\qquad E[TLOG] < \infty \tag{8}$$

If `ENDTIME` then certainly the **FIXIT** algorithm can end, indeed more more. And if the **FIXIT** algorithm can end then `SIEVE`. We shall find criteria that imply `ENDTIME`.

[2] as in ship's log, a record of what happened

4 Preprocessing Randomness

The preprocessing of randomness is a powerful conceptual tool for analyzing randomized algorithms. Here, for each $j \in \Omega$, we create i.i.d. $COLOR[j,t]$ for $t = 0, 1, 2, \ldots$. Now **[FIXIT1]** uses $COLOR[j,0]$ and **[FIXIT3]** uses $COLOR[j,t+1]$ where $COLOR[j,t]$ was the last value "used." Once all the $COLOR[j,t]$ have been determined **FIXIT** then run deterministically. (This is assumming some selection procedure for **[FIXIT2a]** has been settled on.) We will consider the countable number of choices of the $COLOR[j,t]$ to be our underlying probability space in analyzing **FIXIT**. Sometimes the phrase "fictional continuation" is used here. For each j there are a countable number of "coin flips" $COLOR[j,t]$ made even though, a priori, we might only need the $COLOR[j,0]$.

5 Lets Play Tetris!

5.1 Creating the Tetris Picture

As an illustrative example let us take $\Omega = \{1,2,3,4,5,6,7,8\}$ and let $\Sigma = \{A,B,C,D,E,F\}$ with $A = \{1,2,3\}$, $B = \{2,3,4\}$, $C = \{3,4,5\}$, $D = \{4,5,6\}$, $E = \{5,6,7\}$, $F = \{6,7,8\}$. In the general situation there is no need for the $\Xi_\alpha \subset \Omega$ to be made up of consecutive values but this will yield prettier pictures. Lets consider a nonempty string in Σ^*, say $s = ADCFECBF$. Drop down the tetris pieces in that order.

```
                              FFFFFFFFFFFF
      BBBBBBBBBBBBBB
           CCCCCCCCCCCCC
                     EEEEEEEEEEEEEE
           CCCCCCCCCCCCC  FFFFFFFFFFFFF
AAAAAAAAAAAAA  DDDDDDDDDDDDD
 1    2    3    4    5    6    7    8
```

From s we create a substring, called $PYR(s)$ (for pyramid) by taking the last tetris piece and all other pieces that are supporting it. In this case we get $PYR(s) = ADCFEF$.

```
                              FFFFFFFFFFFF
                         EEEEEEEEEEEEEE
           CCCCCCCCCCCCC  FFFFFFFFFFFFF
AAAAAAAAAAAAA  DDDDDDDDDDDDD
 1    2    3    4    5    6    7    8
```

More precisely, we can describe $PYR(s)$ algorithmically by working backwards on string s and deciding which terms to accept. We accept the last term and then we accept a term if and only if it overlaps with a term already accepted.

We call a nonempty string $s \in \Sigma^*$ a *pyramid* if $PYR(s) = s$, that is, all the tetris pieces are supporting the last one. Equivalently, $s = X_1 \cdots X_u$ is

a pyramid if for all $1 \leq v < u$ there exists $v \leq w \leq u$ with $X_v \cap X_w \neq \emptyset$. Let PYR denote the family of all pyramids s.

5.2 Getting the Same Tetris Picture

Observe in our example that if we flip the first AD and/or the following CF in our string (for example, from $s = ADCFEF.$ to $s' = DAFCEF$), we get the same tetris picture. Can we say, in general, when two $s, s' \in PYR$ give the same tetris picture. Yes we can!

Let us define a semigroup on alphabet Σ in which Ξ_α, Ξ_β commute if and only if $\Xi_\alpha \cap \Xi_\beta = \emptyset$. (Such semigroups have been studied by Foata, Viennot and others decades ago and are interesting for their own sake, though here we use only elementary properties.) We can write $s \sim s'$ if one can get from s to s' via a sequence of these allowable transpositions. This is an equivalence relation and we let $\overline{s}$ denote the equivalence class containing s. Strings equivalent to pyramids are necessarily pyramids so we can think of the set of pyramid equivalence classes, denoted $\overline{PYR}$.

