3SAT: Satisfiability with at Most Three Literals per Clause

Nagasai Chandra

May 2020

1 Problem Definition

The satisfiability problem is the problem of determining if there exists an interpretation that satisfies a given Boolean formula. It asks whether the variables of a given Boolean formula can be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE. It is the first problem that was proven to be *NP-complete*.

Satisfiability (SAT)
Input: A Boolean formula C comprised of a set of n variables $x_1, \ldots, x_n$, in clauses $C_1, C_2, \ldots C_p$ where each term $t_i \in x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}$, subject to the constrain that a variable and its negation may not be present together in the same clause.
Goal: Determine if the variables in the formula can be assigned in such a way as to make the formula evaluate to TRUE, or determine that no such assignment exists.

3-SAT is a special case of k-satisfiability problem, where each clause contains exactly $k = 3$ literals. 3-SAT problem is one of Karp's 21 NP-complete problems[2].

3-Satisfiability (3SAT)
Input: A Boolean formula C in conjunctive normal form comprised of a set of n variables $x_1, \ldots, x_n$, where each clause contains exactly $k = 3$ literals/terms, where each term $t_i \in x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}$, subject to the constrain that a variable and its negation may not be present together in the same clause.
Goal: Return an assignment S, if it exists, such that at least one literal in every clause is True, or return False otherwise, i.e., if there is an assignment

$$\text{S: } \mathrm{x}_1, x_2, \ldots, x_n \rightarrow 0, 1$$

such that every clause c_i is satisfied or return False.

The above statement of 3SAT is the optimization version of the problem. We can re-frame the problem as a decision problem:

> **3-Satisfiability (3SAT)**
> **Input:** A Boolean formula C in conjunctive normal form, comprised of a set of n variables $x_1, \ldots, x_n$, where each clause contains exactly $k = 3$ literals/terms, where each term $t_i \in x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}$, subject to the constrain that a variable and its negation may not be present together in the same clause, and a positive integer p.
> **Goal:** Return an assignment S, that can satisfy at least p number of clauses $Q \in C$, such that at least one literal in every clause of Q is True, or return False otherwise, i.e., if there is an assignment
>
> $$\text{S: x}_1, x_2, \ldots, x_n \rightarrow 0, 1$$
>
> such that every clause $q_j \in Q$ is satisfied or return False.

We will now check if there is a checking algorithm that can check the correctness of decision version of 3SAT problem in polynomial time, i.e., if 3SAT $\in$ NP.

2 3SAT $\in$ NP

We need to design an algorithm that takes as input an instance of 3SAT I, which is a boolean expression C, and a solution S, an assignment of all variables in C and checks, in polynomial time, whether S is a correct solution for I or not.

Check3SAT$(I, p), S)$

Input: An instance of 3SAT, I, and a positive integer p, where I is a boolean expression $C = c_1 \wedge c_2 \wedge \cdots \wedge c_n$ and where every clause $c_i \in C$ has exactly three literals.
Output: True if the assignment S: $x_1, x_2, \ldots, x_n \rightarrow 0, 1$ is used on C at least one literal in at least p clauses of C is satisfied, False if it does not.

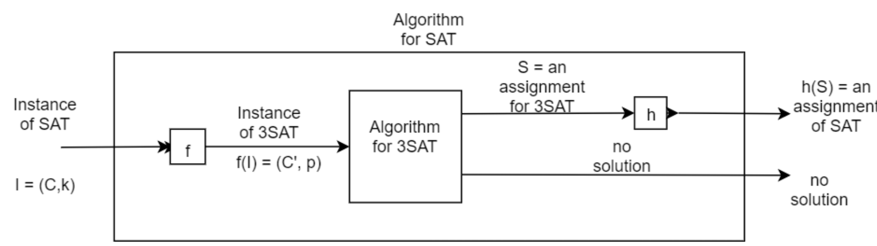
```
for all x_i ∈ X do
    assign values 0,1 to x_i according to S
    if at least one literal in at least p clauses are True then
        return True
return False
```

We have given an algorithm [2], Check3SAT, with running time $O(|C|)$ (definitely polynomial), that correctly checks if a proposed solution to 3SAT is in fact a solution. Thus 3SAT $\in$ NP.

