数理解析研究所講究録 1325 巻 2003 年 164-168

Test Instance Generation for MAX 2SAT with Fixed Optimal Value

北陸先端科学技術大学院大学・情報科学研究科 元木 光雄 (Mitsuo Motoki)
Japan Advanced Institute of Science and Technology
School of Information Science

Abstract

We propose a randomized instance generation algorithm for MAX 2SAT. This algorithm generates a test instance at random such that the number of unsatisfied clauses at the optimal solution is exactly 1 with probability 1. We prove that the number of clauses of obtained instances is larger than the number of variables with high probability. We note that this threshold coincides with the threshold for unsatisfiability of random 2CNF [3, 4].

1 Introduction

MAX 2SAT is one of famous combinatorial optimization problems. It stated as follows: given a 2CNF formula, find a truth assignment that maximizes the number of satisfied clauses. Since MAX 2SAT is NP-complete, many polynomial time approximation algorithms have been proposed (for example, see [1, 5, 6]). To evaluate performance of these approximation algorithms, there are two methods. One is theoretical analysis and the other is empirical study. In theoretical analysis, the worst ratio of an approximation value obtained by the approximation algorithms to the optimal value, a performance guarantee, is often used to characterize their performance. Usually, this performance guarantee only considers the worst case. On the other hand, the empirical study can evaluate from several point of view, for example, the average ratio of an approximation value to the optimal value. However, test instances are necessary for evaluating approximation algorithms experimentally. When one esitimates the performance guarantee experimentally, test instances should have their optimal solution. Otherwise it is impossible to estimate how good approximation solutions are obtained. Of course, there exist a number of benchmark problems with their optimal solution. Nevertheless, we do not have enough number of test instances and we want to evaluate for various test instances. Thus we consider a problem generating test instance with its optimal solution at random.

In this paper, we propose a randomized instance generation algorithm generating a test instance where the number of unsatisfied clauses at the optimal solution is exactly 1. The outline of algorithm is as follows: First, choose one truth assignment t as the optimal solution. Then add one randomly chosen clause unsatisfied by t. Finally add clauses satisfied by t at random until the instance is satisfiable. Since 2SAT is linear time solvable, it is feasible to recognize wrong instances, e.g., satisfiable formulas. Thus instances generated by our algorithm have their optimal solution.

For MAX 3SAT, we proposed a similar randomized test instance generation algorithm [8]. This algorithm generates a 3CNF formula and a truth assignment that is the optimal solution with high probability. The difference between our instance generators for MAX 2SAT and MAX 3SAT is that instance

```
input: the number of variables n
begin
  let F = ∅;
  choose 4 distinct variables x_r1, x_r2, x_r3, and x_r4 at random;
  add {x_r1, x_r2} and {¬x_r3, ¬x_r4} to F;
  while F is satisfiable do
    choose a pair of two distinct variables (x_r1, x_r2) at random
    add {x_r1, ¬x_r2} to F;
  end;
  choose a truth assignment f̂ ∈ {0,1}^n at random;
  negate literals whose underlying variables are assigned 0 on f̂;
  output F and f̂;
end.
```

Figure 1: Algorithm.

generator for MAX 3SAT sometimes outputs wrong optimal solution because of NP-completeness of 3SAT.

For this MAX 2SAT instance generation algorithm, we analyze that how many clauses are added until the algorithm halts. We prove that a threshold of the number of clauses is the number of variables, that is, the algorithm halts when the number of clauses is larger than the number of variables with high probability, but does not halt when the number of clauses is equal to the number of variables with high probability. We note that this result coincides the threshold for unsatisfiability of 2CNF formula [3, 4].

Here, we describe some notations. Let $X = \{x_1, \ldots x_n\}$ be a set of n propositional variables. A *literal* is a propositional variable x or its negation $\overline{x}$. A *clause* is a disjunction of exactly 2 literals which is denoted by complement-free set of exactly 2 literals. We use a multiset of m clauses $\{C_1, \ldots, C_m\}$ to denote the formula F. Throughout this paper, we use n and m to denote the number of available variables and the number of clauses respectively.

2 Algorithm

Now we introduce our instance generation algorithm formally.

