Advanced Algorithms (V)

Yijia Chen
Fudan University
Review
Randomized Algorithms
Theorem

There is a randomized algorithm which computes the following selection problem

Input: A list $S$ of numbers and an integer $k$.
Output: The $k$th smallest element of $S$.

in expected linear time.

Corollary

There is a randomized algorithm which computes the median of a given list in expected linear time.
Probabilistic methods
The **probabilistic method** was invented by Paul Erdős.

*To prove an object with certain properties exists, we argue that an random object satisfies the desired properties with non-zero probability.*
Proper 2-coloring

Let $\ell \in \mathbb{N}$ and a set $S$. Moreover,

$$S_1, \ldots, S_m \subseteq S$$

with $|S_i| = \ell$ for all $i \in [m]$.

**Question**

*Can we color the elements of $S$ by 2 colors such that none of $S_i$’s is monochromatic?*

**Theorem**

*If $m < 2^{\ell-1}$, then there exists a proper 2-coloring defined as above.*
**Definition**
Let $A \subseteq \mathbb{N}$. Then $A$ is **sum-free**, if there do not exist $a_1, a_2, a_3 \in A$ with $a_1 + a_2 = a_3$.

**Theorem (Erdős, 1965)**
Let $B \subseteq \mathbb{N}$. Then there exists a sum-free $A \subseteq B$ with $|A| > \frac{|B|}{3}$. 

The Lovász Local Lemma
Let $A_1, \ldots, A_n$ be a set of bad events. Our goal is to establish

$$\Pr\left[ \bigcup_{i \in [n]} A_i \right] < 1,$$

i.e., it is possible that no bad event happens.
Two extreme cases

1. We have no assumption/prior knowledge about the (in)dependency of the events, then by the union bound

$$\Pr \left[ \bigcup_{i \in [n]} A_i \right] \leq \sum_{i \in [n]} \Pr[A_i].$$

Thus we need to guarantee that $\Pr[A_i] < 1/n$ for all $i \in [n]$.

2. The events are independent, then

$$\Pr \left[ \bigcup_{i \in [n]} A_i \right] = 1 - \prod_{i \in [n]} \Pr[\bar{A}_i].$$

Thus, we only need to ensure $\Pr[A_i] \neq 1$ for all $i \in [n]$. 
Independence

Definition
An event $A$ is independent of events $B_1, \ldots, B_n$ if for all nonempty subsets $I \subseteq [n]$ we have

$$\Pr \left[ A \cap \bigcap_{i \in I} B_i \right] = \Pr[A] \cdot \Pr \left[ \bigcap_{i \in I} B_i \right].$$

Definition
Let $A_1, \ldots, A_n$ be events. A directed graph $G = (V, E)$ with $V = [n]$ is a dependency digraph of $A_1, \ldots, A_n$ if, for every $i \in [n]$, the event $A_i$ is independent of the following set of events

$$\{ A_j \mid (i,j) \notin E \}.$$
Lemma

Let $A_1, \ldots, A_n$ be events. Moreover there are real numbers $x_1, \ldots, x_n$ such that $0 \leq x_i < 1$ and

$$\Pr[A_i] \leq x_i \cdot \prod_{(i,j) \in E} (1 - x_j)$$

for all $i \in [n]$. Then

$$\Pr \left[ \bigcap_{i \in [n]} \bar{A}_i \right] \geq \prod_{i \in [n]} (1 - x_i) > 0.$$
Proof (1)

We first prove by induction on the size of $S \subsetneq [n]$ that for any $i \in [n] \setminus S$

$$\Pr \left[ A_i \bigg| \bigcap_{j \in S} \bar{A}_j \right] \leq x_i.$$ 

This is trivial for $|S| = 0$. So we assume $|S| \geq 1$ and let

$$S_1 := \{ A_j \in S \mid (i,j) \in E \}.$$ 

and $S_2 := S \setminus S_1$. Then

$$\Pr \left[ A_i \bigg| \bigcap_{j \in S} \bar{A}_j \right] = \frac{\Pr \left[ A_i \cap \bigcap_{j \in S_1} \bar{A}_j \bigg| \bigcap_{j \in S_2} \bar{A}_j \right]}{\Pr \left[ \bigcap_{j \in S_1} \bar{A}_j \bigg| \bigcap_{j \in S_2} \bar{A}_j \right]}.$$ 