Let $s = X_1 \cdots X_r \in PYR$. For any $1 \leq t \leq r$ and any $j \in X_t$ define the flip number $flip(t, j)$ to be the number of s, $1 \leq s \leq t$, with $j \in X_s$. (This includes $s = t$ so the flip number is at least one.) For example, with $s = X_1X_2X_3X_4X_5X_6 = ADCFEF$, $flip(5, 3) = 3$ as 3 appears in A, C, E. If we interchange XY to YX (with, recall, $X \cap Y \neq \emptyset$, we do not change the flip numbers of the X and Y.

Theorem 5.1 *Let $s, s' \in PYR$. Then the following are equivalent:*

1. *$s \sim s'$*
2. *Write $s = X_1 \cdots X_r$ and $s' = Y_1 \cdots Y_l$. Then each $\Xi \in \Sigma$ appears the same number of times in s, s'. (So $r = l$). When X_t, $Y_{t'}$ are both the r-th appearance of the same letter Ξ in their respective sequences then the flip numbers $flip(t, j)$ for s and $flip(t', j)$ for s' are the same.*

`Proof:` There are two parts.
(1 implies 2): Suppose s' is reached from s by a single transpostion. That is, $s = a\Xi\Gamma b$ and $s' = a\Gamma\Xi b$ with $a, b \in \Sigma^*$ and $\Gamma \cap \Xi = \emptyset$. Transposing Γ, Ξ does not change the respective flip numbers as they do not overlap. Then any sequences of transpositions of this form preserves the flip numbers.
(2 implies 1): Suppose the first letter of s is A and the first appearance of A is s' is as Y_j. The flip numbers for all $j \in A$ are one in s and therefore must be one for Y_j. This means that Y_j cannot overlap any of the earlier sets $Y_1, \ldots, Y_{j-1}$. Thus in s' we can use commutativity to move Y_j to the front, to start $Y_jY_1, \ldots, Y_{j-1}$. Now s, s' both start with A. Removing A from both keeps all the flip numbers the same so we can continue the process (or, more formally, use induction on the length) until s' is transposed to s.

6 Reaching a Pyramid

Let $s = X_1 \cdots X_t$ be a pyramid. For each $1 \leq l \leq t$ and each $j \in X_l$ we have a flip number $flip(j, l)$. Let $BAD[l]$ denote that, writing $X_l = \Xi_\alpha$,

BAD_α occurs with $COLOR[j] = flip(j,l)$ for all $j \in \Xi_\alpha$. Let $REACH[s]$ denote that all $BAD[l]$, $1 \le l \le t$, occur. As the flip numbers are preserved in the equivalence class we consider this event as $REACH[\overline{s}]$, defined for $\overline{s} \in \overline{PYR}$.

The connection between $REACH[\overline{s}]$ and our **FIXIT** algorithm is a bit subtle, depending on the ambiguities inherent in **FIXIT2a**, our selection of which BAD_α to fix. First suppose **FIXIT2a** is *trying* to start the LOG with some particular representative s of the class $\overline{s}$. By that I mean that at the i-th application, $1 \le i \le t$, of **FIXIT2a** X_i is chosen if it can be, that is, if its corresponding bad event is occuring at that moment. Employing that **FIXIT2a**, $REACH[\overline{s}]$ is precisely the event that LOG starts with s.

For a partial converse consider the event we call $\texttt{PREFIX}[\overline{s}]$. This event is that for some prefix $X_1 \cdots X_u$ of LOG, $PYR[X_1 \cdots X_u] \sim s$.

