The Max-Sat Problem:

Input:

- CNF Formula: $\Psi = C_1 \land C_2 \land \ldots \land C_m$
- $C_1, C_2, \ldots, C_m$ clauses, each clause is a sum of Literals
- $C_j = (l_{i_1} \lor l_{i_2} \lor \ldots \lor l_{i_n})$
- Weight $w_j$ for each clause $C_j$

Objective: Find the substitution of the truth values for variables, such that the number of true clauses is maximized.
ILP Formulation:

- For every clause $C_j$ we will create a variable $z_j$ that can have a value of zero or 1.
- To every variable $x_i$ we will match a variable $y_i$ that can be zero or 1.
- Let $P_j$ denote the set of variables appearing in $C_j$ without negation and $N_j$ denote the set of variables appearing in $C_j$ with negation.

$$ W = \max_{1 \leq j \leq m} \sum_{l \in P_j} w_j z_j $$

$$ \text{ST} \sum_{l \in P_j} y_i + \sum_{x_i \in N_j} (1 - y_i) \geq z_j, \forall C_j $$

$$ \forall 1 \leq i \leq n, y_i \in \{0,1\} $$

$$ \forall 1 \leq j \leq m, z_j \in \{0,1\} $$

LP Relaxation:

$$ W = \max_{1 \leq j \leq m} \sum_{l \in P_j} w_j z_j $$

$$ \text{ST} \sum_{l \in P_j} y_i + \sum_{x_i \in N_j} (1 - y_i) \geq z_j, \forall C_j $$

$$ \forall 1 \leq i \leq n, 0 \leq y_i \leq 1 $$

$$ \forall 1 \leq j \leq m, 0 \leq z_j \leq 1 $$
Simplest possible algorithm!

- Set each variable to be true randomly and independently; i.e. $x_i$ to true with probability $\frac{1}{2}$ independently.

**Johnson 74**: The above randomized algorithm is a $\frac{1}{2}$-approximation algorithm:

Consider the random variables $z_j$ and objective function $W$ defined as before, then by linearity of expectation:

$$E[W] = \sum_{j=1}^{m} w_j E[z_j] = \sum_{j=1}^{m} w_j \Pr[\text{clause } j \text{ is satisfied}]$$

$$= \sum_{j=1}^{m} w_j \left( 1 - \left( \frac{1}{2} \right)^k \right) \geq \frac{1}{2} \sum_{j=1}^{m} w_j \geq \frac{1}{2} OPT$$

---

De-randomization:

- We can make the algorithm deterministic using a method known as Conditional Expectation. In the simplest terms, we will set the values of $x_2, \ldots, x_n$ randomly, but set $x_1$ in the best possible way; assume $x_1$ both true and false, and find the expected value of solution:

  if $E[W \mid x_1 \leftarrow True] \geq E[W \mid x_1 \leftarrow False]$ then
  
  Set $x_1 \leftarrow True$

  Else
  
  Set $x_1 \leftarrow False$
Observe that:

- By definition of conditional expectation:
  \[ E[W] = E[W | x_1 \leftarrow True].Pr[x_1 \leftarrow True] + E[W | x_1 \leftarrow False].Pr[x_1 \leftarrow False] \]
  \[ = \frac{1}{2}(E[W | x_1 \leftarrow True] + E[W | x_1 \leftarrow False]) \]

- If \( x_i \) is set to \( b_i \) as above, then the overall expectation is
  \[ E[W | x_1 \leftarrow b_1] \geq E[W] \geq \frac{1}{2}OPT \]

De-randomization:

- In general, suppose that we have already set the values for \( x_1, \ldots, x_i \) with above procedure. We can use the following step to set the value \( x_{i+1} \)
  
  if \[ E[W | x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow True] \]
  \[ \geq E[W | x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow False] \]
  then
    Set \( x_{i+1} \leftarrow True \)
  Else
    Set \( x_{i+1} \leftarrow False \)
De-randomization:

- The computations work as before:
  \[ E[W \mid x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i] = \]
  \[ E[W \mid x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow True]. \Pr[x_{i+1} \leftarrow True] \]
  + \[ E[W \mid x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow false]. \Pr[x_{i+1} \leftarrow false] \]
  Setting \( x_{i+1} \) as before:
  \[ E[W \mid x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow b_{i+1} ] \]
  \[ \geq E[W \mid x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i ] \]
  \[ \geq E[W] \]
  \[ \geq \frac{1}{2} OPT \]

Algorithm:

