13.1 “Monte Carlo” Algorithms

This type of random algorithm produces random results that may or may not be correct. Such an algorithm is useful if the probability of producing a correct result is “large enough.” In this lecture, we will study two problems that are suitable for this type of algorithm: the min-cut problem and the problem of verifying polynomial identities.

13.2 Min-Cut Problem

**Input:** A connected, undirected graph $G = (V, E)$.
**Output:** A minimum cut in $G$, i.e., a minimal set of edges whose removal breaks $G$ into two or more components.

13.2.1 Simple Algorithm

A simple algorithm for the Min-Cut problem is to find a max-flow between every pair of vertices $(s, t)$ in the graph. For each pair of vertices, find the minimum $s$–$t$ cut of the corresponding max-flow. Then return the minimum of these $s$–$t$ cuts. Obviously this algorithm has a running time of $O(|V|^5)$ since finding the max-flow takes time $O(|V|^3)$ (using the Karzanov algorithm [K74]), and there are $O(|V|^2)$ pairs of vertices.

An improvement can be made by fixing a single source $s$. Since in the final min-cut, this particular source has to be on one side of the cut, it is sufficient to find the minimum $s$–$t$ cut for a single source. This way the algorithm runs in time $O(|V|^4)$.

13.2.2 Karger’s Algorithm

1. **while** the number of vertices $> 2$
2. select an edge $e = (u, v)$ uniformly random;
3. merge $u$ and $v$ into a single vertex, preserve all edges in the graph except those between $u$ and $v$;
4. **return** the remaining edges of the resulting graph.

Figure 13.1 shows an example of how this algorithm works. The input graph contains 10 edges labeled from $a$ to $k$. Suppose at first edge $h$ is picked, randomly, to be eliminated. The two vertices merge together into a single vertex. Now edge $e$ and $f$ have the same vertices, and so do edges $i$ and $j$. Next edge $c$ is chosen, and then edge $e$. Note that when merging the vertices of edge $e$, edge $f$ also disappears. This process continues until two vertices are left. At this point, the remaining edges are $b$ and $k$. 

13-1
Figure 13.1: Karger’s Algorithm

Luckily, these two edges indeed form a minimum cut of the original graph. In fact, the only two minimum cuts of this graph are \{a, c\} and \{b, k\}. If at any two steps, one edge from the \{a, c\} cut and one edge from the \{b, k\} cut are chosen to be eliminated, then the result of the Karger’s algorithm would be incorrect. Although we cannot prevent this from happening, we can guarantee a certain probability that a correct cut being returned by Karger’s algorithm, as seen in the following theorem:

**Theorem 13.1** Let \( C \) be a minimum cut of graph \( G = (V, E) \), then

\[
Pr(C \text{ returned by Karger's algorithm}) \geq \frac{2}{n^2},
\]

where \( n = |V| \)
\textbf{Proof:} Let $k$ be the number of edges in $C$, then every vertex in $G$ must have degree at least $k$. This is so because otherwise, if some vertex $v$ has degree less than $k$, then all of the edges of $v$ form a cut which is smaller than $C$.

Since there are $n$ vertices in $G$, there are at least $nk/2$ edges in $G$. Given this, we have

\[ Pr(\text{first edge chosen randomly } \in C) \leq \frac{k}{nk/2} = \frac{2}{n}. \]

Suppose at some step of the algorithm, there are $l$ vertices left. We call the graph at this step $G_l$. Since a cut in $G_l$ is necessarily a cut in $G$, the size of the minimum cut in $G_l$ is at least $k$. Following the same argument, there are at least $kl/2$ edges in $G_l$. We say $C$ is “hit” if some edge in $C$ is randomly chosen. Thus:

\[ Pr(C \text{ is hit when there are } l \text{ vertices left } | C \text{ is not hit before}) \leq \frac{k}{kl/2} = \frac{2}{l}. \]