We have to be careful here. $\texttt{PREFIX}[\overline{s}]$ is not an event in our probability space, which we defined as the choices $COLOR[j,t]$. $\texttt{PREFIX}[\overline{s}]$ can depend on the choice mechanism **FIXIT2a**. But a *necessary condition* for $\texttt{PREFIX}[\overline{s}]$ to hold is that $REACH[\overline{s}]$ holds. For if $s = PYR[X_1 \cdots X_u]$ then the remaining elements of $X_1 \cdots X_u$ would not affect the flip numbers (otherwise they would be in the pyramid!) and so the various BAD events would have to occur with the given flip numbers.

`Example:` Continuing our Tetris example above we look at the event that, say $CD = PYR[X_1 \cdots X_u]$ for some prefix of LOG. (As C, D overlap, $\overline{CD} = CD$.) Let BAD_C, BAD_D denote the bad events corresponding to C, D. For all we know, $C = X_{57}$, the algorithm **FIXIT** was busy fixing up other stuff. But in that case we would know that none of $X_1, \ldots, X_{56}$ would have overlapped C, as had they done so they would have been put in the pyramid. Thus BAD_C must have occured with $COLOR[3,0]$, $COLOR[4,0]$, $COLOR[5,0]$. Now say $D = X_{83}$. Then of $X_1, \ldots, X_{82}$ only $C = X_{57}$ overlaps D. Thus BAD_D must have occured with $COLOR[4,1]$, $COLOR[5,1]$, $COLOR[6,0]$.

6.1 A Surprising Independence

Let $s = X_1 \cdots X_t \in PYR$ and let BAD_i, $1 \le i \le t$ denote the bad event corresponding to $X_i \in \Sigma$.

Theorem 6.1

$$\Pr[REACH[\overline{s}]] = \Pi_{i=1}^{t} \Pr[BAD_i] \tag{9}$$

The example above indicates what is happening. Let BAD_C, BAD_D denote the bad events corresponding to C, D. In our original probability space these events are dependent as they both use $COLOR[4]$ and $COLOR[5]$. But for $REACH[\overline{CD}]$ we need precisely that BAD_C occurs with $COLOR[3,0]$, $COLOR[4,0]$, $COLOR[5,0]$ and that BAD_D occurs with $COLOR[4,1]$, $COLOR[5,1]$, $COLOR[6,0]$. No $COLOR[j,t]$ is in common and so these are now independent events.

The argument works in general. The event $REACH[\overline{s}]$ is that for $1 \le i \le t$, BAD_i occurs with the appropriate flip numbers. When a value $j \in \Omega$ occurs a second time it has a different flip number. So the events BAD_i are mutually independent.

7 Analyzing FIXIT

Given a running *LOG* of **FIXIT** the various prefixes generate various pyramids. These pyramids cannot be equivalent.

Theorem 7.1 *Let $t < s$ and $X_1, \ldots, X_s \in \Sigma$. Then $PYR(X_1 \cdots X_t)$ and $PYR(X_1 \cdots X_s)$ cannot be equivalent.*

`Proof:` As $PYR(X_1 \cdots X_t)$ ends with X_t and $PYR(X_1 \cdots X_s)$ ends with X_s and the final term in a pyramid can't be moved under $\sim$ we would need $X_t = X_s$. Then X_t would be part of $PYR(X_1 \cdots X_s)$ and thus all of $PYR(X_1 \cdots X_t)$ would be part of $PYR(X_1 \cdots X_s)$ and so $PYR(X_1 \cdots X_s)$ would be strictly longer than $PYR(X_1 \cdots X_t)$.