For the numerator

$$\Pr \left[ A_i \cap \bigcap_{j \in S_1} \bar{A}_j \bigg| \bigcap_{j \in S_2} \bar{A}_j \right] \leq \Pr \left[ A_i \bigg| \bigcap_{j \in S_2} \bar{A}_j \right] = \Pr[A_i] \leq x_i \cdot \prod_{(i,j) \in E} (1 - x_j).$$
To bound the denominator, we need the following equation.

$$\Pr \left[ \bigcap_{j \in [m]} B_m \right] = \prod_{j \in [m]} \Pr \left[ B_j \bigg| \bigcap_{k \in [j-1]} B_k \right].$$

Therefore, assume $S_1 = \{j_1, \ldots, j_r\}$ for some $r \geq 0$, then

$$\Pr \left[ \bigcap_{j \in S_1} \bar{A}_j \bigg| \bigcap_{j \in S_2} \bar{A}_j \right] = \prod_{k \in [r]} \Pr \left[ \bar{A}_{j_k} \bigg| \bigcap_{\ell \in [k-1]} \bar{A}_{j_\ell} \cap \bigcap_{j \in S_2} \bar{A}_j \right] = \prod_{k \in [r]} \left( 1 - \Pr \left[ A_{j_k} \bigg| \bigcap_{\ell \in [k-1]} \bar{A}_{j_\ell} \cap \bigcap_{j \in S_2} \bar{A}_j \right] \right) \geq \prod_{k \in [r]} (1 - x_{j_k}) \geq \prod_{(i,j) \in E} (1 - x_j)$$

by induction hypothesis. It follows $\Pr \left[ A_i \bigg| \bigcap_{j \in S} \bar{A}_j \right] \leq x_i$. 

Proof (2)
Finally

\[
\Pr \left[ \bigcap_{i \in [n]} \bar{A}_i \right] = \prod_{i \in [n]} \Pr \left[ \bar{A}_i \left| \bigcap_{j \in [i-1]} \bar{A}_j \right. \right] \geq \prod_{i \in [n]} (1 - x_i).
\]
Lovász Local Lemma (symmetric case)

Lemma (Lovász, 1977)
Let $A_1, \ldots, A_n$ be events with $\Pr[A_i] \leq p$ for all $i \in [n]$. If there is a dependency digraph of $A_1, \ldots, A_n$ with outdegree at most $d$, and $e \cdot p \cdot (d + 1) \leq 1$. Then

$$\Pr\left[\bigcap_{i \in [n]} \overline{A}_i\right] > 0.$$
Proof

Without loss of generality we can assume \( d \geq 2 \). Then we apply the general local lemma by setting

\[
x_i := \frac{1}{d + 1} < 1\]

for all \( i \in [n] \). Then

\[
x_i \cdot \prod_{(i,j) \in E} (1 - x_j) = \frac{1}{d + 1} \cdot \left(1 - \frac{1}{d + 1}\right)^d
\]

\[
= \frac{1}{d + 1} \cdot \frac{1}{(1 + 1/d)^d}
\]

\[
\geq \frac{1}{e \cdot (d + 1)} \geq p \geq \Pr[A_i].
\]

Thus, we conclude

\[
\Pr \left[ \bigcap_{i \in [n]} \bar{A}_i \right] \geq \prod_{i \in [n]} (1 - x_i) = \left(1 - \frac{1}{d + 1}\right)^n > 0.
\]
Theorem
Let $S_1, \ldots, S_m \subseteq S$ with $|S_i| = \ell$ for all $i \in [m]$. Moreover, every $S_i$ intersects at most $d$ other $S_j$’s. If $e \cdot (d + 1) \leq 2^{\ell-1}$, then there exists a proper 2-coloring of $S$. 
Proof