Then

\[
Pr(C \text{ is returned}) = \prod_{l=3}^{n} Pr(C \text{ is not hit where there are } l \text{ vertices left } | C \text{ is not hit before}) \geq \left(1 - \frac{2}{n}\right) \left(1 - \frac{2}{n-1}\right) \left(1 - \frac{2}{n-2}\right) \cdots \left(1 - \frac{2}{4}\right) \left(1 - \frac{2}{3}\right) \\
= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdot \frac{n-5}{n-3} \cdots \frac{2}{4} \cdot \frac{1}{3} \\
= \frac{2 \cdot 1}{n(n-1)} \geq \frac{2}{n^2}
\]

\[ \blacksquare \]

### 13.2.3 Revised Karger’s Algorithm

We can simply repeat Karger’s algorithm $m$ times and return the smallest cut found. Given $\epsilon > 0$, we can ensure that the probability that this algorithm fails is less than $\epsilon$ by adjusting $m$ accordingly. In fact, if we let

\[ m = \frac{n^2}{2} \cdot \ln \frac{1}{\epsilon}, \]

then

\[ Pr(C \text{ is never returned}) \leq \left(1 - \frac{2}{n^2}\right)^m \leq \left(1 - \frac{2}{n^2}\right)^{\frac{2}{n^2} \cdot \ln \frac{1}{\epsilon}} \leq \left(1 - \frac{1}{\ln \frac{1}{\epsilon}}\right)^{\frac{1}{\ln \frac{1}{\epsilon}}} = \epsilon. \]

The second inequality is from that fact that

\[ \left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e}. \quad (13.1) \]
To get a sense of what this result means, consider $\epsilon = e^{-100}$, then we only need to make

$$m = \frac{n^2}{2} \ln e^{100} = 50n^2.$$

How small is this probability $\epsilon$? Well, the probability of a person getting hit by a meteor is about $e^{-\infty}$.

### 13.3 Verifying Polynomial Identities

**Input:** $n$-variable polynomial $Q(X_1, X_2, \ldots, X_n)$ of degree $d$, with a representation that is “easy” to evaluate at any point.

**Output:** The answer to the question “Is $Q(X_1, X_2, \ldots, X_n) \equiv 0$ ?”

The $\equiv$ sign means *equivalent*, i.e., no matter what values the variables take, the polynomial is always zero. For example,

$$Q = X_1^2X_2^3 + X_3 \neq 0,$$

while

$$Q = (X_1 + X_2)(X_1 - X_2) + (X_2 + X_3)(X_2 - X_3) + (X_3 + X_1)(X_3 - X_1) \equiv 0. \quad (13.2)$$

Polynomial 13.2 can be easily seen to not be equivalent to zero. To see that 13.3 is equivalent to zero, we can simply expand the polynomial and all the terms cancel each other out.

An equivalent problem to this is to test if two polynomials, possibly in different representations, are actually the same. We can simply test if the difference of the two polynomials is equivalent to zero.

In general, there is no known polynomial time deterministic algorithm for this problem. However, a Monte Carlo type of algorithm works well for this problem.

#### 13.3.1 Schwartz’s Algorithm

1. Let $S$ = a set of distinct real numbers
2. For $i = 1$ to $n$ do
   1. pick $z_i \in S$ uniformly random
3. if $Q(z_1, z_2, \ldots, z_n) = 0$ then output “YES”
4. else output “NO”

We observe that a “No” answer from Schwartz’s algorithm is always correct, since we know for sure there is a set of numbers that makes the polynomial not equal to zero. The “Yes” answer from the algorithm might be wrong. However, the following theorem states that the probability that it’s wrong can be bounded.

