CS 683: Advanced Design and Analysis of Algorithms*

Lecture 6, February 1, 2008
Lecturer: John Hopcroft
Scribes: Shaomei Wu, Ethan Feldman

February 7, 2008

1 Threshold for $k - CNF$ Satisfiability

In the previous lecture, we discussed the threshold for $3 - CNF$ problem and claimed that for any $k - CNF$, there is a threshold $r_k = 2^k \ln 2$ for formula satisfiability. Today's lecture gives a more formal proof for this statement.

Theorem 1 *Given a $k - CNF$ formula with c clauses(c is a fixed number) and n variables, assume $c = r_k n$, then $r_k = 2^k \ln 2$ is a threshold for the given formula to be satisfiable.*

Proof:

(1) Pick a random assignment S for n variables.

(2) For any clause (with k literals), the probability that it is false under assignment S is $\frac{1}{2^k}$, hence

$$Prob(\text{clause is true under } S) = 1 - \frac{1}{2^k}.$$

(3) Given c clauses, the formula is true with probability

$$Prob(\text{formula is true}) = (1 - \frac{1}{2^k})^c$$

(*The clauses are considered to be generated independently.)
When $r_k = 2^k \ln 2$ and $c = r_k n$,

$$\begin{aligned} Prob(\text{formula is true}) &= (1 - \frac{1}{2^k})^c \\ &= (1 - \frac{1}{2^k})^{2^k \ln 2 \times n}. \end{aligned}$$

*The second part of this lecture on graph connectivity was re-stated and corrected in February 3's lecture, hence we reduced the amount of content in that part. Please refer to later notes for more clear proof and explanations

For large k, $(1 - \frac{1}{2^k})^{2^k} \longrightarrow e^{-1}$, hence

$$Prob(\text{formula is true}) = (1 - \frac{1}{2^k})^{2^k \ln 2 \times n} \longrightarrow e^{-(\ln 2)n} = \frac{1}{2^n}.$$

In other words, the probability that the given formula is satisfied by a random assignment of variables is $\frac{1}{2^n}$.

Let's use a set of indicator random variables $I_1, I_2, ..., I_{2^n}$ to represent the events of 1st, 2nd, ..., and $2^n t h$ assignments satisfying the given formula, respectively. Knowing that each assignment has the same probability of making the formula true, the expected values for all I_i's are same as

$$E(I_1) = E(I_2) = ... = E(I_{2^n}) = 1 \times \frac{1}{2^n} = \frac{1}{2^n}.$$

Therefore, the expected number of assignments that satisfy the given formula would be

$$E(I) = E(I_1) + E(I_2) + ... + E(I_{2^n}) = 2^n \times \frac{1}{2^n} = 1$$

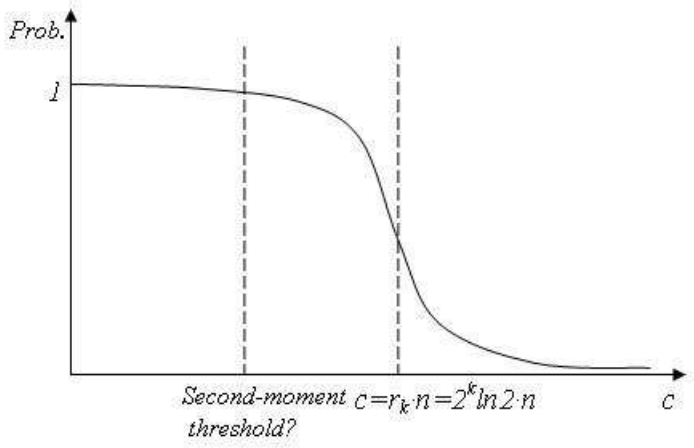


Figure 1: Probability of k-CNF being satisfied by a random assignment.

As proved in previous lectures, the number of satisfying assignements for the given formula is a monotone property. Increasing c will introduce a decrease of the probability that the formula is true under a random assignment, which means, when c is greater than $r_k \bullet n$, the expected number of satisfying assignments will be less than 1. As shown in Figure 1, $r_k = 2^k \ln 2$ is the place where the upperbound of threshold for k-CNF appears.

For $k = 3$, Chao and Franco [1] presented a heuristic algorithm based on Unit-Clause rule and proved that: when $r_k \leq \frac{2^k}{k}$, the probability of the heuristic finding a solution is approaching 1; while when $r_k \geq 2^k \ln 2$, the probability is approaching 0. For example, assuming there are 10 variables appearing in a 3-CNF formula: if the number of clauses is below 26, the heuristic algorithm can almost always find the satisfying assignment; if the number of clauses is above 55, it would be nearly impossible for the heuristic algorithm to generate a solution.

However, the situation between $r_k = \frac{2^k}{k}$ and $r_k = 2^k \ln 2$ is still unclear to us, and it is possible that there is a second-moment threshold existing in this range. One guess can be made by observing the change of solution space according to r_k: starting from $r_k = 0$, the solution space is a fully-connected graph; along with the increase of r_k, the solution space will fall apart and form a set of connected components; after passing the threshold of $r_k = 2^k \ln 2$, the solution space becomes very sparse, consists of isolated vertices. Investigating the satisfiability of k-CNF from the solution space view brings us back to the connectivity problem of graphs, which will be discussed in the following section.

