CMPUT 675: Approximation Algorithms **Fall 2015**

Lecture 6 (Sep 17, 2015): Max SAT

Lecturer: Mohammad R. Salavatipour *Scribe: Samuel Fischer*

6.1 Max SAT (Continuation)

6.1.1 Derandomizing the "random assignment using a fair coin"-algorithm

Last lecture we saw that randomly assigning true/false values (using a fair coin) to the variables in the Max SAT problem yields a $\frac{1}{2}$-approximation. Recall that we named this algorithm Max SAT 1.

This algorithm can be derandomized using the method of conditional expectation.

Lemma 1 *Suppose we have assigned the first i boolean variables $x_1 = a_1, \ldots, x_i = a_i$. Then we can compute the expected value of solution in polynomial time.*

Proof. Observe that we can calculate the expectation of W conditioned on any partial set of assignments to the variables if a literal is false, then remove it from all the clauses in which it appears; if it is true, then ignore the clauses which contain it, as they are already satisfied. Then the conditional expectation of W is the unconditioned expectation of W in the reduced set of clauses plus the weight of the already satisfied clauses. Thus let f' be the formula over variables $x_{i+1}, \ldots, x_n$ obtained from original formula f by substituting values of $x_1, \ldots, x_i$ and simply we can compute the expected value of f'. ■

This lemma suggests the following simple algorithm.

Algorithm 1 Derandomized Max SAT 1 with conditional expectation

- Consider the variables in arbitrary order.
- Whenever we consider a variable, consider assigning $True$ and $False$ to it and compute the respective expected values, if all the not yet considered variables were assigned randomly.
- Choose that boolean value for the current variable that led to the greater expected value.
- Proceed until all variables have been regarded.

To see why this algorithm returns a solution that is as good as the expected value of the randomized algorithm Consider x_1. For each of the two assignment of $x_1 = T$ or F, we compute the expected value of the solution. If $E[W|x_1 = T] > E[W|x_1 = F]$, then we assign $x_1 = T$, otherwise, $x_1 = F$. Now set $x_1 = \tau$ as above and write the expected value of W as a weighted average of conditional expectations.

$$\begin{aligned} \mathrm{E}[W] &= \mathrm{E}[W|x_1 = T]\,\mathbf{Pr}(x_1 = T) + E[W|x_1 = F]\,\mathbf{Pr}(x_1 = F) \\ &= \frac{1}{2}\left(E[W|x_1 = T] + E[W|x_1 = F] \right) \end{aligned}$$

Then we have $E[W|x_1 = \tau] \geq \mathrm{E}[W] \geq \frac{1}{2}OPT$ So if we go through all the variables and always choose the assignment that gives a larger expected W, eventually we will find some assignment to all the variables such that the value of W is at least $\frac{OPT}{2}$.

6.1.2 Random assignment using a biased coin

As a pre-stage to a more sophisticated approximation algorithm let us consider a simplified version of the problem. Suppose that all literals of size 1 appear in positive form. Unless a variable appears in a size-1-clause in positive as well as in an other one in negative form, we can do this assumption without loss of generality.

We will see that a slightly altered form of algorithm 1 leads to better results.

Algorithm 2 (Max SAT 2) Random assignment with bias

- Randomly assign truth values to the variables as we did in algorithm 1. However, let the probability for $True$ be $p > \frac{1}{2}$.

Lemma 2 *In the results of Max SAT 2, it is* $\mathbb{P}[c_j \text{ is satisfied}] \geq \min\left(p, 1-p^2\right)$.

Proof. We perform a case-by-case analysis:

- If $|c_j| = 1$, then $\mathbb{P}[c_j \text{ is satisfied}] = p$. This is true, because the only literal in the clause appears in positive form.
- If $|c_j| \geq 2$, then $\mathbb{P}[c_j \text{ is satisfied}] = 1 - p^{\alpha}(1-p)^{\beta} \geq 1 - p^{\alpha+\beta} \geq 1 - p^2$, whereby α is the number of positive literals in c_j and β the respective number of negative literals. For the first inequality we used that $1 - p < p$ and for the second that $\alpha + \beta = |c_j| \geq 2$.