Again we color each element in $S$ by white and black independently and uniformly at random. Then for each $i \in [m]$

$$\Pr[A_i] := \Pr[S_i \text{ is monochromatic}] = \frac{1}{2^{\ell-1}}.$$ 

Observe that $A_i$ is independent of all $A_j$'s such that $S_i \cap S_j = \emptyset$. So we can construct a dependency digraph $G = ([m], E)$ such that there is an edge from $i$ to $j$ if and only if $S_i \cap S_j \neq \emptyset$. Then the outdegree of any vertex in $G$ is bounded by $d$.

The assumption for Lovász Local Lemma now translates to

$$e \cdot p \cdot (d + 1) = \frac{e \cdot (d + 1)}{2^{\ell-1}} \leq 1.$$ 

$\square$
Theorem

Given a conjunctive normal form formula such that each clause contains \( \ell \) variables and that each variable appears in at most \( k \) clauses. If

\[
e \cdot (\ell \cdot k + 1) \leq 2^{\ell - 1},
\]

then the formula is satisfiable.
The Moser-Tardos algorithm
Lovász Local Lemma only guarantees the existence of a proper 2-coloring, i.e.,

**Theorem**

Let \( S_1, \ldots, S_m \subseteq S \) with \( |S_i| = \ell \) for all \( i \in [m] \). Moreover, every \( S_i \) intersects at most \( d \) other \( S_j \)'s. If \( d + 1 \leq 2^{\ell-1}/e \), then there exists a proper 2-coloring of \( S \).

1. In 1975, Lovász and Erdős proved the existence for \( d + 1 \leq 2^{\ell-3} \).

2. In 1991, Beck gave an algorithm to find a proper two-coloring given that \( d + 1 \leq 2^{\ell/48} \).

3. In 2009, Moser gave an algorithm given that \( d + 1 \leq 2^{\ell-1}/((1+\varepsilon) \cdot e) \). It is later extended by Moser and Tardos to cover many more problems.
1. Randomly assign a color to each element in $S$ independently and uniformly.

2. While there is still a set $S_i$ that is monochromatic, randomly assign each element of $S_i$ independently and uniformly.
Expected running time

**Theorem**
Let $S_1, \ldots, S_m \subseteq S$ such that for all $i \in [m]$ we have $|S_i| = \ell$ and

$$\left| \{ j \in [m] \setminus \{i\} \mid S_j \cap S_i \neq \emptyset \} \right| \leq d.$$ 

Moreover

$$c \cdot e \cdot (d + 1) \leq 2^{\ell - 1}$$

for some constant $c > 1$. Then the Moser-Tardos algorithm finds a proper 2-coloring in expected time

$$(m + \ell)^{O(1)}.$$
Definition
A log of execution is the sequence

$$(1, S_{i_1}), (2, S_{i_2}), \ldots,$$

where $S_{ij}$ is the set which gets resampled at the $j$-th round of the algorithm.

Our goal is to show that long logs happen with low probability.
Definition
Let $j \in \mathbb{N}$. Then a witness tree for step $j$ is constructed as follows.

1. Start with a rooted tree with the only root $S_{ij}$.

2. For each $t = j - 1, j - 2, \ldots, 0$ if $S_{jt}$ intersects with at least one of the vertices in the current witness tree, then add $S_{it}$ to the tree by letting its parent be a vertex that intersects $S_{it}$ with maximum depth.

Observe that long logs are equivalent to high witness trees.
**Definition**
Let $\tau$ be a witness tree. Then the $\tau$-check procedure is defined as follows.

1. Visit vertices of $\tau$ in reversed BFS order (i.e., maximum depth first).
2. Assign a color to each vertex in the corresponding set as done in the algorithm.
3. Check whether the set is monochromatic.
4. If all the sets tested are monochromatic, then the check is passed. Otherwise, the check is failed.