For i = 1 to n
  \[ W_T = E[W \mid x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_{i-1} \leftarrow b_{i-1}, x_i \leftarrow True] \]
  \[ W_F = E[W \mid x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_{i-1} \leftarrow b_{i-1}, x_{i+1} \leftarrow False] \]
  If \( W_T \geq W_F \) then
    Set \( x_i \leftarrow True \)
  Else
    Set \( x_i \leftarrow False \)
Primal-dual Algorithm:

- How do we calculate:

\[
E[W \mid x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i]
\]

- By linearity of expectation, we know that

\[
E[W \mid x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i] = \sum_j w_j E[Y_j \mid x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i]
\]

- Furthermore:

\[
E[Y_j \mid x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i] = \Pr[\text{clause } j \text{ is satisfied} \mid x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i]
\]

De-randomization:

- It is not hard to see that:

\[
\Pr[\text{clause } j \text{ is satisfied} \mid x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i] =
\begin{cases}
1 & \text{if } x_1 \leftarrow b_1, x_2 \leftarrow b_2, \ldots, x_i \leftarrow b_i \text{ already satisfies } C_j \\
1 - \left(\frac{1}{2}\right)^\alpha & \text{Otherwise, where } \alpha \text{ of } x_{i+1}, \ldots, x_n \text{ in } C_j
\end{cases}
\]
Example:

Consider for example \( x_5 \lor \overline{x}_5 \lor \overline{x}_7 \lor x_{11} \). It is not hard to see that

\[
\text{Pr}[\text{clause satisfied } | \ x_1 \leftarrow T, x_2 \leftarrow F, x_3 \leftarrow T, x_4 \leftarrow F ] = 1
\]

since \( x_3 \leftarrow T \) satisfies the clause. On the other hand,

\[
\text{Pr}[\text{clause satisfied } | \ x_1 \leftarrow T, x_2 \leftarrow F, x_3 \leftarrow F, x_4 \leftarrow F ] = 1 - \left( \frac{1}{2} \right)^3 = \frac{7}{8},
\]

Since only the bad setting of \( x_5, x_7, x_{11} \) will make the clause unsatisfied.

Flipping Bend Coin (Johnson):

- We will bias the probabilities for each Boolean variable. Let us set \( x_i \) to true with probability \( p \) (at least \( \frac{1}{2} \), the exact value to be determined). If \( k \), the length of a clause, is 1 then:

  \[
  \text{Pr}[\text{clause } j \text{ is satisfied}] = p
  \]

- If \( k \) is at least 2, then

  \[
  \text{Pr}[\text{C}_j \text{ is satisfied}] = 1 - p^a(1 - p)^b \geq 1 - p^{a+b} \geq 1 - p^2
  \]

  Where \( a \) and \( b \) number of negated and un-negated variables, respectively. Consequently:

  \[
  \text{Pr}[\text{clause } C_j \text{ is satisfied}] \geq \min( p, 1-p^2 )
  \]

- We set \( p = 1 - p^2 \) that means \( p = 0.618 \)
In fact:

- Flipping biased coins as above is a $p$-approximation algorithm for MAX-SAT:

$$E[W] = \sum_{j} w_j \Pr[C_j \text{ is satisfied}] \geq p \sum_{j} w_j \geq p.OPT$$

Randomized Rounding (Goemans-Williamson)

- Let us consider what will happen if we tried to give different biases to determine each $x_i$. Let us look back at linear programming relaxation of Max SAT. With variables $z_j$ for each clause $C_j$ and indicator variables $y_i$ for literal $x_i$, defined as follows:

$$z_j = \begin{cases} 1 & \text{if clause } j \text{ is satisfied} \\ 0 & \text{otherwise} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{if } x_i \leftarrow \text{true} \\ 0 & \text{otherwise} \end{cases}$$
LP Relaxation:

\[
W = \max \sum_{1 \leq j \leq m} w_j z_j \\
\text{ST} \sum_{l \in P_j} y_i + \sum_{x \in N_j} (1 - y_i) \geq z_j, \forall C_j \\
\forall 1 \leq i \leq n, 0 \leq y_i \leq 1 \\
\forall 1 \leq j \leq m, 0 \leq z_j \leq 1
\]

Rounding:

- Solve the LP relaxation and find the solution \( y^* = (y_1^*, ..., y_n^*) \).
- Set \( x_i \leftarrow \text{True} \) with probability \( y_i^* \), independently for all \( i = 1, ..., n \).

