New ¾ - Approximation Algorithms for MAX SAT

Michel X. Goemans[1] and David P. Williamson[2]

Report

Dineshbalu Balakrishnan (2698478)

1 Department of Mathematics – MIT – MA.

2 School of OR – Cornell University – NY.

The topics that are covered in this report are the following:

- Problem Statement
- Previous Results
- Approx Algorithms for MAX SAT – Classification
  - Johnson’s Algorithm and the Probabilistic Method
  - A (1 – 1/e) – Approximation Algorithm
  - Actual ¾ - Approximation Algorithm
- Remarks
- Applications - SAT
- References & Links

Introduction:

Problem Statement:

First let’s state the problem of MAX SAT as follows,

An instance of the MAX SAT (Maximum Satisfiability) Problem is defined by,

- A collection ‘C’ of Boolean clauses ($C_1, C_2, \ldots, C_m$)
- Each clause is a disjunction of literals
- A literal is drawn from a set of variables $\{x_1, x_2, \ldots, x_n\}$
- A literal may be a variable x or it’s negation $\overline{x}$
- Also, each clause $C_{\mathbf{j}} \in C$ has an associated nonnegative weight $w_{\mathbf{j}}$.

An **Optimal Solution** is achieved by maximizing the sum of the weight of the satisfied clauses (clauses with 1 or more literals), by the assignment of truth values to variables $x_1, x_2, \ldots, x_n$..

Note: If each clause contains at most two literals then the problem is said to be MAX2SAT and so on for others.

Previous Results:

Let's see some of the previous results in MAX SAT problems:

1) Johnson's algorithm demonstrated by Johnson obtained a ½ - Approximation Algorithm, which is also an
$(1 – 1/2^k)$ – Approximation Algorithm, if each clause contains at least k literals.

2) Lieberherr and **Specker** gave a $\frac{\sqrt{5} - 1}{2}$ Approximation algorithm (0.618), which is achieved when the clause set does not contain both clauses x_i & $\overline{xi}$ for any i.

3) Kohli and **Krishnamurthy** gave a randomized algorithm whose solution has expected weight of at least 2/3 of the optimal solution.

4) Yannakakis recently developed a ¾ - Approximation Algorithm, but the instance does not contain any unit clauses (clauses with only 1 literal). This is achieved by transforming the MAX SAT instance into an equivalent instance. Maximum flow computations are used to transform MAX 2SAT instances.

Johnson's Algorithm and the Probabilistic Method:

Johnson's algorithm demonstrated by Johnson obtained a ½ - Approximation Algorithm, which is also an $(1 – 1/2^k)$ – Approximation Algorithm, if each clause contains at least k literals.

This is explained as follows,

If we independently and randomly set each variable x_i to be true with probability p_i, then

$$\hat{W} = \sum_{Cj \in C} w_j \left(1 - \prod_{i \in Ij} (1 - p_i) \prod_{i \in Ij} p_i\right)$$

where,

$\hat{\mathbf{W}}$ = expected weight of clauses satisfied by the probabilistic assignment.

$\mathbf{I^+_j}$ **(and** $\mathbf{I^-_j}$**)** = set of variables appearing unnegated (and negated) in C_j.

In the method of conditional probabilities, the value of the ith variable is determined in the ith iteration. Given the values of x1, ..., x_{i-1} , calculate the expected weight of clauses satisfied by the probabilistic assignment, given the current assignment to x1, ..., x_{i-1} and the assignment $x_i = 1$. Then, calculate the expected weight given the assignment to x1, ..., x_{i-1} and $x_i = 0$. Finally assign the value that maximizes the conditional expectation. Each of these can be calculated in polynomial time and hence the algorithm takes polynomial time and the assignment produced has weight at least $\hat{W}$.

Johnson's algorithm sets p_i=1/2 for all i and uses the method of conditional probabilities.

For this choice of p_i, we have,

$$\hat{W} \geq \sum_{Cj \in C} (1 - ½)\ wj = ½ \sum_{Cj \in C} w_j$$

Since the optimum assignment can have weight at most $\sum_{j} w_j$,

this proves that Johnson's algorithm is a **½**-approximation algorithm.

Also, if all clauses have at least **k** literals, then,

$$\hat{W} \geq (1 - ½^k) \sum_{Cj \in C} w_j$$

this proves that Johnson's algorithm is a (1 – $½^k$)-approximation algorithm for this case of clauses having k literals.