**Lemma**

$$\Pr[\text{$\tau$-check passes}] = \left(\frac{1}{2^{\ell-1}}\right)^{|V(\tau)|}.$$
Lemma

Pr[\(\tau\) occurs as a witness tree] \(\leq\) Pr[\(\tau\)-check passes].
Lemma

Let $i \in [m]$. The number of the witness trees of size $s$ rooted at $S_i$ is bounded by

$$\binom{s \cdot (d + 1)}{s - 1}.$$
For every $j \in [m]$ let
\[ S_j^1, \ldots, S_j^{k_j} \]
be the sets that intersect with $S_j$ including $S_j$ itself. Note
\[ k_j \leq d + 1. \]

Let $\tau$ be a witness tree rooted in $S_i$ of size $s$, and we observe that
\begin{itemize}
  \item every $S_j$ in $\tau$ must have children among $\{ S_j^1, \ldots, S_j^{k_j} \}$;
  \item every child must be distinct.
\end{itemize}
Proof (2)

Let 

$$S_{\tau_1}, S_{\tau_2}, \ldots, S_{\tau_s}$$

be the sequence of $\tau$-vertices visited by BFS. If a parent $S_j$ has more than one child, always push the children in the order that they appear in $S_j^1, S_j^2, \ldots, S_j^{k_j}$.

Now, define a binary string $b_1 \ldots b_{s \cdot (d+1)}$ as follows.

$$b_{t \cdot (d+1) + r} := \begin{cases} 
1 & \text{if } r \leq k_{\tau_{t+1}} \text{ and } S^r_{\tau_{t+1}} \text{ is a child of } S_{\tau_{t+1}} \text{ in } \tau, \\
0 & \text{otherwise}
\end{cases}$$

all $t \in [0, s-1]$ and $r \in [d+1]$.

Clearly, this binary has exactly $s-1$ one’s. Furthermore, it is easy to see that no two witness trees get mapped to the same binary string.
Therefore, the number of the witness trees rooted at $S_i$ of size $s$ is bounded by the number of binary strings of length $s \cdot (d + 1)$ with exactly $s - 1$ one's, which is

\[
\binom{s \cdot (d + 1)}{s - 1}.
\]
Recall

**Theorem**

Let $S_1, \ldots, S_m \subseteq S$ such that for all $i \in [m]$ we have $|S_i| = \ell$ and

$$\left| \{ j \in [m] \setminus \{ i \} \mid S_j \cap S_i \neq \emptyset \} \right| \leq d.$$

Moreover

$$c \cdot e \cdot (d + 1) \leq 2^{\ell - 1}$$

for some constant $c > 1$. Then the Moser-Tardos algorithm finds a proper 2-coloring in expected time

$$O(m + \ell)^{O(1)}.$$
Clearly

the length of the log of execution

\[ = \sum_{\tau} \text{the number of times } \tau \text{ appears as a witness tree} \]

\[ = \sum_{\tau} X_\tau, \]

where

\[ X_\tau := \begin{cases} 
1 & \text{if } \tau \text{ appears as a witness tree} \\
0 & \text{otherwise.} 
\end{cases} \]

Here we use the simple fact that no \( \tau \) can appear twice.

Therefore

\[ \mathbb{E}[\text{the length of the log of execution}] = \mathbb{E} \left[ \sum_{\tau} X_\tau \right] = \sum_{\tau} \mathbb{E}[X_\tau]. \]
Then we deduce

$$\sum_{\tau} \mathbb{E}[X_{\tau}] = \sum_{\tau} \Pr[\tau \text{ appears as a witness tree}]$$

$$= \sum_{s=1}^{\infty} \sum_{i \in [m]} \sum_{|\tau|=s} \Pr[\tau \text{ appears as a witness tree}]_{\tau \text{ rooted at } S_i}$$

$$\leq \sum_{s=1}^{\infty} \sum_{i \in [m]} \binom{s \cdot (d + 1)}{s - 1} \cdot \left(\frac{1}{2^{\ell-1}}\right)^s$$

$$\leq m \cdot \sum_{s=1}^{\infty} \binom{s \cdot (d + 1)}{s - 1} \cdot \left(\frac{1}{2^{\ell-1}}\right)^s$$

$$\leq m \cdot \sum_{s=1}^{\infty} (e \cdot (d + 1))^s \cdot \left(\frac{1}{2^{\ell-1}}\right)^s$$

$$= m \cdot \sum_{s=1}^{\infty} \left(\frac{1}{c}\right)^s = O(m).$$
Satisfiability
An instance of the satisfiability problem:

