22C:253 Lecture 10

Scribe: Chen Zhang

October 16, 2002

Last week three algorithms for MAX-SAT were introduced. The final algorithm was a combination of the first two and gave us a factor-$\frac{3}{4}$ approximation algorithm. The final algorithm was:

Algorithm 3 : Toss an unbiased coin and depending on the outcome, pick algorithm 1 or algorithm 2 and run it.

For this algorithm we showed that if W is the random variable denoting the weight of the satisfied clauses then $E[W] \geq \frac{3}{4} \cdot OPT$. Alternately, a factor-$\frac{3}{4}$ approximation algorithm for MAX-SAT is given as:

> Run algorithm 1 and algorithm 2 respectively, and pick the better solution.

Claim: Let W' denote the random variable that is the weight of clauses satisfied by the alternate algorithm, then $E[W'] \geq \frac{3}{4} \cdot OPT$. **Proof:**

$$W' = \max\{W^1, W^2\} \geq \frac{W^1 + W^2}{2}$$

obtain expectation of both sides,

$$E[W'] \geq \frac{E[W^1] + E[W^2]}{2}$$

the right side is the expected weight of solution if we use algorithm 3, which $\geq \frac{3}{4} \cdot OPT$ □

Derandomization. For each of these algorithms, we have only given approximation guarantees in an expected sense, which means it does not *guarantee* that we will not get a bad solution. However, both Algorithms 1 and 2 can be derandomized, that is, we can construct equivalent deterministic algorithms. We will derandomize Algorithm 1 using the technique of *conditional probabilities* to have a guaranteed factor-$\frac{1}{2}$ approximation.

In the computation tree for MAX-SAT, a level k node is identified by the k-tuple of the truth values $(a_1, a_2, a_3, \ldots, a_k)$, where $x_1 = a_1, x_2 = a_2, \ldots, x_k = a_k$ So each leaf of the computation tree represnets a truth assignment Define the conditional expectation of a node $(a_1, a_2, a_3, \ldots, a_k)$ as

$E[W | x_1 = a_1, x_2 = a_2, \ldots, x_k = a_k]$

From the definition and previous algorithms, we obtain:

Remarks:

- The conditional expectation of the root is $E[W]$.

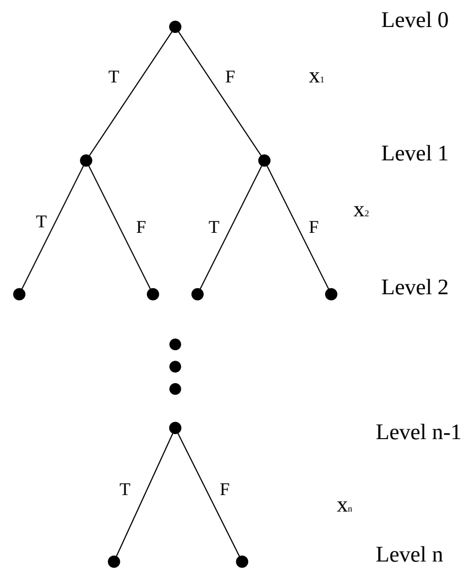

Figure 1: Computation tree for MAX-SAT

- The conditional expectation of a leaf $(a_1, a_2, a_3, \ldots, a_n)$ is the sum of the weights of satisfied clauses obtained by setting $x_i = a_i, i = 1, 2, \ldots, n$
- The conditional expectation of any node can be computed in polynomial time.

Lemma 1 *We can compute a path from root to a leaf, such that the conditional expectation at every node in the path* $\geq E[W]$.

Proof: A node $(a_1, a_2, a_3, \ldots, a_k)$ has two children $(a_1, a_2, a_3, \ldots, a_k, T)$, and $(a_1, a_2, a_3, \ldots, a_k, F)$. Since we toss a coin to decide the truth value for x_{k+1}, the conditional expectation at this node is

$$\begin{aligned} E[w|x_1 = a_1, x_2 = a_2, \ldots, x_k = a_k] &= \frac{1}{2}E[w|x_1 = a_1, x_2 = a_2, \ldots, x_k = a_k, x_{k+1} = T] \\ &+ \frac{1}{2}E[w|x_1 = a_1, x_2 = a_2, \ldots, x_k = a_k, x_{k+1} = F] \end{aligned}$$

If at node $(a_1, a_2, a_3, \ldots, a_k)$

$$E[w|x_1 = a_1, x_2 = a_2, \ldots, x_k = a_k] \geq E[W],$$

then the conditional expectation of at least one child $\geq E[W]$. This implies that if we go to the child with a higher conditional expectation until we reach a leaf, the conditional expectation at every node in the path $\geq E[W]$. □

So at the end of this path, the truth assignment represented by the leaf gives a solution which satisfies $W \geq E[W^1] \geq \frac{1}{2} \cdot OPT$

For the Algorithm 2 also, though conditional expectation at a note might not be the average of conditional expectations at its children, it is still true that the conditional expectation of at least one child $\geq E[W]$. So this works for Algorithm 2 as well.

Scheduling on unrelated parrellel machines (SUPM)
INPUT: A set J of jobs and a set M of machines, for each job $j \in J$ and machine $i \in M$, a processing time $p_{ij} \in Z^+$.
OUTPUT: An assignment of jobs to machines with minimum makespan.

Current status of this problem:

- A factor-2 approximate algorithm using LP-relaxation.
- A factor-1.5 hardness approximate algorithm (that is, if there exsits a factor-1.5 approximation algorithm, then $P = NP$).

Here is the IP for SUPM

$$\min t$$

such that

$$\begin{aligned} t &\geq \sum_{j \in J} p_{ij} \cdot x_{ij} \text{ for each } i \in M \\ \sum_{i \in M} x_{ij} &= 1 \text{ for each } j \in J \\ x_{ij} &\in \{0,1\}\ \forall i \in M, j \in J \end{aligned}$$

And the LP-relaxation for SUPM is obtained by replacing $x_{ij} \in \{0,1\}$ by $x_{ij} \geq 0,\ \forall i \in M, j \in J$

The integrality gap between this IP and LP-relaxation is huge! Let OPT be the optimal makespan, and OPT_f be the optimal makespan of the LP-relaxation. Consider such a case where there is a single job, so

$$J = \{1\}, \quad M = \{1, 2, \ldots, m\}$$
$$p_{ij} = T \text{ for all } i \in M$$

and

$$OPT = T, \quad OPT_f = \frac{T}{m}(i.e.,\ x_{i1} = \frac{1}{m} \text{ for } i \in M)$$

So the gap is m, and one might wonder if we can add constraints to the LP-relaxation to reduce this gap. Let (x, T) be a feasible solution of the IP. If $p_{ij} > T$ then job j is not assigned to machine i by the IP, and so $x_{ij} = 0$. However X_{ij} may be non-zero for the LP-relaxation. We could add a constraint C to the LP-relaxation as follows:

Constraint C: For each $i \in M, j \in J$, if $p_{ij} > T,$ then $x_{ij} = 0$

The problem here is that this is not a linear constraint. Let T^* be solution of the LP-relaxation with constraint C. Then

$$OPT_f \leq T^* \leq OPT$$

We know that the integrality gap between OPT and OPT_f is large, but the gap between OPT and T^* could be small. In fact, this is the case. So our algorithm for the problem will be:
Algorithm:
Step1. Compute (x^*, T^*) using parametric pruning.
Step2. Use rounding to go from x^* to an integral solution with makespan $T \leq 2 \cdot T^* \leq 2 \cdot OPT$.