NEW ¾ APPROXIMATION ALGORITHMS FOR THE MAXIMUM SATISFIABILITY PROBLEM

MICHEL X. GOEMANS AND DAVID P. WILLIAMSON (1994)

Some Terminology

- We have a collection C of boolean clauses and set of variables $\{x_1,x_2....x_n\}$.
- Each clause is a disjunction of some literals taken from the set of variables.
- A literal can be either the variable x or its negation !x
- For example, x_1 v x_5 v $!x_6$ (same as x_1 or x_5 or $!x_6$) is a clause

MAXIMUM SATISFIABILITY PROBLEM

- We have associated a non-negative weight w_j with each of our clauses in our collection C
- Now the problem is to assign boolean values 0/1 to each of our variable so that the sum of the weights of the satisfied clauses is maximum. Satisfied clauses mean clauses with at least one true literal. This is the MAX SAT problem.
- For example *clause* C_1 □ $x_1 \vee x_2$, *clause* C_2 □ $!x_2 \vee !x_3$, *clause* C_3 □ $!x_1 \vee x_3$ *and the weights be* $w_1 = 5$ *,* $w_2 = 10$ *and* $w_3 = 3$ *, then for the assignment* $\{x_1, x_2, x_3\} = \{1, 0, 0\}$ *we have the Total wt =* $5 + 10 + 0 = 15$ *because clauses 1 and 2 are satisfied but clause 3 is not*

MAXIMUM SATISFIABILITY PROBLEM

- MAX SAT problem is known to be NP-complete even if we restrict each clause to have at most two literals (called the MAX 2SAT problem) , hence we rather focus on good approximate algorithms for this problem rather than optimal ones.
- So we focus on algorithms called as **α**-approximation algorithms. An **α**-approximation algorithm for MAX-SAT problem is a poly-time algorithm which gives us an assignment of values for the variables so that the

$$\text{Total gain} >= \alpha \text{ (Optimal Total gain)}$$

where $\alpha \epsilon [0,1]$

MAXIMUM SATISFIABILITY PROBLEM

- The paper we are presenting gives a ¾ - approximation algorithm for the MAX-SAT problem
- *Johnson's Algorithm* gives a ½- approximation algorithm which additionally is actually a $(1-1/2^k)$ - approximation algorithm if each clause is restricted to have at least k-literals. We'll see Johnson's algorithm in the upcoming slides.
- The author of this paper himself gives a (1-1/e)-approximation algorithm using technique of randomized rounding to the solution of a linear program that is a LP-relaxation of an ILP formulation for the MAX-SAT problem.
- A ¾-approximate algorithm is arrived at by combining the above two techniques cleverly.
- A ¾ -approximate algorithm is also directly obtained by modifying the randomized rounding scheme of (1-1/e)-approximation algorithm.

Johnson's algorithm and the probabilistic method

- Set each variable x_i independently and randomly to be true with probability p_i

$$\hat{W} = \sum_{C_j \in C} w_j \left(1 - \prod_{i \in I_j^+} (1 - p_i) \prod_{i \in I_j^-} p_i\right)$$

- Since we see that expected value of our total weight is W, we notice there has to exist an assignment of truth values whose weight is at least the expected value because if all possible assignments give lesser weight compared to W, then the expected value cannot be W.
- Johnson's algorithm essentially sets p_i = ½ for all i. For this choice of p_i's, we see

$$\hat{W} \geq \sum_{C_j \in C} \left(1 - \frac{1}{2}\right) w_j = \frac{1}{2} \sum_{C_j \in C} w_j.$$

Johnson's algorithm and the probabilistic method

$$\hat{W} \geq \left(1 - \frac{1}{2^k}\right) \sum_{C_j \in C} w_j,$$

- Now if we restrict every clause to have at least k literals then our expectation become better with increasing k.
- So Johnson's algorithm is an $(1-1/2^k)$ - approximation algorithm

Constructing an assignment

- We use the method of conditional probabilities, the value for ith variable is determined in the i^{th} iteration , so x_i variable is determined using the already assigned values of x_1 , x_2 .... x_{i-1} . Calculate expected wt using values of (x_1 x_2 ... x_{i-1}) and setting x_i=1 , and then using x_i=0. Choose the maximum of the two.