The desired statement follows directly from the above observations. ■

Claim 1 *Algorithm Max SAT 2 is a* 0.618*-approximation for the simplified maximum satisfiability problem.*

Proof. We have to choose the parameter p such that the probability that a clause is satisfied is maximized. That is, we want to find $p_{opt} := \arg\max_p \left(\min\left\{p,\ 1-p^2\right\}\right)$.

In the domain $[0,1]$ it is $f_1(p) := p$ strongly monotonously increasing and $f_2(p) := 1-p^2$ strongly monotonously decreasing. Therefore, the value p_{opt}, in which the minimum of the two functions is maximized, is at the intersection of f_1 and f_2[1].

Solving $p = 1 - p^2$ leads to $p = p_{opt} = \frac{1+\sqrt{5}}{2} \approx 0.618$. Using this result, we can calculate the expected total weight obtained using the Max SAT 2 algorithm:

$$\mathbb{E}[w] = \sum_j w_j \cdot \mathbb{P}[c_j \text{ is satisfied}] \geq p_{opt}\ opt.$$

■

However, these finding apply only to our simplified version of the Max SAT problem. What happens, if a variable x_i appears in pure and negated form in two 1-literal-clauses? That is, what if there are k and l such that $c_k = x_i$ and $c_l = \bar{x}_i$?

[1] This statement can easily be shown: Choose p_{opt} such that $f_1(p_{opt}) = f_2(p_{opt})$. For all $p < p_{opt}$ it is $f_1(p) < f_1(p_{opt})$ and for all $p > p_{opt}$ it is $f_2(p) < f_2(p_{opt})$. Thus, $\min(f_1(p), f_2(p))$ is maximized at p_{opt}.

Claim 2 *Algorithm Max SAT 2 is also a* 0.618*-approximation for the general maximum satisfiability problem.*

Proof. Suppose without loss of generality that for all k and l with $c_k = x_i$ and $c_l = \bar{x}_i$ for some i it is $w_k \geq w_l$. Let us furthermore call the set of indices of 1-literal-clauses in which a variable appears in negative form $N := \{j \ : \ \exists\, i \text{ s.t. } c_j = \{\bar{x}_i\}\}$.

In the best case, all clauses with more than 1 literal are satisfied. Then it is of course optimal to choose always those 1-literal-clauses in which the variable is positive. Hence, $\sum\limits_j w_j - \sum\limits_{j \in N} w_j \geq opt$.

With the Max SAT 2 algorithm we obtain

$$\begin{aligned} \mathbb{E}[w] &\geq p_{opt} \sum_{j \notin N} w_j \\ &= p_{opt} \left(\sum_j w_j - \sum_{j \in N} w_j \right) \\ &\geq p_{opt} \cdot opt. \end{aligned}$$

■

6.1.3 LP-rounding algorithm for Max SAT

We can find an even better approximation using LP-rounding. In preparation of this approach let us recall the definition

$$z_j := \begin{cases} 1 & \text{if clause } c_j \text{ is satisfied} \\ 0 & \text{else} \end{cases}$$

for all clauses c_j. Furthermore, let P_j the set of all positive literals in a clause c_j and N_j the set of the respective negative literals. Finally, remember that we defined for all variables x_j

$$y_j := \begin{cases} 1 & \text{if } x_j \text{ is } True \\ 0 & \text{else.} \end{cases}$$

The Max SAT problem is equivalent to maximizing $\sum\limits_j w_j z_j$ under the constraint that for all j it is $\sum\limits_{i \in P_j} y_i + \sum\limits_{i \in N_j} (1 - y_i) \geq z_i$. The constraint ensures that each satisfied clause contains a literal that is $True$.