<u>Let's prove the Johnson's algorithm by an example</u>:

Let x, y, z, w be the set of variables.
Consider the collection of clauses (made up of literals),

$$(X \vee Y \vee Z) \wedge (\overline{X} \vee \overline{Y} \vee W) \wedge (\overline{W} \vee Y \vee Z)$$

where,

$\overline{X}$, $\overline{Y}$ and $\overline{W}$ are negated variables.

Now let's assign a weight '2' to each clause. Substitute the values in the equation,

$$\hat{W} = \sum_{Cj \in C} w_j \left(1 - \prod_{i \in Ij} (1 - p_i) \prod_{i \in Ij} p_i\right)$$

$2(1-(½\ ½\ ½)(0)) + 2(1-(½)(½\ ½)) + 2(1-(½\ ½)(½))$

=> 2(1) + 2(1-1/8) + 2(1-1/8)
=> 2 + 14/8 + 14/8

=> 5.5 = $\hat{W}$

$$\hat{W} \geq ½ \sum_{Cj \in C} w_j$$

=> 5.5 ≥ ½ (2+2+2)

=> 5.5 ≥ 3

This proves the result when the number of literals is 3.

Now, when the number of literals is 1:

Let X, Y be the set of variables.

$(X) \wedge (\overline{X}) \wedge (Y)$ => Weight of each clause = 1

=> 1(1 – ½ (0)) + 1(1 – 0 (½)) + 1(1 – ½ (0))

=> 1 + 1 + 1 = 3 = $\hat{W}$

$$\hat{W} \geq ½ \sum_{Cj \in C} w_j$$

=> 3 ≥ ½ (1+1+1)

=> 3 ≥ 3/2

This proves the result when the number of literals is 3.

Note that in the one literal case, the sum of the weights of the clauses will be exactly half of the expected weight of the clauses. Also, this case is the worst case for the Johnson’s Algorithm.

A (1 – 1/e) – Approximation Algorithm:

Next is the (1 – 1/e) approximation algorithm,
Consider the following integer program:

$$\text{Max} \sum_{Cj \in C} w_j z_j$$

subject to:

$$\text{(IP)} \qquad \sum_{Cj \in C} y_i + \sum_{Cj \in C} (1 - y_i) \geq z_j$$

$$y_i \in \{0,1\} \qquad 1 \leq i \leq n$$
$$0 \leq z_j \leq 1 \qquad \forall\, C_j \in C$$

By associating, y_i=1 with x_i set true, y_i=0 with x_i set false, Z_j=1 with clause C_j satisfied, Z_j=0 with clause C_j not satisfied, we can see that the IP exactly corresponds to the MAX SAT problem and the optimal value Z^*_{IP} is equal to the optimal value of the MAX SAT problem.

Now make *linear programming relaxation* of IP i.e., replace $y_i \in \{0,1\}$ constraints with the constraints **0 £ y_i £ 1**. Call this linear program.

Obviously the optimal value of LP is an upper bound on the optimal value of IP; that is, **Z^*_{LP} ³ Z^*_{IP}.**

Here we note that,

- When no unit clauses, the solution y_i=1/2 for all i and Z_j=1 for all j, which is of value $\sum_{Cj \in C} w_j$, is optimal, independent of the weights w_j. Hence the relaxation is vacuous in this case.
- When there are unit clauses, the relaxation provides some useful information (proved in the coming lemma).

Now we show that by using randomized rounding we obtain a (1 – 1/e)-approx algorithm:

- First solve the LP. Let (y^*,z^*) be the optimal solution.
- Secondly apply the method of conditional probabilities with $p_i=y^*_I$ for all i to derive an assignment.

Here, $\hat{W}$ is compared to Z^*_{LP} and not to $\sum_{Cj \in C}$ as in the case of Johnson's algorithm.

If $1 - \prod_{i\in Ij} (1 - y_i) \prod_{i\in Ij} y_i \geq \alpha z_j$ -------- **1**

for any feasible solution (y,z) to LP and for any clause C_j, then,

$$\hat{W} = \sum_{Cj} w_j \{ (1 - \prod_{i\in Ij} (1 - p_i) \prod_{i\in Ij} p_i) \}$$

$$= \sum_{Cj} w_j \{ (1 - \prod_{i\in Ij} (1 - y^*_i) \prod_{i\in Ij} y^*_i)\}$$

$$³ \alpha \sum_{Cj} w_j z^*_j = \alpha Z^*_{LP} \geq \alpha Z^*_{IP}$$

(from 1)

The resulting algorithm is an α-approximation algorithm.