Our basic idea is as follows. Let F and $\hat{t}$ be the output instance and truth assignment of our generator for given the number of variables n. Without loss of generality, we assume that 1^n is chosen as $\hat{t}$. There must exist exactly one clause unsatisfied by $\hat{t}$. This clause shall consist of two negative literals. We show a pseudo code of our algorithm in Figure 1.

To simplify the analisys, we restrict the number of clauses that is unsatisfied by 0^n to 1. This clause consists of two positive literals. Hence the remaining clauses consist of one positive literal and one negative literal.

3 Analysis

Our results are proven by similar way to the threshold of random 2CNF [4].

Before we describe our results, we introduce a *formula graph* [2]. A formula graph over n variables is a directed graph $G_F = (V, E)$, with V is the set of all literals, $X \cup \{\overline{x} \mid x \in X\}$, and

$$E = \left\{(v_i, \overline{v_j}) \mid v_i, v_j \in V, v_i \neq v_j, v_i \neq \overline{v_j}, \text{and } \left\{v_i, v_j\right\} \in F\right\}.$$

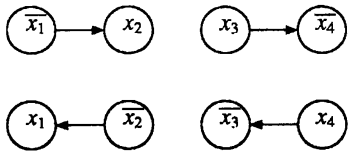

Figure 2: Initial state of formula graph G_F

We note that $(v_i, \overline{v_j}) \in E \Leftrightarrow (\overline{v_i}, v_j) \in E$. We say a directed cycle containing v and $\overline{v}$ simultaneously is a *contradictory cycle*. There exists a contradictory cycle in G_F iff F is unsatisfiable.

Theorem 1. *If the number of clauses of F is equal to $n+2$, algorithm does not output F with probability at least $1/e^3$.*

Proof. We assume that x_1, x_2, x_3, and x_4 are chosen as x_{r1}, x_{r2}, x_{r3}, and x_{r4} respectively. Hence, F has 2 clauses, i.e., $\{x_1, x_2\}$ and $\{\overline{x_3}, \overline{x_4}\}$ at the beginning of our algorithm. In this case, G_F has 4 edges (Figure 2). Since other clauses added to F have one positive literal and one negative literal, say x_i and $\overline{x_j}$, corresponding edges are (x_j, x_i) and $(\overline{x_i}, \overline{x_j})$. Therefore, no path that consist of such edges does not connect negative and positive literals. Now we consider paths from x_1. It is easy to see that there exists no contradictory cycle if there exists no path from x_1 to x_2, x_3, nor x_4.

Let P_l be the number of paths from x_1 to x_2, x_3, or x_4 with length l ($1 \leq l \leq n-3$) when F has all clauses in the form of $(x_i, \overline{x_j})$. Then

$$P_l = 3\binom{n-4}{l-1}(l-1)! = 3\frac{(n-4)!}{(n-1-l)!}.$$

Each edge is added to F with probability $\frac{n}{n(n-1)} = \frac{1}{n-1}$., after adding n clauses to F. Let Y be the number of paths from x_1 to x_2, x_3, or x_4. Then the expectation of Y is

$$\mathrm{E}[Y] = \sum_{l=1}^{n-3}\left(\frac{1}{n-1}\right)^l \frac{3(n-4)!}{(n-3-l)!}$$

because of linearity of expectation. Then we can bound $\mathrm{E}[Y]$ as follows.

$$\begin{aligned}
\mathrm{E}[Y] &< 3\sum_{l=1}^{n-3}\left(\frac{1}{n-1}\right)^l (n-4)^{l-1} \\
&= \frac{3}{n-1}\sum_{l=1}^{n-3}\left(\frac{n-4}{n-1}\right)^{l-1} \\
&= 1 - \left(\frac{n-4}{n-1}\right)^{n-4} \\
&< 1 - \frac{1}{e^3}.
\end{aligned}$$

Therefore, the probability that there exists no path from x_1 to x_2, x_3, nor x_4 is at least $1/e^3$. □

Theorem 2. *For any positive constant ε and sufficiently large n, if the number of clauses of F is equal to $(1+\varepsilon)n+2$, algorithm outputs F with high probability.*