- \( n \) Boolean variables, \( x_1, x_2, \ldots, x_n \).
- \( m \) constraints: each constraint is a clause, i.e., a logical OR of some variables or their negations.

The representation:

1. \( x_i \in \{0, 1\} \), 0 for ‘false’ and 1 for ‘true.’ The negation of \( x_i \) is \( \bar{x}_i \equiv 1 - x_i \). A variable or its negation is a literal, i.e., \( z_i \) denotes any of \( x_i, \bar{x}_i \).

2. Let \( a \in [m] \). Then the clause \( C_a \) on \( k_a \) variables forbids exactly one among the \( 2^{k_a} \) possible assignments to these \( k_a \) variables. It is written as a logical OR (denoted by \( \lor \)) function of some variables or their negations.

\( \partial C_a \) is the subset \( \{i_1^a, \ldots, i_{k_a}^a\} \subseteq [n] \) containing the indices of the \( k_a = |\partial C_a| \) variables in clause \( C_a \). Thus

\[
C_a = z_{i_1^a} \lor z_{i_2^a} \lor \cdots \lor z_{i_{k_a}^a}.
\]

An instance of the satisfiability problem in \textbf{conjunctive normal form (CNF)}:

\[
F = C_1 \land C_2 \land \cdots \land C_m.
\]
Given the formula $F$, the question is whether there exists an assignment of the variables $x_i$ to \{0, 1\} (among the $2^n$ possible assignments) such that the formula $F$ is true.

An algorithm that solves the satisfiability problem must be able, given a formula $F$, either to answer yes (the formula is then said to be SAT), and provide such an assignment, called a SAT-assignment, or to answer no, in which case the formula is said to be UNSAT.

Let $k \in \mathbb{N}$. If $k_a = k$ for all $a \in [n]$, then the satisfiability problem becomes the $K$-satisfiability (or $k$-SAT) problem.
The algorithm of Davis, Putnam, Logemann, and Loveland

DPLL (formula \( F \))

1. if \( F \) is empty then return the empty satisfying assignment
2. if \( F \) contains an empty clause then return UNSAT
3. Choose an index \( x_i \) in \( F \)
4. \( B \leftarrow \text{DPLL}(F[x_i \leftarrow 0]) \)
5. if \( B = \text{UNSAT} \) then \( B \leftarrow \text{DPLL}(F[x_i \leftarrow 1]) \)
6. else return \( \{x_i = 0\} \cup B \)
7. if \( B = \text{UNSAT} \) then return UNSAT
8. else return \( \{x_i = 1\} \cup B \).
Walk (CNF formula $F$ in $n$ variables)

1. for each variable $i$, set $x_i = 0$ or $x_i = 1$ with probability $1/2$
2. repeat $3 \cdot n$ times
3. if the current assignment satisfies $F$
4. then return it and stop
5. else
6. choose an unsatisfied clause $C_a$ uniformly at random
7. choose a variable $x_i$ uniformly at random in $\partial C_a$
8. flip the variable $x_i$ (i.e., $x_i \leftarrow 1 - x_i$)
9. end.
**Lemma**

Denote by $p(F)$ the probability that the routine Walk, when executed on a formula $F$ in $k$-CNF, returns a satisfying assignment. If $F$ is SAT, then $p(F) \geq p_n$, where

$$p_n = \frac{2}{3} \left( \frac{k}{2 \cdot (k - 1)} \right)^n.$$ 

If the routine is run $20/p_n$ times, the probability of not finding any solution is

$$(1 - p_n)^{20/p_n} \leq e^{-20}.$$