COMPSCI 632: Approximation Algorithms	October 4, 2017
Lecture 11	
Lecturer: Debmalya Panigrahi	*Scribe: Kevin Sun*

1 Overview

Last lecture, we obtained a $3/4$-approximation for MAX-SAT and saw that the corresponding LP has an integrality gap of $4/3$, so we can't expect to do better without a new idea. In this lecture, we modify the probabilities according to which we round to obtain an alternative $3/4$-approximation.

We also use deterministic rounding to obtain a 2-approximation for the Generalized Assignment Problem (GAP). The algorithm is based off a well-known result concerning weighted bipartite matchings.

2 MAX-SAT, revisited

Recall that in the MAX-SAT problem, we are given a collection of m clauses $\{C_j\}_{j=1}^m$ over Boolean variables $\{x_1,\ldots,x_n\}$, and we want to find a truth assignment to the Boolean variables that maximizes the total number of satisfied clauses. As we saw last lecture, the corresponding LP is the following:

$$\begin{aligned} &\max \sum_{j=1}^{m} z_j \\ \forall j \in [m] \quad &\sum_{i\in P_j} y_i + \sum_{i\in N_j} (1-y_i) \geq z_j \\ \forall i \in [n] \quad &0 \leq y_i \leq 1 \\ \forall j \in [m] \quad &0 \leq z_j \leq 1, \end{aligned}$$

where P_j and N_j denote the set of positive and negative literals in C_j, respectively. Our algorithm from last lecture, which we now call Alg-1, was the following: obtain an optimal (fractional) solution (y^*, z^*) to the above LP and set each x_i to true with probability y_i^*. We noticed

$$\Pr(C_j \text{ is satisfied by Alg-1}) = 1 - \prod_{i\in P_j}(1-y_i^*)\prod_{i\in N_j} y_i^*$$

and applied the AM-GM inequality to show Alg-1 is a randomized $(1-\frac{1}{e})$-approximation.

However, we notice that since our analysis is dealing with a large product, perhaps converting from an arithmetic scale to a logarithmic scale would be beneficial. So today, we look at algorithm Alg-2: obtain an optimal (fractional) solution (y^*, z^*) to the above LP, fix some function f such that

$$1 - 4^{-x} \leq f(x) \leq 4^{x-1} \quad \forall x \in [0,1],$$

and set x_i to true with probability $f(y_i^*)$.

Theorem 1. *Alg-2 is a randomized* $3/4$*-approximation for MAX-SAT.*

Proof. The analysis of Alg-2 closely resembles that of Alg-1. We notice that for each clause C_j,

$$\begin{aligned}\Pr(C_j \text{ is satisfied by Alg-2}) &= 1 - \prod_{i \in P_j}(1 - f(y_i^*)) \prod_{i \in N_j} f(y_i^*) \\ &\geq 1 - \prod_{i \in P_j} 4^{-y_i^*} \prod_{i \in N_j} 4^{y_i^* - 1} \\ &= 1 - 4^{-\sum_{i \in P_j} y_i^* - \sum_{i \in N_j}(1 - y_i^*)} \\ &\geq 1 - 4^{-z_j^*},\end{aligned}$$

where the last inequality is due to the LP constraint.

Again, $f(z_j^*) = 1 - 4^{-z_j^*}$ is concave, so its graph always lies above the line connecting the points $(0,0)$ and $(1, 1 - 1/4)$. Thus, we have

$$\Pr(C_j \text{ is satisfied by Alg-2}) \geq (1 - \frac{1}{4}) z_j^*.$$

By summing over all C_j and applying linearity of expectation, we can conclude the statement of the theorem. □

3 Deterministic Rounding

Recall our 2-approximation from Lecture 1 for vertex cover: solve the relaxed LP to obtain a fractional optimal solution x^*, and pick vertex v if and only if $x_v^* \geq 1/2$. Our new tool is an integral linear program: one whose extreme points (the corners of the constraint polytope) are integral. With any objective, optimality is achieved achieved at an extreme point, so these LPs have no integrality gap.