For each $\overline{s} \in \overline{PYR}$ let $\chi(\overline{s})$ be the indicator random variable for some $s' \sim s$ being some $PYR(X_1 \cdots X_t)$. From Theorem 7.1

$$TLOG = \sum_{\overline{s} \in \overline{PYR}} \chi(\overline{s}) \tag{10}$$

Again we need to caution that $\chi(\overline{s})$ depends on the particular implementation of **FIXIT2a**. But we know that $REACH[\overline{s}]$ is necessary for $\chi(\overline{s}) = 1$. This yields the central result:

$$E[TLOG] \leq \sum_{\overline{s} \in \overline{PYR}} \Pr[REACH[\overline{s}]] \tag{11}$$

where $\Pr[REACH[\overline{s}]]$ is given by Theorem 9.

We define a statement `[KNUTH]` by

$$[\texttt{KNUTH}] \qquad \sum_{\overline{\texttt{s}} \in \overline{\texttt{PYR}}} \Pr[\texttt{REACH}[\overline{\texttt{s}}]] < \infty \tag{12}$$

Theorem 7.2 *If* `[KNUTH]` *then* `[SIEVE]`. *Moreover the expected number of times that* **FIXIT** *algorithm will call* **[FIXIT3]** *is at most*

$$\sum_{\overline{s} \in \overline{PYR}} \Pr[REACH[\overline{s}]]$$

8 Pyramids to Trees

OK, say we are given Σ and for each $\Xi_\alpha \in \Sigma$ the value $p_\alpha = \Pr[BAD_\alpha]$. Property `[KNUTH]` is now an analytic statement. There is a lot of beautiful analysis of `[KNUTH]` using algebraic combinatorics and some great generating functions, but here we'll just show the symmetric result Theorem 1.1.

We will associate to each $s = X_1 \cdots X_t \in PYR$ a rooted tree $T = TREE(s)$. For definiteness we fix some ordering of Σ. We place X_t as the root of the tree. Then for $u = t-1$ down to 1 we shall determine the parent of X_u. Suppose this has been done down to X_{u+1} and consider X_u. As s is a pyramid there will be r, $u < r \leq t$, with $X_u \cap X_r \neq \emptyset$. If there is only one such r we make X_u the child of X_r. But a critical step occurs when there is more than one such r. We select that r with $X_u \cap X_r \neq \emptyset$. which is *lowest*

on the rooted tree (that is, furthest from the root) as constructed thus far. For definiteness (the less important part) if there are still ties we select that X_r that is earliest in the ordering of Σ.

Suppose a letter Ξ appears twice (or more) in s. The appearances of Ξ in the tree must then appear in different levels. For if $\Xi = X_a = X_b$ with $a < b$ then the first criteria means that X_a will be placed at least as low as a child of X_b. We also deduce that the tiebreaker does break ties, we can't have X_u overlapping two copies of Ξ, both at the same level.

`Example:` With $s = ADCFEF$ as in our previous examples we have F as the root and define a parent function π working backwards. (We abuse notation slightly and write $\pi(\Xi) = \Xi'$, though Ξ, Ξ' refer to particular appearances in the string.) We set $\pi(E) = F$. Then $\pi(F) = E$ (since E is lower than F). Then $\pi(C) = E$, only choice. Then $\pi(D) = C$, at the same level as $F = X_4$ but we use our tiebreaker. Finally $\pi(A) = C$, only choice.

Theorem 8.1 *If $s \sim s'$ then $TREE[s] = TREE[s']$ Conversely, if $s, s' \in PYR$ and $TREE[s] = TREE[s']$ then $s \sim s'$.*

`Proof:` Suppose $s \sim s'$. We can reach s' from s by transposing nonintersecting letters so it suffices to consider the case where $s = a\Xi\Gamma b$, $s' = a\Gamma\Xi b$ $(a, b \in \Sigma^*)$ and Ξ, Γ not overlapping. But then the choices of the parents of Ξ and Γ do not interfere with each other.

Conversely, suppose $TREE[s] = TREE[s']$. For each $j \in \Omega$ consider the positions $X_{i_1}, \ldots, X_{i_r}$ where j appears in s, in the order that they appear in s. The earlier X's must appear lower in the tree. So the tree $TREE[s]$ determines the flip number for j for each of the $X_{i_1}, \ldots, X_{i_r}$. When $TREE[s] = TREE[s']$ the corresponding flip numbers are equal and so $s \sim s'$ by Theorem 5.1.