To transform this problem into a linear programming problem, we relax the constraints for y_j and z_j such that they can take all values in $[0, 1]$. That way, we can formulate the algorithm with LP-rounding:

Algorithm 3 (Max SAT 3) LP-rounding

- Solve the linear program formulated above and let (y^*, z^*) be the obtained optimal solution.
- Set $x_i = True$ with probability y_i^*.

Theorem 1 (Goemans & Williamson '94) *Algorithm Max SAT 3 is a $\left(1 - \frac{1}{e}\right)$-approximation for the Max SAT problem.*

Proof. We will prove this theorem in multiple steps. First of all, we recall a theorem from calculus:

Lemma 3 *Let $a_1, \dots a_k$ be non-negative numbers. Their arithmetic mean is greater or equal to the geometric mean:*

$$\frac{1}{k}\sum_{i=1}^{k} a_i \geq \sqrt[k]{\prod_{i=1}^{k} a_k} \ .$$

Next, we do an observation regarding concave functions:

Lemma 4 *Suppose $f(x)$ is a function with $f(0) = 0$ and $f(1) = \alpha$, which is concave in $[0, 1]$. Then f is lower bounded by a line going through the origin and $(1, \alpha)$.*

Proof. This follows directly from the definition of concavity. In addition, figure 6.1 allows to understand the statement visually and intuitively. ■

Now we use lemmas 3 and 4 to find a lower bound for the probability that a clause is satisfied:

Lemma 5 *For any clause c_j let $k := |c_j|$. Then, $\mathbb{P}[c_j \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) z_j^*$.*

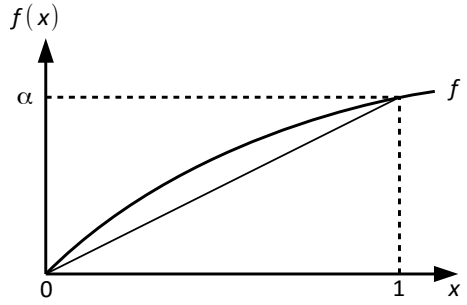

Figure 6.1: Concave function with $f(0) = 0$ and $f(1) = \alpha$. Obviously, the straight line between $(0, 0)$ and $(1, \alpha)$ is a lower bound for the function in $[0, 1]$.

Proof. The probability that a clause is satisfied is one minus the probability that no literal is true:

$$\begin{aligned}
\mathbb{P}[c_j \text{ is satisfied}] &= 1 - \prod_{i \in P_j} (1 - y_i^*) \prod_{i \in N_j} y_i^* \\
&= 1 - \left(\sqrt[k]{\prod_{i \in P_j} (1 - y_i^*)} \right)^k \left(\sqrt[k]{\prod_{i \in N_j} y_i^*} \right)^k \\
&\overset{\text{lemma 3}}{\geq} 1 - \left(\frac{1}{k} \sum_{i \in P_j} (1 - y_i^*) + \frac{1}{k} \sum_{i \in N_j} y_i^* \right)^k \\
&\overset{|P_j|+|N_j|=k}{=} 1 - \left(1 - \frac{1}{k} \left(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} (1 - y_i^*) \right) \right)^k \\
&\overset{\text{constr. for } z_j}{\geq} 1 - \left(1 - \frac{z_j^*}{k} \right)^k \\
&\geq 1 - \left(1 - \frac{1}{k} \right)^k z_j^*
\end{aligned}$$

and the lemma is proven. For the last inequality we used lemma 4: Let $g(z) := 1 - \left(1 - \frac{z_j^*}{k}\right)^k$. Clearly, it is $g(0) = 0$ and $g(1) = 1 - \left(1 - \frac{1}{k}\right)^k$. Furthermore, $g''(z) = \frac{1-k}{k} \left(1 - \frac{z}{k}\right)^{k-2} \leq 0 \ \forall \ z \in [0,1]$, if $k \geq 1$. That is, g is concave in $[0,1]$. Therefore, the line $1 - \left(1 - \frac{1}{k}\right)^k z_j^*$ is a lower bound for g in the interval $[0,1]$. ■