3 3SAT $\in$ NP-Complete

Here we give a reduction to show that SAT is NP-Complete.

3.1 SAT $\leq_p$ 3SAT



An instance of SAT is a boolean expression $C = c_1 \wedge c_2 \wedge \cdots \wedge c_m$ and a positive integer k. We need to transform this into an instance for 3SAT: a boolean expression $Z = z'_1 \wedge z'_2 \wedge \cdots \wedge z'_n$ and a positive integer p.

- $f((C, k)) = (Z, p)$ where Z denotes an instance of 3SAT where each clause z_i has exactly 3 literals.

Now, suppose we have an assignment, S, to 3SAT for (Z, p), then we will return the assignment for C i.e., the assignment for SAT is the same as the assignment for 3SAT.

- $h(S) = S$

The function $f(I)$ transforms instance I, in polynomial time, in the following manner: for a clause $c_i \in C$:

- Case 1: If c_i contains three literals, no reduction is needed,i.e., $z'_i = c_i$.
- Case 2: If c_i contains only one literal l_i, introduce 2 new variables $y_{i,1}$ and $y_{i,2}$ and replace c_i with conjunction of clauses z_i, where,

$$z_i = (l_i + y_{i,1} + y_{i,2}) \wedge l_i + \overline{y_{i,1}} + y_{i,2}) \wedge l_i + y_{i,1} + \overline{y_{i,2}}) \wedge l_i + \overline{y_{i,1}} + \overline{y_{i,2})}$$

- Case 3: If $c_i = (l_{i,1} + l_{i,2})$ contains two literals $l_{i,1}$ and $l_{i,2}$, introduce a new variable y_i and replace c_i with a conjunction of clauses z_i, where,

$$z_i = (l_{i,1} + l_{i,2} + y) \wedge (l_{i,1} + l_{i,2} + \overline{y_i})$$

- Case 4: If $c_i = (l_{i,1} + l_{i,2} + \cdots + l_{i,k})$, where $k > 3$, then introduce $(k-3)$ new variables: $y_1, y_2, \ldots y_{k-3}$ and replace c_i with a sequence of clauses z_i, where,

$$z_i = (l_{i,1} + l_{i,2} + y_1) \wedge (\overline{y_1} + l_{i,3} + y_2) \ldots (\overline{y_{j-2}} + l_{i,j} + y_{j-1}) \ldots (\overline{y_{k-4}} + l_{i,k-2} + y_{k-3}) \wedge (\overline{y_{k-3}} + l_{i,k-1} + l_{i,k})$$

This reduction will be correct if the following theorem is true:

Theorem 1 *For all $z_i \in Z$, where Z is an instance to 3SAT, z_i is satisfiable if and only if $c_i \in C$ is satisfiable, where C is an instance to SAT.*

Proof: ($\Rightarrow$) When k=1, let $c_i = l_i$, then z_i is,

$$z_i = (l_i + y_{i,1} + y_{i,2}) \wedge l_i + \overline{y_{i,1}} + y_{i,2}) \wedge l_i + y_{i,1} + \overline{y_{i,2}}) \wedge l_i + \overline{y_{i,1}} + \overline{y_{i,2})}$$

if l_i is True, then z_i is True and if l_i is False, then z_i is False.

When k=2, let $c_i = (l_{i,1} + l_{i,2})$, then z_i is,

$$z_i = (l_{i,1} + l_{i,2} + y_i) \wedge (l_{i,1} + l_{i,2} + \overline{y_i})$$

if either $l_{i,1}$ or $l_{i,2}$ are True, then z_i is True and if both $l_{i,1}$ and $l_{i,2}$ are False, then z_i is False.