A (1 - 1/e) - approximation algorithm

- Consider the following integer linear program which exactly solves our MAX SAT problem.

$$
(IP) \qquad
\begin{array}{lll}
\text{Max} & \displaystyle\sum_{C_j \in \mathcal{C}} w_j z_j & \\
\text{subject to:} & & \\
& \displaystyle\sum_{i \in I_j^+} y_i + \sum_{i \in I_j^-} (1 - y_i) \geq z_j & \forall C_j \in \mathcal{C} \\
& y_i \in \{0, 1\} & 1 \leq i \leq n \\
& 0 \leq z_j \leq 1 & \forall C_j \in \mathcal{C}.
\end{array}
$$

- Here y_i represents our variables and z_j represent our clauses, We associate y_i = 1 when i^{th} variable is to be set true and z_j = 1 when clause C_j is satisfied.

A (1 - 1/e) - approximation algorithm

- We can now relax the ILP to get a LP
- LP is formed by replacing $y_i \in \{0,1\}$ to $0 \leq y_i \leq 1$
- Since obviously the constraints of LP are less tight compared to the ILP, hence the solution of our LP cannot be any worse than ILP's optimum.
- Observe additionally that for our Linear program, if we restrict every clause to have at least two literals, then an optimal solution to the LP is clearly visible which is setting $y_i=½$ for all i and $z_j=1$ for all j which gives the optimal value of $\sum w_j$

A (1 - 1/e) - approximation algorithm

- Consider our LP. Let (y^*, z^*) be an optimal solution of LP
- Now we'll present a (1-1/e) - approximation algorithm for our MAX SAT problem which makes use of (y^*,z^*)
- The idea here is to apply the method of conditional probabilities with p_i=y^*_i
- In Johnson's method , all p_i were set to 0.5 and here we need to set them to respective y^*_i and construct an assignment like we did previously.
- All that is left to prove is that this algorithm is a (1-1/e) - approximation algorithm
- Consider the following inequality and suppose that any optimal solution of the LP, (y,z) that we use satisfies this for all clauses C_j and some fixed **α**, then we can show our algorithm becomes an **α**-approximation algorithm.

$$1 - \prod_{i \in I_j^+} (1 - y_i) \prod_{i \in I_j^-} y_i \geq \alpha z_j$$

A (1 - 1/e) - approximation algorithm

- For any feasible solution (y, z) of LP and any clause C_j which has k-literals we can prove that

$$1 - \prod_{i \in I_j^+} (1 - y_i) \prod_{i \in I_j^-} y_i \geq \beta_k z_j$$

$$\beta_k = 1 - \left(1 - \frac{1}{k}\right)^k .$$

- The value $\boldsymbol{\beta}_k$ decreases with k , hence the previous two results together show that the algorithm is $\boldsymbol{\beta}_k$ approximate for the class of MAX SAT instances in which number of literals in any clause is at most k.

- Since $\lim_{k \to \infty} \boldsymbol{\beta}_k = 1 - 1/e$, the algorithm is (1-1/e) approximate for MAX SAT in general (this bound is tight as well)

A simple ¾ - approximation algorithm

- Now the idea is to choose the best truth assignment between the output by Johnson's algorithm and the previous algorithm.
- Let W_1 and W_2 denote the expected weight corresponding to $p_i = ½$ and $p_i = y_i^*$ for all i where (y^* , z^*) is a solution to the LP relaxation. Then,

$$\max(\hat{W}_1, \hat{W}_2) \geq \frac{\hat{W}_1 + \hat{W}_2}{2} \geq \frac{3}{4} Z^*_{LP}.$$

- So, the algorithm would be, with probability ½, set the the values p_i's to be either $p_i = ½$ for all i, or $p_i = y_i^*$ for all i, and apply the method of conditional probabilities.