With lemma 5 we can easily compute the expected value of the result of algorithm Max SAT 3:

$$\begin{aligned}
\mathbb{E}[w] &= \sum_j w_j \mathbb{P}[c_j \text{ is satisfied}] \\
&\overset{\text{lemma 5}}{\geq} \sum_j w_j z_j^* \left(1 - \left(1 - \frac{1}{k} \right)^k \right) \\
&\geq opt \left(1 - \frac{1}{e} \right)
\end{aligned}$$

For the last inequality we used that $\sum_j w_j z_j^* \geq opt$ and that $\left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e}$. The latter fact holds true, because $\left(1 - \frac{1}{k}\right)^k$ is a monotonic increasing function in k and $\lim_{k \to \infty} \left(1 - \frac{1}{k}\right)^k = \frac{1}{e}$ (here: without proof). A plot of the function can be seen in figure 6.2.

Hence, algorithm Max SAT 3 is a $\left(1 - \frac{1}{e}\right)$-approximation for the Max SAT problem. ■

Note that $\left(1 - \frac{1}{e}\right) \approx 0.632$ and therefore algorithm Max SAT 3 slightly better than Max SAT 2.

Remark 1 Algorithm Max SAT 3 can be derandomized using conditional expectation.

Remark 2 If all clauses are small, this algorithm returns a better result than $\left(1 - \frac{1}{e}\right)$.

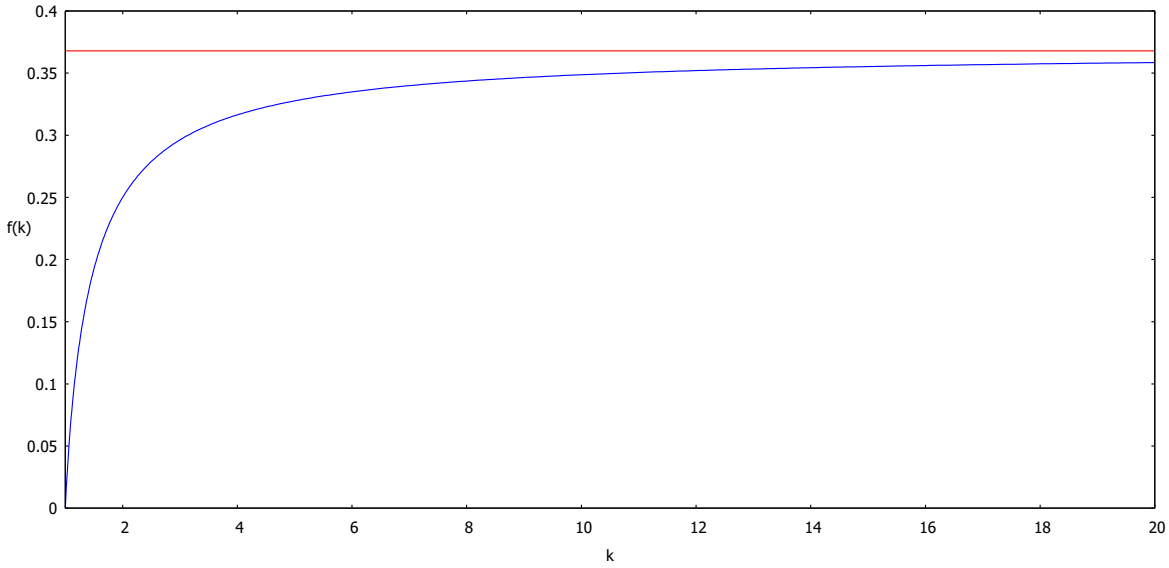

Figure 6.2: The function $f(k) := \left(1 - \frac{1}{k}\right)^k$ (blue) and the constant $\frac{1}{e}$ (red line) that f approaches.