When k¿3, let $c_i = (l_{i,1} + l_{i,2} + \cdots + l_{i,k})$, then z_i is,

$$z_i = (l_{i,1} + l_{i,2} + y_1) \wedge (\overline{y_1} + l_{i,3} + y_2) \ldots (\overline{y_{j-2}} + l_{i,j} + y_{j-1}) \ldots (\overline{y_{k-4}} + l_{i,k-2} + y_{k-3}) \wedge (\overline{y_{k-3}} + l_{i,k-1} + l_{i,k})$$

where $y_1, y_2, \ldots, y_{k_3}$ are new variables. First lets say c_i is satisfiable. We need to prove that z_i is also satisfiable.

- If either $l_{1,1}$ or $l_{i,2}$ is True, set all the additional variables $y_1, y_2, \ldots y_{k-3}$ to False. This implies that the first term of all the clauses in z_i other than the first clause has a literal $\overline{y_n}$ that evaluates to True, implying that z_i has a satisfying assignment.

- If either $l_{i,k-1}$ or $l_{i,k}$ is True, set all the variables $y_1, y_2, \ldots, y_{k-3}$ to True. This implies that the third term of all the clauses in z_i other than the last has a literal y_n which evaluates to True, implying that z_i has a satisfying assignment.

- If $l_{i,j}$ where $j \notin 1, 2, k-1, k$ is True, set $y_1, \ldots, y_{j-2}$ to True and $y_{j-1}, \ldots, y_{k-3}$ to False. Let us call the clause in z_i containing $l_{i,j}$ as C'. Then it implies that the third term of all the clauses in z_i left to C' has a literal y_n, where $n \in 1, \ldots j-2$, that evaluates to True and the first term of all the clauses in z_i right to C' has a literal $\overline{y_n}$, where $n \in j-1, \ldots, k-3$, that evaluates to True, implying that z_i has a satisfying assignment.

Now we prove that if c_i has no such assignment,i.e., if c_i is not satisfiable then z_i is also not satisfiable. When c_i is not satisfiable, then no literal in $l_{i,1}, \ldots, l_{i,k}$ is True. For z_i to be satisfiable, all it's clauses must evaluate to True. For the first clause to be True, y_1 =True. Likewise even if all the variables $y_2, \ldots, y_{k-3}$ are True, the last clause evaluates to False. Hence, z_i is not satisfiable, if c_i is not. $\rightarrow\leftarrow$.

■

4 Brute Force Algorithm

BruteForce3SAT($G = C, p$)

Input: A boolean formula C and a positive integer p, where each clause $c_i \in C$ has exactly 3 variables.

Output: An assignment for the variables in C such that at least p clauses in C are satisfied.

Let A be a set of all possible kinds of assignments on variables in C.

```
for For all assignments A_i in A do
    if at least p clauses in C are satisfied then
        return A_i
return "The boolean formula C is not satisfiable for at least p clauses"
```

The algorithm considers all possible assignments of variables in C. If there are n variables in C, then BruteForce3SAT checks if C is satisfiable for at least p clauses in time $O(2^n)$.

5 Approximation Algorithms

Here we give a 7/8-approximation algorithm for the optimization version of 3SAT that relies on the property of random variables, the linearity of expectation property. The optimization version of 3SAT is called Max 3SAT or maximal 3SAT, and is defined as follows:

MAX 3SAT
Input: A collection of clauses: $c_1, c_2, \ldots, c_m$
Goal: Find the assignment to variables of C: $x_1, \ldots, x_n$ that satisfies the maximum number of clauses.

Clearly, since 3SAT is NP-Complete as shown above, it implies that MAX 3SAT is NP-Hard. Essentially, the MAX 3SAT is a maximization problem. The randomized approximation algorithm for this problem is as follows:

Approx-MAX3SAT(C)

Input: A boolean formula C that is a collection of clauses: $c_1, c_2, \ldots, c_m$
Output: The assignment to variables of C: $x_1, \ldots, x_n$ that satisfies the maximum number of clauses.

```
for i = 1 to n do
    Flip a fair coin
    if Heads then
        x_i ← 1
    if Tails then
        x_i ← 0
return x
```

The running time of this algorithm is $O(n)$ (certainly polynomial).