Our problem is the *generalized assignment problem* (GAP). In GAP, we are given n jobs, m machines, and a time bound λ. Job j has a processing time of p_{ij} on machine i, and assigning job j to machine i incurs a cost of c_{ij}. We want to find a minimum-cost assignment of the jobs to the machines whose makespan is at most λ. (The *makespan* of an assignment is the completion time of the job processed last.)

Now let's look at the integer program corresponding to GAP, which we call IP-1:

$$\begin{aligned}\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \forall j \in [n] \quad \sum_{i=1}^{m} x_{ij} = 1 \\ \forall i \in [m] \quad \sum_{j=1}^{n} p_{ij} x_{ij} \leq \lambda \\ \forall i \in [m], j \in [n] \quad x_{ij} \in \{0,1\},\end{aligned}$$

where $x_{ij} = 1$ indicates assigning job j to machine i. Relaxing the integrality constraint of IP-1 to

$$\forall i \in [m], j \in [n] \quad x_{ij} \geq 0$$

gives us a linear program, which we call LP-0. However, we note that LP-0 may perform quite poorly as a proxy model: consider a single large job j with $c_{ij} = 1$ and $p_{ij} = m$ on every machine i. Any integral

assignment has makespan equal to m, but fractionally, this job can be split evenly over the m machines to achieve a makespan of 1.

To rectify this issue, we strengthen LP-0 by adding the additional constraints

$$\forall i \in [m], j \in [n] \quad x_{ij} = 0 \quad \text{if } p_{ij} > \lambda, \tag{1}$$

which are clearly satisfied by any feasible integer assignment, and call the resulting linear program LP-1.

Theorem 2. *Given a feasible (fractional) solution x^* to LP-1 with cost C, we can round x^* to an (integer) assignment with cost at most C and makespan at most 2λ.*

Our rounding scheme will depend on a well-known result from matching theory. Let $G = (V, E)$ be a bipartite graph where $V = A \cup B$, $A \cap B = \emptyset$, and $|A| \leq |B|$. A *complete matching* is a subset M of E such that for each $u \in A$, there is exactly one edge in M incident to u, and for each $v \in B$, there is at most one edge in M incident to v.

Now suppose each edge (u, v) has an associated cost d_{uv}, and we want to find a complete matching with minimum total cost. This problem can be formulated as an integer program, IP-2:

$$\begin{aligned} \min \sum_{(u,v)\in E} d_{uv} y_{uv} \\ \forall u \in A \quad \sum_{v \in B} y_{uv} = 1 \\ \forall v \in B \quad \sum_{u \in A} y_{uv} \leq 1 \\ \forall (u,v) \in E \quad y_{uv} \in \{0,1\}, \end{aligned}$$

where $y_{uv} = 1$ indicates that the edge (u, v) is in the matching. (Note the similarities between IP-1 and IP-2.) As we did for IP-1, we relax the binary constraints to the non-negativity constraints

$$\forall (u,v) \in E \quad y_{uv} \geq 0,$$

to obtain a linear program LP-2. For this problem, the following fact is well-known; note its similarity with the statement of Theorem 2.

Fact 3. *Given a feasible (fractional) solution to LP-2 with cost D, we can find, in polynomial time, a feasible integer solution with cost at most D.*

Using this fact on a specially-constructed bipartite graph, we will be able to find our integer solution to GAP with makespan at most twice the given time bound.

Proof of Theorem 2. Start by solving LP-1 to obtain a fractional assignment x^* with total cost C. Our goal is to round x^* to an integer assignment so that the cost does not increase, and the makespan increases by at most a factor of 2.