6.1.4 Combining the algorithms for an even better approximation

We have seen that algorithm Max SAT 3 works better, if the clauses are small, while algorithm Max SAT 1 works better with large clauses. This is a fact that we can take advantage on, if we combine the two algorithms:

Algorithm 4 (Max SAT 4) Combination of unbiased random assignment and LP-rounding

- Choose randomly either algorithm Max SAT 1 or Max SAT 3 with probability $\frac{1}{2}$, respectively.
- Apply the chosen algorithm to the problem.

Theorem 2 *Algorithm Max SAT 4 is a $\frac{3}{4}$-approximation for the Max SAT problem.*

Proof. Define

$$a := \begin{cases} 1 & \text{if we chose algorithm Max SAT 1} \\ 0 & \text{else.} \end{cases}$$

Let again $k := |c_j|$ be the number of literals in c_j and $v_j := w_j z_j$ be the weight contribution of clause c_j to our solution. The expected weight contribution of a clause is given by

$$\begin{aligned}
\mathbb{E}[v_i] &= \frac{1}{2}\mathbb{E}[v_j \mid a = 0] + \frac{1}{2}\mathbb{E}[v_j \mid a = 1] \\
&= \frac{w_j}{2}\left(\mathbb{P}[c_j \text{ is satisfied} \mid a = 0] + \mathbb{P}[c_j \text{ is satisfied} \mid a = 1]\right) \\
&\geq \frac{w_j}{2}\left(\left(1 - \frac{1}{2^k}\right) + \left(1 - \left(1 - \frac{1}{k}\right)^k\right) z_j^*\right) \\
&\overset{0\leq z_i\leq 1}{\geq} \frac{w_j z_j^*}{2}\left(1 - \frac{1}{2^k} + 1 - \left(1 - \frac{1}{k}\right)^k\right) \\
&\geq \frac{3}{4} w_j z_j^*.
\end{aligned}$$

For the first inequality we used our observations regarding the Max SAT 1 algorithm from the last lecture and lemma 5. To show the last inequality we regard the function $f(k) := 2 - \frac{1}{2^k} - \left(1 - \frac{1}{k}\right)^k$. Simply calculating the respective values shows that $f(1) = f(2) = \frac{3}{2}$. Furthermore, $\frac{1}{2^k} \leq \frac{1}{8}$ for $k \geq 3$ and as we noted already above $\left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e}$. Therefore, since $2 - \frac{1}{8} - \frac{1}{e} > \frac{3}{2}$, it is $f(k) \geq \frac{3}{2}$ for all $k \geq 1$. Moreover, a simple curve sketching could show us that $f(k) \geq 2 - \frac{1}{e} \approx 1.632$ for $k \geq 5$.

The expected value for our solution is therefore

$$\begin{aligned}
\mathbb{E}[w] &= \mathbb{E}[\sum_j v_j] \\
&= \sum_j \mathbb{E}[v_j] \\
&\geq \sum_j \frac{3}{4} w_j z_j^* \\
&\geq \frac{3}{4}\, opt.
\end{aligned}$$

■

Claim 3 *Our analysis of algorithm Max SAT is tight. That is, we cannot find a better lower bound than $\frac{3}{4}opt$ for the expected value of our result.*

Proof. Suppose we are given the CNF

$$(x_1 \vee x_2) \wedge (x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee x_2) \wedge (\overline{x}_1 \vee \overline{x}_2)$$

and the weight of all clauses is 1. Clearly, at most 3 of the clauses can be satisfied, i.e. $opt = 3$. On the other hand, choosing $y_i = \frac{1}{2}$ and $z_i = 1$ for $i \in \{1, 2\}$ satisfies all LP constraints and leads to an LP solution with total weight 4. Hence, the integrality gap is at least $\frac{4}{3}$ which is equivalent to the statement we want to show. ■

The best approximation for Max SAT that is currently known uses semi-definite programming to get to a 0.7968-approximation. For the special case Max E 3 SAT, in which each clause has exactly 3 literals, algorithm Max SAT 1 returns a $\frac{7}{8}$-approximation. [H01] showed that a better approximation algorithm cannot be found for this problem, unless $P = NP$. Since Max E 3 SAT is a special case of Max SAT, $\frac{7}{8}$ is also an upper bound for Max SAT approximations.