<u>Lemma</u>: (for k literals),

For any feasible solution (y,z) to LP and for any clause C_j with k literals, we have

$$1 - \prod_{i\in Ij} (1 - y_i) \prod_{i\in Ij} y_i \; ³ \; \beta_k Z_j$$

where,

$$\beta_k = 1 - (1 - 1/k)^k$$

This proof uses the following results,

- a concave function f(x) satisfies f(x) ³ ax+b over the interval [l,u], if the endpoints of the interval, namely f(l) ³ al+b and f(u) ³ au+b.
- the arithmetic/geometric mean inequality states that,

$$\frac{a_1 + a_2 + \dots + a_k}{k} \quad ³ \quad \sqrt[k]{a1a2...ak}$$

for any collection of nonnegative numbers $a_1, a_2, \dots, a_k$

<u>3/4 – Approximation Algorithms</u>:

Johnson's algorithm is a ¾-approximation algorithm if all clauses contain **at least 2 literals**.

The previous alg. is a ¾-approximation alg. if all clauses contain **at most 2 literals** (MAX 2SAT).

The **¾-approximation algorithm** in this section is obtained by choosing the best truth assignment between the 2 outputs by the previous algorithms. More formally,

<u>Theorem</u>: Let $\hat{W}_1$ denote the expected weight corresponding to $p_i=1/2$ for all i and let $\hat{W}_2$ denote the expected weight corresponding to $p_i=y^*_i$ for all i where (y^*, z^*) is an optimum solution to the LP relaxation.

Then,

$$\max(\hat{W}_1, \hat{W}_2) \geq (\hat{W}_1 + \hat{W}_2)/2 \geq ¾\ Z^*_{LP}$$

The explanation for this theorem is as follows,
The first inequality is trivially satisfied. Then, let C^k denote the set of clauses with exactly k literals.

From Johnson's algorithm, we know that,

$$\hat{W}_1 = \sum_{k \geq 1} \sum_{Cj \in C} \alpha_k w_j \geq \sum_{k \geq 1} \sum_{Cj \in C} \alpha_k w_j z^*_j$$

where, $\alpha_k = (1 - 1/2^k)$

From Lemma, $\hat{W}_2 \geq \sum_{k \geq 1} \sum_{Cj \in C} \beta_k w_j z^*_j$

where, $\beta_k = 1 - (1 - 1/k)^k$

As a result, (sum both of them and divide by 2),

$$(\hat{W}_1 + \hat{W}_2) / 2 \geq \sum_{k \geq 1} \sum_{Cj \in C} (\alpha_k + \beta_k)/2 \ w_j z^*_j$$

Now apply values ok k to α and β ,

Clearly, $\alpha_1 + \beta_1 = \alpha_2 + \beta_2 = 3/2$
For **k ³ 3**, $\alpha_k + \beta_k \geq 3/2$

Therefore, we obtain,

$$(\hat{W}_1 + \hat{W}_2) / 2 \geq \sum_{k \geq 1} \sum_{Cj \in C} 3/4 \ w_j z^*_j = \tfrac{3}{4} Z^*_{LP}$$

This proves that this is a ¾-Approximation Algorithm.

Remarks:

Some of the remarks on this paper and on MAX SAT are stated below:

- Performance guarantee of Johnson's algorithm = **¾**, if $k \geq 2$

  (k =no. of literals)

- Better performance guarantee is possible only by strengthening the linear programming relaxation. Recent research has shown that a **0.878-approximation algorithm** for MAX 2SAT is obtained by using a form of randomized rounding on a non-linear programming relaxation.
- MAX SAT is NP-Complete (Nondeterministic Polynomial time complete), even for MAX 2SAT. So, polynomial-time algorithms cannot optimally solve MAX SAT.
- It is a core of computationally intractable NP-complete problems (as in mathematical logic and computing theory).
- The first problem ever shown to be NP-complete was the *satisfiability problem*.
- Traditional methods treat SAT as a discrete, constrained decision problems, but in recent research, many optimization methods, parallel algorithms and practical techniques have been developed to solve SAT problems.

Applications:

The SAT problems have direct applications in,

- Mathematical logic
- Artificial Intelligence
- Robotics
- VLSI Engineering
- Computing theory

- Machine Vision
- Computer-aided manufacturing
- Computer Graphics
- Text processing

References & Links:

New ¾ - Approximation Algorithms for MAX SAT

by Michel X. Goemans and David P. Williamson

SIAM Journal of Discrete Mathematics, Volume 7, Number 4, 1994

http://citeseer.nj.nec.com/12959.html

Applications - SAT:

More information about the applications of SAT can be obtained from,

http://manip.crhc.uiuc.edu/Wah/paper.html