**Theorem 13.2** If $Q \equiv 0$, then the output of Schwartz’s algorithm is always correct. If $Q \not\equiv 0$, then the probability that the output is wrong is less than $d/|S|$, i.e.,

$$Pr\left(Q(z_1, z_2, \ldots, z_n) = 0\right) \leq \frac{d}{|S|}. \quad (13.4)$$

**Proof:** The first half of the theorem is obvious. The second half of the theorem is proved by induction on $n$. 
**Base case:** If \( n = 1 \), \( Q \) is a single variable polynomial of degree \( d \). Then there are at most \( d \) real numbers that makes \( Q \) evaluate to 0. Thus
\[
Pr\left(Q(z_1) = 0\right) \leq \frac{d}{|S|}.
\]

**Inductive step:** Suppose that for all polynomials with less than \( n \) variables, the claim of the theorem is true. Now consider a degree \( d \) polynomial with \( n \) variables, \( Q(X_1, X_2, \ldots, X_n) \). We can rewrite \( Q \) as a polynomial of \( X_1 \), (treating other variables as constants):
\[
Q(X_1, X_2, \ldots, X_n) = \sum_{i=0}^{k} Q_i(X_2, \ldots, X_n) X_1^i,
\]
where \( k \) is the maximum degree of \( X_1 \). Obviously \( Q_k(X_2, \ldots, X_n) \neq 0 \) because otherwise there will be no term of \( X_1 \) with degree \( k \). Also note that the degree of \( Q_k \) can be no larger than \( d - k \). Now pick \( z_2, z_3, \ldots, z_n \) randomly from \( S \). Since \( Q_k(X_2, \ldots, X_n) \) is a \( n - 1 \) variable polynomial, by induction,
\[
Pr\left(Q_k(z_2, \ldots, z_n) = 0\right) \leq \frac{d - k}{|S|}.
\]
On the other hand, if \( Q_k(z_2, \ldots, z_n) \neq 0 \), then
\[
Q(X_1, z_2, z_3, \ldots, z_n) = \sum_{i=0}^{k} Q_i(z_2, \ldots, z_n) X_1^i
\]
is a one variable \( (X_1) \) polynomial with degree \( k \). Thus given that \( Q_k(z_2, \ldots, z_n) \neq 0 \),
\[
Pr\left(Q(z_1, z_2, z_3, \ldots, z_n) = 0 \mid Q_k(z_2, \ldots, z_n) \neq 0\right) \leq \frac{k}{|S|},
\]
which is equivalent to saying that
\[
Pr\left(Q(z_1, z_2, z_3, \ldots, z_n) = 0 \mid Q_k(z_2, \ldots, z_n) \neq 0\right) \leq \frac{k}{|S|}.
\]
Thus,
\[
Pr\left(Q(z_1, \ldots, z_n) = 0\right) = Pr\left(Q(z_1, \ldots, z_n) = 0 \mid Q_k(z_2, \ldots, z_n) = 0\right) + Pr\left(Q(z_1, \ldots, z_n) = 0 \mid Q_k(z_2, \ldots, z_n) \neq 0\right) \leq \frac{k}{|S|} + \frac{d - k}{|S|} = \frac{d}{|S|}.
\]

### 13.3.2 Discussion

If we set \(|S| = 2d\), then the probability that Schwartz’s algorithm returns the wrong answer is less than 1/2, given that \( Q \neq 0 \). If we repeat the algorithm for \( \log_2(1/e) \) times, then the probability that algorithm returns the wrong answer all the time is less than:
\[
\left(\frac{1}{2}\right)^{\log_2(\frac{1}{e})} = e.
\]
13.3.3 Boolean Satisfiability

**Input:** Boolean formula $\phi(X_1, X_2, \ldots, X_n)$.

**Question:** Is $\phi \equiv FALSE$?

This problem looks similar to the polynomial identity verification problem, but actually it’s not. A random algorithm can not solve this problem with high enough probability of success within a reasonable time period. For example, let $\phi = X_1 \land X_2 \land \cdots \land X_n$. We know that $\phi \not\equiv FALSE$. However,

$$Pr\left(\phi(z_1, z_2, \ldots, z_n) = TRUE\right) = \frac{1}{2^n},$$

thus,

$$Pr\left(\phi(z_1, z_2, \ldots, z_n) = FALSE\right) = 1 - \frac{1}{2^n}.$$

That is, if $\phi \not\equiv FALSE$, the probability that the algorithm gives a wrong answer is $1 - 1/2^n$. We will need to run the algorithm an exponential number of times to bring down the error probability.

References