Now, consider the following for each machine i: the (fractional) assignment x^* assigns $\sum_{j=1}^{n} x^*_{ij}$ jobs to machine i. To bound the makespan of our integer solution, let $k_i = \lceil \sum_{j=1}^{n} x^*_{ij} \rceil$. Machine i will be split into k_i copies of itself, and each copy of machine i will be assigned at most one job.

With this idea, let's construct a bipartite graph $G = (V, E)$ with $V = J \cup S$, and $J \cap S = \emptyset$, where

$$S = \{(i,k) : i \in [m], k \in [k_i]\},$$

is a set containing k_i copies of every machine i. Note that $|J| \leq |S|$ because x^* completely assigns each job to some subset of machines.

To create E, one natural idea is to simply include the edge $(j,(i,k))$ with cost c_{ij} whenever $x_{ij} > 0$. If we solve LP-2 and apply Fact 3 to the resulting graph, the cost would be at most C and the number of jobs assigned to each machine copy would be at most 1, but we have no control on the makespan of the integer solution.

So instead, let's construct E with the following "bin-packing" procedure, focusing on one machine i at a time. We want to place its n fractional job pieces x^*_{ij} into its k_i copies, where each copy acts as a bin with capacity 1. Without loss of generality, assume $p_{i1} \geq p_{i2} \geq \cdots \geq p_{in}$, and pack the jobs into the copies of i in this order. Starting with $k = 1$, we fill bin (i,k) as much as possible (by possibly splitting a job piece) before placing anything in bin $(i,k+1)$.

After this packing is done, we can form our edge set E: an edge $(j,(i,k))$ is in E with weight c_{ij} if and only if this bin-packing procedure places a positive amount of x^*_{ij} into the machine copy (i,k). Denote this positive amount by $y_{j,(i,k)}$, and note the following:

$$\forall i \in [m], j \in [n] \quad x^*_{ij} = \sum_{k=1}^{k_i} y_{j,(i,k)}. \tag{2}$$

Furthermore, the $y_{j,(i,k)}$ variables form a fractional solution to LP-2 on the bipartite graph $G = (J \cup S, E)$ with cost C. So now, we apply Fact 3 to obtain a complete matching $\hat{y}$ with total cost at most C. We will show that if we assign each machine i the jobs that its k_i copies collectively receive by $\hat{y}$, then the resulting makespan is at most 2λ.

Now we fix a machine i and let $p(i,k)$ denote the integer load by $\hat{y}$ on (i,k), the k-th copy of machine i. We claim that

$$\sum_{k=2}^{k_i} p(i,k) \leq \lambda. \tag{3}$$

We only start filling bin $(i,k+1)$ after completely filling bin (i,k), so

$$\forall k \in [k_i - 1] \quad \sum_{j \in [n]} y_{j,(i,k)} = 1,$$

and since we our jobs are sorted by decreasing p_{ij}, we also have

$$\forall k \in \{2, \ldots, k_i\} \quad p(i,k) \leq \sum_{j \in [n]} y_{j,(i,k-1)} p_{ij}.$$

Summing this over k and applying (2) gives us

$$\sum_{k=2}^{k_i} p(i,k) \leq \sum_{k=1}^{k_i} \sum_{j \in [n]} y_{j,(i,k)} p_{ij} = \sum_{j \in [n]} x^*_{ij} p_{ij},$$

and since x^*_{ij} is feasible for LP-1, this last term is at most λ, so we've shown (3) is true.

Finally, constraint (1) of LP-1 guarantees $p(i,1) \leq \lambda$. Thus, the job assigned by $\hat{y}$ to $(i,1)$ adds at most λ to the the completion time of machine i, so together with (3), the total makespan of $\hat{y}$ is at most 2λ, as desired. □

4 Summary

In this lecture, we saw an alternative $3/4$-approximation for MAX-SAT based off modifying the probabilities according to which we round.

We also used deterministic rounding to approximate the generalized assignment problem: for each input with a feasible solution, our algorithm finds a solution with cost no higher than the optimum and violates makespan feasibility by at most a factor of 2.