A class of ¾-approximation algorithms

- We can use randomized rounding to directly get a ¾-approximate algorithm
- For this we instead of using $p_i = y_i^*$ we use $p_i = f(y_i^*)$ for some carefully chosen function f : [0,1] □[0,1]
- As before if we show the following inequality to be true for any feasible solution (y, z) of LP and for any clause C_j then the algorithm will be ¾-approximate

$$1 - \prod_{i \in I_j^+} (1 - f(y_i)) \prod_{i \in I_j^-} f(y_i) \geq \frac{3}{4} z_j \tag{1}$$

A class of ¾-approximation algorithms

- The previous inequality motivates the following definition :

DEFINITION 5.1. *A function $f : [0,1] \to [0,1]$ has property $\frac{3}{4}$ if*

$$1 - \prod_{i=1}^{l}(1 - f(y_i)) \prod_{i=l+1}^{k} f(y_i) \geq \frac{3}{4} \min\left(1, \sum_{i=1}^{l} y_i + \sum_{i=l+1}^{k} (1 - y_i)\right)$$

for any k, l with $k \geq l$ and any $y_1, \ldots, y_k \in [0,1]$.

- From previous discussions it is easy to see that any function with property ¾ induces a ¾ - approximate algorithm.
- Now we need to prove that functions with property ¾ exist

A class of ¾-approximation algorithms

- Some class of functions with property ¾ are given as follows :

THEOREM 5.2. *Any function f satisfying*

$$1 - 4^{-y} \leq f(y) \leq 4^{y-1}$$

for all $y \in [0, 1]$ has property $\frac{3}{4}$.

THEOREM 5.3. *The linear function $f_\alpha(y) = \alpha + (1 - 2\alpha)y$, where*

$$2 - \frac{3}{\sqrt[3]{4}} \leq \alpha \leq \frac{1}{4},$$

has property $\frac{3}{4}(2 - \frac{3}{\sqrt[3]{4}} \approx .11)$.

THEOREM 5.4. *The function*

$$f(y) = \begin{cases} \frac{3}{4}y + \frac{1}{4} & \text{if } 0 \leq y \leq \frac{1}{3} \\ \frac{1}{2} & \text{if } \frac{1}{3} \leq y \leq \frac{2}{3} \\ \frac{3}{4}y & \text{if } \frac{2}{3} \leq y \leq 1 \end{cases}$$

has property $\frac{3}{4}$.

A class of ¾-approximation algorithms

- For proving that the functions mentioned earlier have ¾ property we can first prove the following lemma , to restrict our attention to the case l = k

LEMMA 5.5. *Let* $g : [0,1] \to [0,1]$ *be a function satisfying*

$$1 - \prod_{i=1}^{k}(1 - g(y_i)) \geq \frac{3}{4}\min\left(1, \sum_{i}^{k} y_i\right) \tag{2}$$

for all k *and all* $y_1, y_2, \ldots, y_k \in [0,1]$. *Consider any function* $f : [0,1] \to [0,1]$ *satisfying*

$$g(y) \leq f(y) \leq 1 - g(1-y) \tag{3}$$

for all $y \in [0,1]$. *Then* f *has property* $\frac{3}{4}$.

- Using the above lemma we can prove that the functions mentioned earlier have ¾ property.

Concluding Remarks

- The above discussions prove an upper bound on the optimal LP solution, that is

$$Z^*_{LP} \leq \frac{4}{3} Z^*_{IP}.$$

- We can also show that this a tight upper bound.
- Note that in any case with no unit clauses, all of the previous algorithms reduce to Johnson's algorithm, which is a ¾ - approximation algorithm for such cases and the author gives instances that are tight for his algorithm.
- So, the performance guarantee of ¾ of the discussed algorithms is also tight.

Thank You

- Aman Tiwari (160094)
- Ankush Tiwari (160130)
- Pushkar Dixit (160529)