To show that Approx-MAX3SAT returns a 7/8-approximation algorithm, consider the following definition of linearity of expectations property, and the theorem that follows.

Linearity of expectations
Definition: Given two random variables, X, Y (not necessarily independent) we have the joint expectation $\boldsymbol{E}[X+Y] = \boldsymbol{E}[X] + \boldsymbol{E}[Y]$.

Theorem 2 Approx-MAX3SAT *is a 7/8-approximation algorithm to MAX 3SAT that runs in polynomial time.*

Proof: Let $x_1, x_2, \ldots, x_n$ be the n variables used in the given instance. The algorithm works by randomly assigning values to $x_1, x_2, \ldots, x_n$, independently, with equal probability, to 0 or 1, for each one of the variables. Let Y_i be the indicator variables which is 1 if and only if the ith clause is satisfied by the random assignment and 0 otherwise, for $i = 1, \ldots, m$, i.e.,

$$Y_i = \begin{cases} 1, & \text{if } \mathrm{C}_i \text{ is satisfied by the generated assignment} \\ 0, & \text{otherwise} \end{cases}$$

The number of clauses satisfied by the above assignment is $Y = \sum_{i=1}^{m} Y_i$, where m is the number of clauses in the input. From linearity of expectations, we have,

$$\boldsymbol{E}[Y] = \boldsymbol{E}[\textstyle\sum_{i=1}^{m} Y_i] = \sum_{i=1}^{m} \boldsymbol{E}[Y_i]$$

Now, the probability that $Y_i = 0$ is that all three literals appear in the clause C_i are evaluated to be False. As a literal cannot be repeated in the same clause or a literal and it's compliment cannot be in the same clause, the three literals in C_i are three distinct variables and their assignment events are independent. So the probability that C_i is not satisfied is,

$$\boldsymbol{Pr}[Y_i = 0] = 1/2 * 1/2 * 1/2 = 1/8$$

Thus, in the eight possible assignments that can happen for variables in a clause C_i, the probability of picking all three assignments as False is 1/8. So, the probability of picking an assignment such that at least one of the variable is True is 7/8 and the expectation of Y_i being True is,

$$\boldsymbol{E}[Y_i] = \boldsymbol{Pr}[Y_i = 0] * 0 + \boldsymbol{Pr}[Y_i = 1] * 1 = 7/8$$

Since the optimal solution for MAX 3SAT can satisfy at most m clauses and the expectation that all clauses are satisfied is $\boldsymbol{E}[Y] = \sum_{i=1}^{m} \boldsymbol{E}[Y_i] = (7/8)M$, it is shown that the above Approx-MAX3SAT algorithm gives a (7/8)-approximation to MAX 3SAT problem. ■

6 Applications/Other Interesting Things

An interesting application of 3-SAT that I have found is to provide secure set membership, a cryptographic primitive. Secure set membership is a general problem for participants holding set elements to generate a representation of their set that can then be used to prove knowledge of set elements to others. Set membership protocols are used for authentication problems such as digital credentials and some signature problems such as time stamping [1].

Another interesting application of 3-SAT or SAT that I have found is a multi-agent task allocation problem. In this problem, there are N agents, M projects and K goals such that, for each goal k, there is a set M^k projects. Any of the projects in M^k can fulfill the goal k. For each agent i, there is a set M_i of projects he can complete, but he has time to finish exactly one of them. The task here is to find an assignment, if it exists, so it can fulfill all goals [3].

References

[1] Michael de Mare and Rebecca N Wright. Secure set membership using 3sat. In *International Conference on Information and Communications Security*, pages 452–468. Springer, 2006.

[2] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[3] Holger Voos. Market-based control of complex dynamic systems. In *Proceedings of the 1999 IEEE International Symposium on Intelligent Control Intelligent Systems and Semiotics (Cat. No. 99CH37014)*, pages 284–289. IEEE, 1999.