From Theorem 8.1 we can refer to $TREE[\overline{s}]$ for $\overline{s} \in \overline{PYR}$. Let TR denote the the family of finite rooted trees T with vertices labelled from Σ such that if Ξ is the child of Ξ' then $\Xi \cap \Xi' = \emptyset$. For $\Xi \in Sig$, let TR_Ξ denote thos $T \in TR$ with root Ξ.

For a tree T with nodes labelled $X_1 \cdots X_t$ we define its weight $w(T)$ as the product of the values $\Pr[BAD_i]$, where BAD_i is the bad event corresponding to X_i. Thus when $T = TREE[\overline{s}]$, $w(T) = \Pr[REACH[\overline{s}]$. We define a statement `[KNUTHWEAK]` by

$$[\texttt{KNUTHWEAK}] \qquad \sum_{T \in TR} w(T) < \infty \tag{13}$$

Theorem 8.2 *If* `[WEAKKNUTH]` *then* `[SIEVE]`.

This is immediate as the sum in (13) is at most the sum in (12). We have given ground here as not all trees $T \in TR$ are possible values of $TREE[s]$.

`Example:` Continuing our example, assumme lexicographical order on Σ. the pyramid $CAEC$ would generate a tree with root C having two children A, E and then the initial C being a child of A, assumming lexicographical order on Σ. Because of the tiebreaking the tree with root C having two children A, E and then C being a child of E would not appear as $TREE[s]$ for any string s.

9 The Symmetric Case

We now analyze [KNUTHWEAK] in the symmetric case. For any $\Xi \in \Sigma$ we define

$$y_\Xi = \sum_T w(T) \tag{14}$$

where the sum is over all $T \in TR$ with Ξ as a root. As Σ is finite it is necessary and sufficient that all y_Ξ be finite.

For convenience of exposition we will replace the first two assumptions of Theorem 1.1 with

1. All $\Pr[BAD_\alpha] = p$.
2. Each $\alpha \in I$ has $\alpha \sim \beta$ for precisely d indices $\beta \in I$

Now we claim that if all the y_Ξ are finite

$$y_\Xi = p \prod_{\Gamma \sim \Xi} (1 + y_\Gamma) \tag{15}$$

This follows from the recursive nature of trees. Every tree with root Ξ has a factor p corresponding to the root. For each $\Gamma \sim \Xi$ there is a factor of $1 + y_\Gamma$. If Γ is not a child of the root Ξ make the factor one. But when Γ is a child of the root Ξ we can have any tree with root Γ as the tree "beginning" from Γ and this would give y_Γ.

The existence of a finite system $y_\Xi \geq p$ satisfying (15) is thus a necessary condition for [KNUTHWEAK] but it turns out also to be sufficient. Suppose $y = y(p)$ satisfies

$$y = p(1 + y)^d \text{ and } y \geq p \tag{16}$$

We claim all $y_\Xi \leq p$. For any $i \geq 0$ let $y_{\Xi,i}$ denote the sum of $w(T)$ over $T \in TR$ with root Ξ and depth at most i. Then $y_{\Xi,0} = p$ tautologically. Assume by induction that all $y_{\Xi,i-1} \leq y$. Then (15) is modified to

$$y_{\Xi,i} = p \prod_{\Gamma \sim \Xi} (1 + y_{\Gamma,i-1}) \leq p(1 + y)^d = y \tag{17}$$

Finally, we are left with the Calculus problem, for which p, d is there a y satisfying (16). We reexpress it as $p = y(1 + y)^{-d}$. This function of y has a maximum at $y = \frac{1}{d-1}$, with $p = (d-1)^{d-1} d^{-d}$. For this or any smaller p there is a y by the Intermediate Value Theorem and therefore **FIXIT** works and so we have proven Theorem 1.1.

