I. Vector Program Relaxation for Max Cut problem

We can write the Max Cut problem as:

\[
\text{maximize } \sum_{i,j \in E} w_{ij}^{1/2}(1-v_i^Tv_j) \quad (1)
\]

subject to \(\|v_i\|^2 = 1\)

\[v_i \in \mathbb{R}^n\]

Consider the vectors are in a sphere

We are going to take a random unit vector \(r \in \mathbb{R}^n\). Set \(y_i = +1\) if \((v_i \cdot r \geq 0)\)

As the inner product of two vectors is equal \(v_i^Tv_j = \cos(\theta_{ij})\) and \(0 \leq \theta \leq \pi\). Then substituting in equation (1), we can re-write the problem as:

\[
\text{maximize } \sum_{i,j \in E} w_{ij}^{1/2}(1-\cos(\theta_{ij}))
\]

Where 
- \(\cos(\pi) = -1\)
- \(\cos(\pi/2) = 0\)
- \(\cos(0) = 1\)

Claim: The expected value of the rounded solution will be

\[
\mathbb{E}[\text{rounded solution}] \geq \alpha \sum_{i,j \in E} w_{ij}^{1/2}(1-\cos(\theta_{ij}))
\]
Main Claim: Prob, [v_i and v_j are separated by r] = \theta_{ij}/\pi (Proof by picture above *)

To get the desired expectation we want \theta_{ij}/\pi \geq \alpha \left(1 - \cos(\theta_{ij}) \right)/2 \\
\alpha = \min_{\theta \in [0, \pi]} 2/\pi \left(1 - \cos(\theta)/\theta \right) \geq 0.87856

II. Vector Program Relaxation for Max-2-Sat problem

y_0 \ldots y_i \in \{+1, -1\} \\
y_i \sim \text{propositional } x_i

Interpretation: x_i = \{true if y_i = y_0 \\
false if y_i = -y_0\}

We are trying to maximize:

maximize \tau \sum C \text{val}(C)

val(C) \rightarrow \{1 \text{ if } C/\tau = \text{true} \\
0 \text{ if } C/\tau = \text{false}\}

If C = x_i then val(C) = (1 + y_i y_0)/2

If C = \bar{x}_i then val(C) = 1 - (y_i y_0)/2

And if C = x_i v x_j then

val(C) = 1 - val(\bar{x}_i) \ast val(\bar{x}_j) \\
= a(1 + y_i y_0)/2 + b(1 - y_i y_0)/2

maximize \sum_{i,j} a_{ij}(1 + y_i y_j) + b_{ij}(1 - y_i y_j)

Relaxing \ yi y_j to v_i \cdot v_j, with v_i \in \mathbb{R}^{n+1}, \|v_i\| = 1, \ 0 \leq i \leq n.

Doing the calculations we get the same \alpha \sim 0.87856 approximation.

III. The constructive Lovasz Local Lemma (LLL)

Having a series of events E_1, \ldots, E_m

Prob [E_i] = p < 1

Prob [\bar{E}_1 \wedge \bar{E}_2 \wedge \ldots \wedge \bar{E}_m] > 0, if all E_i are independent

Suppose that each E_i occurs with probability at most p, and such that each event is independent of all the other events except for at most d of them.
If

\[ ep(d + 1) \leq 1 \]

(Where \( e = 2.718... \) is the base of natural logarithms), then there is a nonzero probability that none of the events occurs.

Consider an instance of the LLL,

\[ F = C_1 \lor C_2 \lor \ldots \lor C_m \]

\( F \) in an exactly \( K-CNF \),

\( E_i = C_i \) is not satisfied by a random \( \tau \), then

\[ \text{Prob}[E_i] = 1/2^k \]

\( \bar{E}_i \) means \( C_i \) is satisfied, then

\[ \text{Prob}[\bar{E}_1 \land \bar{E}_2 \land \ldots \land \bar{E}_m] = \text{Prob}[F/\tau \text{ is satisfied}] > 0 \]

Let \( d \) be the number of clauses that share a variable (with a given clause \( C \)), then to apply the LLL we want:

\[ d + 1 \leq 2^k / e \]

which implies that \( d < 2^k / e \).

The Constructive Proof:

(Proof by Moser and Gabor, Tardos + Moser)

Chose any random \( \tau \)

Solve,

\[ \text{while there exists a clause } C \text{ not satisfied by } \tau \]

\[ \text{Call Fix}(C) \]

end while

Fix(\( C \))

Randomly set the bits in \( C \)

While there is a neighboring clause \( D \) that is unsatisfied

\[ \text{Call Fix}(D) \]

End while

Theorem: Let \( r \) be the size of neighborhood. If \( r \) is not too big: \( r \leq 2^k/8 \), then with high probability the algorithm terminates in \( O(m \log m) \) calls to Fix(\( C \)) and hence found a satisfying assignment.

Proof:

Suppose the algorithm takes at least \( s \) recursive calls to fix,
n + s*k bits describes algorithm up to the s\textsuperscript{th} cell at which time we have some true assignment \(\tau'\).