Proof. Throughout this proof, let $k \sim 2\log_{1+\varepsilon} n = \ln n / \ln(1+\varepsilon)$. We consider the number Y of paths from x_1 to x_3 with length k. Since each clause is added to F with probability $\frac{1+\varepsilon}{n-1}$, we have

$$\begin{aligned}
\mathrm{E}[Y] &= \left(\frac{1+\varepsilon}{n-1}\right)^k \binom{n-2}{k-1}(k-1)! \\
&\geq \frac{n^2}{n-1}\left(1-\frac{k-1}{n-1}\right)^{k-1} \\
&\geq \frac{n^2}{n-1}\left(1-\frac{k-1}{n-1}\right)^{\frac{n-1}{k-1}-1} && \text{(for sufficiently large } n\text{)} \\
&\geq \frac{n^2}{n-1}e^{-1} \to \infty.
\end{aligned}$$

Though the expected number of paths goes to infinity this does not ensure that each graph has a path with high probability. To prove this we have to show that $\mathrm{E}\left[Y^2\right]/\mathrm{E}[Y]^2 = 1 + o(1)$.

Let Y_π be a random variable where $Y_\pi = 1$ iff G_F has a path π from x_1 to x_3 with length k. We have $\mathrm{E}\left[Y^2\right] = \sum_{(\pi,\pi')} Y_\pi Y_{\pi'}$. We consider following three conditions:

1. $\pi = \pi'$
2. π and π' are edge disjoint
3. $\pi \neq \pi'$ but π and π' are not edge disjoint

Since each of them is disjoint event, we have $Y^2 = Y_1 + Y_2 + Y_3$ where Y_i is $\sum_{(\pi,\pi') \text{ satisfies condition } i} Y_\pi Y_{\pi'}$. Hence we compute $\mathrm{E}[Y_i]/\mathrm{E}[Y]^2$ for all i. We can show the following lemma easily.

Lemma 1. $\mathrm{E}[Y_1]/\mathrm{E}[Y]^2 = o(1)$ *and* $\mathrm{E}[Y_2]/\mathrm{E}[Y]^2 \leq 1$

In the following, we consider the remaining case.

We say the pair (π, π') satisfies condition 3 with parameter l and s iff π and π' has l common edges that construct s distinct sub-paths on π. We note that $1 \leq l \leq k$ and $1 \leq s \leq \min(l, k+1-l)$. For each path π, the number of paths that share l edges consisting s sub-paths on π is at most

$$\binom{l-1}{s-1}\left\{\binom{k+1-l}{s}\right\}^2 s!.$$

Hence we obtain the expectation of Y_3 as follows:

$$\mathrm{E}[Y_3] = \sum_{l=1}^{k}\sum_{s=1}^{\min(l,k+1-l)} \binom{l-1}{s-1}\left\{\binom{k+1-l}{s}\right\}^2 s!\left(\frac{1+\varepsilon}{n-1}\right)^{2k-r}.$$

By using a similar technique in [9], we can show that $\mathrm{E}[Y_3]/\mathrm{E}[Y]^2$ is $o(1)$.

Therefore, there exist a path from x_1 to x_3 almost always. Other paths also exist almost always, too. □

References

[1] T. Asano and D. P. Williamson. Improved Approximation Algorithms for MAX SAT, *J. Algorithms*, 42:173-202, 2002.

[2] B. Aspvall, M. F. Plass and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inform. Process. Lett.*, 8:121–123, 1979.

[3] V. Chvátal and B. Reed. Mick gets some (the odds are on his side). FOCS '92, pp. 620–627.

[4] A. Goerdt. A threshold for unsatisfiability. MFCS '92, pp. 264–274.

[5] U. Feige and M. X. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. ISTCS '95, pp. 182–189.

[6] M. X. Goemans and D. P. Williamson. .879-approximation algorithms for MAX CUT and MAX 2SAT. STOC '94, pp. 422–431.

[7] P. Hansen and B. Jaumard. Uniquely solvable quadratic Boolean equations. *Discrete Appl. Math.*, 12:147–154, 1985.

[8] M. Motoki. Random instance generation for MAX 3SAT. COCOON '01, pp.502–508.

[9] E. M. Wright. Large cycles in large labelled graphs. *Math. Proc. Camb. Phil. Soc.*, 78:7–17, 1975.