2 When does a graph become connected?

Given a random $G(n, p)$, when does it become connected? There are three states that $G(n, p)$ can be:

1. $G(n, p)$ is connected;
2. $G(n, p)$ consists of isolated vertices;
3. $G(n, p)$ has no isolated vertex, but is not connected either (It is unlikely for this case to happen).

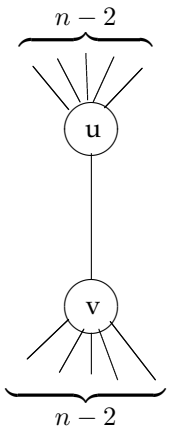
2.1 Disappearance of isolated vertices

Specific vertex is isolated with probability $(1 - p)^{n-1}$.
Threshold at $p = \frac{\log n + c}{n}$

Note: $Prob$(no isolated vertices) was wrong for this lecture because the $Prob$(isolated vertex) is not independent across all vertices. See lecture notes for 01/25/2008. If isolated vertices are the last to disappear before the graph becomes connected, then the probability that the graph is connected is $e^{-e^{-c}}$.

2.2 Disappearance of components of size 2

Given 2 vertices u, v connected by an edge.
Threshold at $p = \frac{\log n + c}{n}$.



$$
\begin{aligned}
Prob(\text{component of size 2}) &= (1-p)^{2n-4} \\
&\approx \left(1 - \frac{\log n + c}{n}\right)^{2n} \\
&= e^{-2(\log n + c)} \\
&= e^{-2logn} e^{-2c} \\
&= \frac{e^{-2c}}{n^2}
\end{aligned}
$$

Number of connected vertices $\approx n \log n$.

Note: $Prob$(no components of size 2) was wrong for this lecture because the $Prob$(component of size 2) is not independent across connected nodes. See lecture notes for 02/04/2008.

2.3 Graphs with Giant Components

$p = \frac{d}{n}, \quad d > 1$

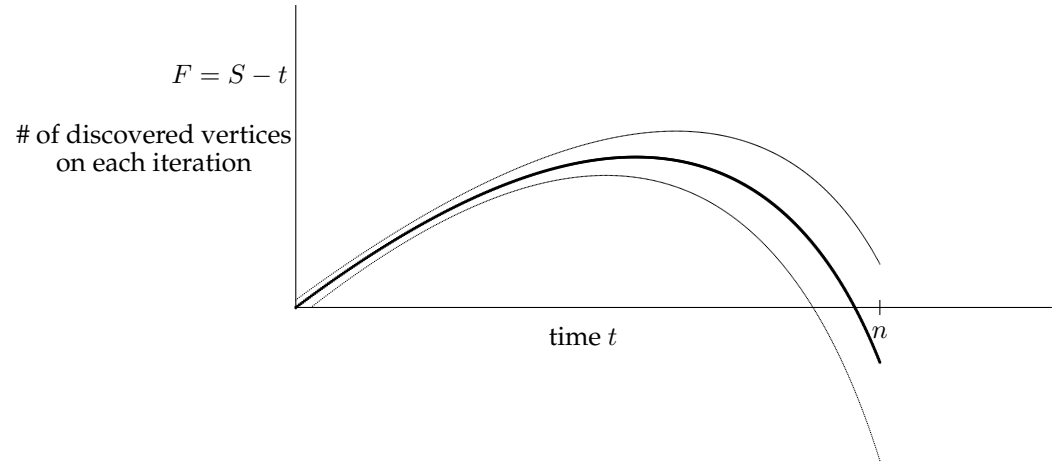
Algorithm for traversing connected components:

1. Select a vertex v
2. Mark all vertices as *undiscovered* and *unexplored*
3. Mark v as *discovered*
4. While $\exists$ *discovered* but *unexplored* vertex, select one and
5. add all adjacent vertices to *discovered* and mark the selected vertex *explored*

Let $d = 2$ for this example.

S = total # of discovered vertices.
t = iteration #.
F = expected # of vertices discovered per iteration (frontier).

Graph:



Initially, the algorithm discovers ≈ 2 vertex per iteration (expected degree of each vertex = 2, so on each iteration we expect to find 2 more vertices), but also t is incremented so initially we expect the line $\frac{F}{n}$ to have a slope ≈ 1. F is an expected value, so there will be some variation from the given curve (indicated by the lighter lines above.) If $F = 0$, the frontier is empty so we just found the last vertex of a connected component, so there exists a connected component of size t. As the graph shows, we expect F to be near 0 only during the first few iterations and then near n (the giant connected component).

Differential Equation:

$$\frac{dS}{dt} = d\left(1 - \frac{s}{n}\right)$$

Solving for S:

$$S = n\left(1 - e^{-\frac{d}{n}t}\right)$$

References

[1] M. T. Chao and J. Franco. Probabilistic analysis of a generalization of the unit clause selection heuristic for the k-satisfiability problem. *Information Sciences*, 51:289–314, 1990.