6.2 The uncapacitated facility location problem

A prominent and well-studied problem in computer science is the facility location problem. It addresses the question where to open facilities in order to serve the demand in an optimal way (see figure 6.3).

Uncapacitated facility location problem:

- Input:
  - A set D of demand points (clients) along with their respective demands d_j for all $j \in D$. The demand can be thought of as a measure of how many clients a demand point holds. Note, however, that we use the word "client" as synonym for "demand point", rather than as a unit of demand.
  - A set F of facilities among with their respective opening costs f_i for all $i \in F$.
  - The distance c_{ij} from any client $j \in D$ to any facility $i \in F$. The distances must be given as costs to get from j to i and be metric, that is $\forall\ i, j, k \in F \times D$ it is $c_{ij} + c_{jk} \geq c_{ik}$ and $c_{ij} \geq 0$.
- Goal: Find a subset $F' \subseteq F$ and a mapping $M : D \to F',\ j \mapsto M(j)$ that assigns each client to exactly one facility in F' such that the total costs $\sum_{i \in F'} f_i + \sum_{j \in D} c_{M(j)j}$ are minimized. Note: M will map each client to its closest open facility.

Special cases of the problem are given if $F = D$ or $d_j = 1\ \forall\ j \in D$.

6.2.1 Formulation as integer program

In order to find an approximate solution to the facility location problem using linear programming, we have to formulate the problem as a linear program first. Let us set

$$x_{ij} := \begin{cases} 1 & \text{if client } j \text{ is served at facility } i \\ 0 & \text{else} \end{cases}$$

and

$$y_i := \begin{cases} 1 & \text{if facility } i \text{ is open} \\ 0 & \text{else.} \end{cases}$$

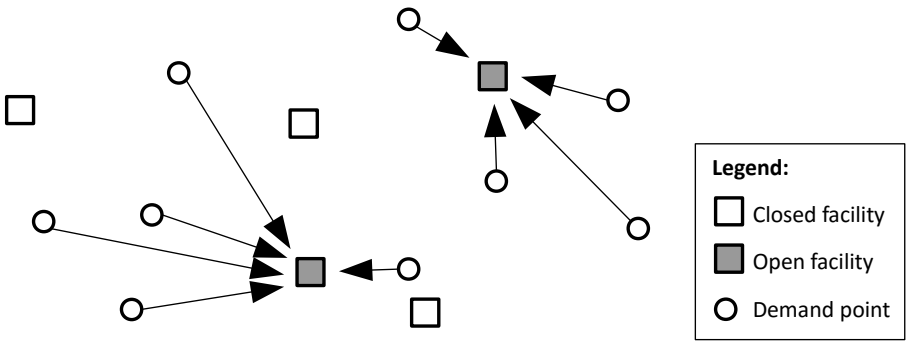

Figure 6.3: Visualization of the facility location problem. A subset of the facilities (squares) has to be opened such that all clients (small circles) can be assigned to respectively one opened facility (grey square) with minimized costs for open facilities and travel to facilities.

We want to minimize the sum of the total opening costs and the costs clients have to pay to get to their respective facilities. That is, we desire to find

$$\min \left\{ \sum_{i \in F} f_i y_i + \sum_{i \in F} \sum_{j \in D} c_{ij} x_{ij} \right\}.$$

Thereby, each client must be served by at least one facility. That is,

$$\forall\ j \in D : \ \sum_{i \in F} x_{ij} \geq 1$$

Furthermore, a client j can only be served at facility i is open, that is,

$$\forall\ i \in F,\ j \in D : \ x_{ij} \leq y_i.$$

As we switch from an integer- to a linear program, we will allow all positive values for all x_{ij} and y_j:

$$\forall\ i \in F,\ j \in D : \ x_{ij}, y_j \geq 0.$$