To prove the performance of this algorithm, we will the following:

- for any non-negative sequence \( a_1, a_2, ..., a_k \), we have
  \[
  \frac{1}{\sqrt[k]{a_1 a_2 ... a_k}} \leq \frac{1}{k} (a_1 + a_2 + ... + a_k)
  \]
- If \( f(x) \) is concave on \([l, u]\), and \( f(l) \geq al + b \), and \( f(u) \geq au + b \), then the function is lower-bounded by a line through the end-points at \([l, u]\), i.e. \( f(x) \geq ax + b \), on \([l, u]\)
Rounding:

Theorem: Randomized rounding is a \( \left( \frac{1}{e} \right) \)-approximation algorithm where \( \left( \frac{1}{e} \right) = 0.632 \).

Proof: Consider an arbitrary clause \( C_j \):

\[
\Pr[\text{clause } C_j \text{ is not satisfied}] = \prod_{i \in P_j} (1 - y_i^*) \prod_{i \in N_j} y_i^* \\
\leq \left[ \frac{1}{k} \left( \sum_{i \in P_j} (1 - y_i^*) + \sum_{i \in N_j} y_i^* \right) \right]^k \\
= \left[ 1 - \frac{1}{k} \left( \sum_{i \in N_j} (1 - y_i^*) + \sum_{i \in P_j} y_i^* \right) \right]^k \\
\leq \left( 1 - \frac{z_j}{k} \right)^k
\]

Rounding:

Consider the probability that \( C_j \) is satisfied: \( \Pr[\text{clause } C_j \text{ is satisfied}] = 1 - \Pr[\text{clause } C_j \text{ is not satisfied}] \geq 1 - \left( 1 - \frac{z_j}{k} \right)^k \).

Observe that, \( z_j = 0 \Rightarrow 1 - \left( 1 - \frac{z_j}{k} \right)^k = 0 \) and

\[
z_j = 1 \Rightarrow 1 - \left( 1 - \frac{z_j}{k} \right)^k = 1 - \left( 1 - \frac{1}{k} \right)^k
\]

and \( 1 - \left( 1 - \frac{z_j}{k} \right)^k \) is concave in terms of \( z_j \).

Therefore, \( \Pr[\text{clause } C_j \text{ is satisfied}] \geq 1 - \left( 1 - \frac{1}{k} \right)^k z_j \).
Therefore:

\[ E[W] = \sum_j w_j \Pr[\text{clause } C_j \text{ is satisfied}] \]

\[ \geq \min_k \left[ 1 - \left( 1 - \frac{1}{k} \right)^k \right] \sum_j w_j z_j \]

\[ \geq \min_k \left[ 1 - \left( 1 - \frac{1}{k} \right)^k \right] OPT \]

\[ \geq \left( 1 - \frac{1}{e} \right) OPT \approx 0.632 OPT. \]

Improvement:

Run Johnson's algorithm, get assign \( x^1 \) of weight \( W^1 \)
Run G - W's algorithm, get assign \( x^2 \) of weight \( W^2 \)
If \( W^1 \geq W^2 \)
    Return \( x^1 \)
Else
    Return \( x^2 \)
Theorem [G-W]:

- Best-of-Two is a \( \frac{3}{4} \)-approximation algorithm for Max-SAT.

\[
E[\max(W^1, W^2)] \geq E\left[\frac{1}{2}W^1 + \frac{1}{2}W^2\right] = \sum_j w_j \left(\Pr[C_j = \text{true}(\text{under Johnson})] + \Pr[C_j = \text{true}(\text{under GW})]\right)
\]

\[
\geq \sum_j w_j \left[\frac{1}{2} \left(1 - \left(\frac{1}{2}\right)^k\right) + \frac{1}{2} \left(1 - \left(\frac{1}{k}\right)^k\right)\right] z_j
\]

\[
\geq \frac{3}{4} \sum_j w_j z_j
\]

\[
\geq \frac{3}{4} \text{OPT}
\]

Theorem [G-W]:

- We need to show to prove:

\[
\frac{1}{2} \left(1 - \left(\frac{1}{2}\right)^k\right) + \frac{1}{2} \left(1 - \left(\frac{1}{k}\right)^k\right) z_j \geq \frac{3}{4} z_j
\]

- The cases \( k = 1, 2 \) are easy. For \( k > 2 \), we take minimum possible value of the two terms:

\[
\frac{1}{2} \times \frac{7}{8} + \frac{1}{2} \left(1 - \frac{1}{e}\right) z_j \geq \frac{3}{4} z_j
\]