Using Kolmogorov complexity, we state the fact that most random strings cannot be compressed.

Now we say that if \(r\) is sufficiently small \(k - \log v - c > 0\)

Then we can describe these \(n + s*k\) bits in a compressed way.

\(n\) bits to describe \(\tau'\).
\(s\) calls to fix
\(C_1 \ldots C_s\) clauses being fixed.

**Claim:** Solve has at most \(m\) clauses.

Any \(C'\) satisfied before \(\text{Fix}(C)\) that is called in \(\text{Solve}\) remains satisfied.

**Claim:** We can recover original \(n + s*k\) bits using

\[
\begin{align*}
n + m \log m + s (\log r + c) & \geq n + s*k \\
\text{(for } \tau' \text{) (calls to fix in solve)}
\end{align*}
\]

\(m \log m \geq s \left( k - \log r - c \right)\)

(Note: Here it is not proved, but the algorithm does not always terminates)

### IV. Primality testing

Some background in primality tests and authors:

- **The Solovay–Strassen primality test**, developed by Robert M. Solovay and Volker Strassen, is a probabilistic test to determine if a number is composite or probably prime. It has been largely superseded by the Miller–Rabin primality test, but has great historical importance in showing the practical feasibility of the RSA cryptosystem.

- **The Miller–Rabin primality test** or **Rabin–Miller primality test** is in an algorithm which determines whether a given number is prime, similar to the Fermat primality test and the Solovay–Strassen primality test. Its original version, due to Gary L. Miller, is deterministic, but the determinism relies on the unproven generalized Riemann hypothesis; Michael O. Rabin modified it to obtain an unconditional probabilistic algorithm.

In general those authors gave a one sided randomized algorithm.

\[
\begin{align*}
\text{Prob[ ALG says } N \text{ prime } | N \text{ composite }] & \leq \frac{1}{2} \\
\text{Prob } [\text{ALG says } N \text{ composite } | N \text{ prime}] & = 0 \\
\Rightarrow \text{ Composite testing } \in \text{ RP}
\end{align*}
\]
Except for a very small (but still infinite) class of numbers, there is a very simple randomized algorithm:

> Fermat’s little theorem: $N$ prime implies $a^{N-1} = 1 \mod N$, where $\gcd(a, N) = 1$

**Lagrange’s Theorem:** If $S$ is a subgroup of a group $G$, then $|S|$ divides $|G|$.

Our group: $\mathbb{Z}_N^* = \{a \mid \gcd(a, N) = 1\}$, under mult’ mod $N$

$S = \{a \mid \gcd(a, N) = 1 \text{ and } a^{n-1} \mod N = 1, \ 1 \leq a \leq n-1 \}$ is a subgroup.

*False test:* choose $a \in \mathbb{Z}_N^*$ randomly.

  If $a^{k-1} \mod N = 1$ -> Output prime

  Else -> Output composite

**Carmichael number (AKA false primes):**

For all $a \in \mathbb{Z}_N^*$, $a^{k-1} \mod N = 1$, and $N$ is composite.

$N$- Carmichael -> $N=N_1N_2N_3$, $N_i$ square free. Gcd $(N_i, N_j) = 1$

- If $N$ is prime, then $\mathbb{Z}_N^*$ is a field, and 1 has exactly 2 square roots $\{-1, +1\}$
- If $N$ is odd then $N-1$ is even -> $N-1 = 2^u$, with $u$ odd

**Algorithm:**

Choose $a \in \mathbb{Z}_N^*$ randomly,

$x_0 = a^u \mod N$, $x_i = a^{N-1} \mod N$

for $i=1$ until $t$

$x_i = x_i^{2^i} \mod N$

if $x_i$ does not belong to $\{-1, +1\}$ then output composite.

end for

if $x_i \neq 1$ then output composite

else output prime.

**Chinese remainder theorem (CRT):**

$\gcd(N_1N_2) = 1$ then for all $u, v, \ N = N_1N_2$

Exist $\lambda$: $\lambda = u \mod N_1$
\[ \lambda = \nu \mod N_2 \text{ and } 0 \leq \lambda < N_1 N_2. \]

V. Multiplicative update

Suppose we have \( n \) experts who every day are predicting the value of a \( \{0, 1\} \) event.

Let \( m_i^t \) be the number of errors by experts \( i \) in first \( t \) days. We want an algorithm that will do well compared to the best expert.

\( w_i \) will be the weight of \( i^{th} \) expert. Initially \( w_0^i = 1 \).

Algorithm:

For \( j=1 \) until \( t \) (for each day)

For each \( i \)

if prediction of expert \( i \) on day \( j \) is wrong then \( w_j^i = w_{j-1}^i (i- \varepsilon) \)

else \( w_j^i = w_{j-1}^i \)

end for

end for

Output: the weighted majority of the expert predictions.

Claim: Let \( m_i^t \) be the number of errors the algorithm make. Then \( m_i^t \leq 2 \ln n / \varepsilon + 2 (1 + \varepsilon) m_i^t \), for all \( i \).