10 Potpourri

10.1 Algorithmic Considerations

If you aren't familiar with Depth First Search, skip this part!

A stumbling block in a naive application of **FIXIT** is at step **FIXIT2**. Checking all $\alpha \in I$ for BAD_α can be very inefficient. Lets suppose (other assumptions are interesting as well) that all $|\Xi| \leq a$, $\Xi \in \Sigma$, and that for each $\alpha \in I$ there are at most d $\beta \in I$ with $\alpha \sim \beta$, but that $|\Sigma| = n$

and consider the asymptotics with a, b fixed and $n \to \infty$. Suppose each BAD_α can be checked in time $O(1)$ and each reset **FIXIT3** takes time $O(1)$. Suppose, as is frequently the case, that the parameters are such that $E[TLOG] = O(n)$. Suppose further we are given a data structure consisting of the $\alpha \in I$, together with their Ξ_α, in some order and further for every $\alpha \in I$ a listing of those $\beta \in I$ with $\beta \sim \alpha$.

An efficient approach for **FIXIT** is then to use a Depth First Search for **FIXIT2**. In the outer loop we run through all $\alpha \in I$. The critical part is that starting at a given α we run Depth First Search. If BAD_α (elsewise we are done for that α) we reset Ξ_α and make a tree consisting of α and all its $\leq d$ children β. Now we go to the first such β and see if BAD_β. If not we delete β and move on. If yes we reset Ξ_β and add all $\gamma \sim \beta$ as children of β on the depth first search tree. At the end of the inner loop on α there would be no BAD_β with β coming before α in the ordering. Thus at the end of the entire procedure there would be no BAD_α whatsoever.

Suppose that during this inner loop or α there were $T1$ calls to **FIXIT3**. The depth first search tree would have at most $d \cdot T1$ vertices and so the actual time for this inner loop (thinking of d fixed) would only be $O(T1)$. Amortizing, the entire time for the procedure would be $O(TLOG)$ plus $O(n)$ times BAD_α was examined in the outer loop and it did not occur.

The end result: Under these moderate assumptions the actual time for **FIXIT** is *linear* in $|\Sigma|$ – linear in the data size!

10.2 A probabilty musing

LLL is a probability result and the original argument for it is a pure probability theorem. Here we have found what seems to be a very different approach. We prove a probability theorem by analyzing an algorithm. One has to wonder whether the various approaches now known for LLL are really all the same at some deeper level that has yet to be discovered.

10.3 A Prescient Adversary

The condition (12) implies that **FIXIT** has bounded $E[TLOG]$ for any implementation of the selection **[FIXIT2a]**. We can think of the selection **[FIXIT2a]** being made adversarially, it can even have randomized choice, but $E[TLOG]$ remains bounded. But we can say something even stronger. **FIXIT** wins against a *prescient* adversary. That is: suppose that the countable number of choices $COLOR[j,t]$ are preprocessed. Now suppose that our wily adversary sees *all* of the $COLOR[j,t]$ so that the adversary "knows the future," knows what the recolorings are going to be. Even under this assumption the number of $\overline{s}$ with $REACH\overline{s}$ has bounded expected value and regardless of the choices of the prescient adversary $TLOG$ cannot be greater than this number!

11 References

The original proof of LLL was in 1975 and arguments can be found in many books, including mine:

Noga Alon, Joel Spencer, The Probabilistic Method (Wiley)

The original result of Robin Moser, modified (and much improved!) by Gábor Tardos, is at

R. Moser, G. Tardos
A constructive proof of the general Lovász Local Lemma,
Journal of the ACM 57 (2010) (2) Art. 11, 15pp.
Available at arXiv:0903.0544.

The approach taken here is based on

K. Kolipaka, M. Szegedy
Moser and Tardos meet Lovász. STOC 2011: 235-244

enhanced by a work in progress of Donald Knuth.