CSC 2421: Approximation Algorithms	3 March 2005

Lecture 8: LP-rounding

Instructor: Allan Borodin — Scribe: Chaitanya Mishra

In this lecture, we look at LP-rounding based approaches to solving the set cover, maximum satisfiability and uniform metric labeling problems. What is common to these approaches is that they all lead to randomized algorithms. The algorithms for the set cover and maximum satisfiability problems illustrate a common randomized rounding scheme which uses the optimal fractional solution to the relaxation of the corresponding integer program as an array of probabilities. The solution to the integer program is then computed by flipping coins biased as per these probabilities. The algorithm for the third problem uses a different kind of randomized rounding scheme which uses more information from the problem.

The algorithms for set cover and maximum satisfiability may be found in chapters 14 and 16 from the textbook [1]. The algorithm for the uniform metric labeling problem is taken from a paper [2] by Jon M. Kleinberg and Éva Tardos.

1 Set Cover

The set cover problem is: Given a universe U of n elements, a collection of subsets of U, $\mathcal{C} = \{S_1, \ldots, S_k\}$ and a cost function $c : \mathcal{C} \rightarrow \mathbf{Q}^+$, find a minimum cost subcollection of $\mathcal{C}$ that covers all elements of U. This can be formulated as an integer program by defining a variable x_S for each set $S \in \mathcal{C}$ which has value 1 if S is picked as a part of the cover and 0 otherwise. The IP formulation is then:

$$Min \sum_{S \in \mathcal{C}} c(S) x_S \qquad s.t. \quad \begin{cases} \sum_{S: u \in S} x_S \geq 1 & \forall u \in U \\ x_S \in \{0,1\} & \forall S \in \mathcal{C} \end{cases}$$

The LP-relaxation of this is done by changing the condition on x_S to $x_S \geq 0$. Let the solution to this problem be: $X^* = \{x^*_{S_1}, \ldots, x^*_{S_k}\}$ with cost OPT_f. We can consider these fractional $x^*_{S_1}$ as probabilities and round the fractional values to get a solution $\overline{X}_{S_i}$ and a cover $\overline{C}$ such that $\overline{X}_{S_i} = 1$ with probability $x^*_{S_1}$ and $\overline{C} = \{S_i | \overline{X}_{S_i} = 1\}$. The expected cost of $\overline{C}$ is:

$$\mathbf{E}[c(\overline{C})] = \sum_{S \in \mathcal{C}} c(S) \mathbf{E}[\overline{x}_S] = \sum_{S \in \mathcal{S}} c(s) x^*_S = OPT_f$$

The problem that still remains is that $\overline{C}$ is not necessarily a cover. However, we can show a lower bound on the probability that a certain element is covered by $\overline{C}$. Suppose u is an element of k' sets of $\mathcal{C}$. Let the probabilities of these sets be $p_1, \ldots, p_{k'}$. The probability that u is covered by $\overline{C}$ is minimized when $p_i = 1/k'$. Thus

$$\mathbf{Pr}[u \text{ is covered by } C] = (1-p_1)\cdots(1-p_{k'}) \geq 1-(1-\frac{1}{k'})^{k'} \geq 1-\frac{1}{e}$$

Now, if we run the algorithm $\ln 4n$ times where $n = |U|$ and take the union $\mathcal{C}'$ of the covers computed by each iteration we have

$$\mathbf{Pr}[u \text{ is not covered by } \mathcal{C}' \leq (\frac{1}{e})^{\ln 4n} \leq \frac{1}{4n}$$

Since the coins are flipped independently, we can sum over all elements $u \in U$ to get the probability that $\mathcal{C}'$ is not a cover.

$$\mathbf{Pr}[\mathcal{C}' \text{ is not a valid set cover}] \leq n \,.\, \frac{1}{4n} = \frac{1}{4} \quad (1)$$

Now, since the expected cost of this cover $\mathbf{E}[c(\mathcal{C}')] \leq \ln 4n \,.\, OPT_f$, we use Markov's inequality to get:

$$\mathbf{Pr}[c(\mathcal{C}') \geq 4\ln 4n \,.\, OPT_f] \leq \frac{1}{4} \quad (2)$$

Combining (1) and (2),

$$\mathbf{Pr}[\mathcal{C}' \text{ is a valid set cover and has cost} \leq 4\ln 4n \,.\, OPT_f] \geq \frac{1}{2}$$

This tells us that the expected number of runs of the above procedure needed to get a cover satisying these conditions is 2.

2 Maximum Satisfiability (MAX-SAT)

The MAX-SAT problem is: Given a CNF formula $f = \bigwedge_{c \in \mathcal{C}} c$ on boolean variables $x_1, \ldots, x_n$, find a truth assignment to the variables that maximizes the total number of satisfied clauses. Each of the clauses c is a disjunction of literals. The problem can also be extended to a weighted version in which each clauses c has an associated weight w_c, and the goal is to maximize the total weight of the satisfied clauses. In what follows, we present two randomized rounding based approaches to solving this problem. We then show how these approaches can be derandomized.

2.1 Naive approach

The naive approach works by setting each boolean variables value independently as either true or false with probability 1/2 and using the resulting truth assignment τ. For a disjunctive clause c_i with k_i literals, the probability that it is satisfied by τ (i.e none of the literals is true) is $\alpha_k = 1-2^{-k_i}$ which is greater than 1/2 for any clause with more than one literal. Therefore we have

$$\mathbf{E}[\#\text{ of clauses satisfied by }\tau] = \sum_{c_i \in \mathcal{C}} (1 - \frac{1}{2^{k_i}}) \geq \frac{1}{2}(\#\text{ of clauses}) \geq \frac{1}{2} OPT$$

Note that this approach does not involve the use of linear programming at any stage. The analysis used above can be used in the weighted case as well.

2.2 LP-rounding approach

The MAX-SAT problem can be formulated as an integer program as well. The truth assignment to a boolean variable x_i can be encoded by a variable y_i such that $y_i = 1$ (0) when $x_i =$ True (False). Similarly, the satisfiability of a clause c under the truth assignment is encoded by the variable z_c. In addition, for each clause c, let S_c^+ and S_c^- denote the set of boolean variables occuring nonnegated and negated respectively in c. With this notation in place, we get the integer program (for the weighted case) as

$$Max \sum_{c \in \mathcal{C}} w_c z_c \quad s.t. \begin{cases} \sum_{i \in S_c^+} y_i + \sum_{i \in S_c^-} (1 - y_i) \geq z_c & \forall c \in \mathcal{C} \\ z_c \in \{0, 1\} & \forall c \in \mathcal{C} \\ y_i \in \{0, 1\} & \forall i \end{cases}$$

The LP-relaxation of this integer program involves changing the last 2 conditions to a fractional form

$$1 \geq z_c \geq 0 \;\; \forall c \in \mathcal{C} \quad 1 \geq y_i \geq 0 \;\; \forall i$$

The first step of the algorithm is to solve the LP to get solutions $\mathbf{y}^*$ and $\mathbf{z}^*$. Then, we set x_i to true with probability y_i^* for $1 \leq i \leq n$. This rounding scheme is on the same lines as the one used previously for set cover. The resulting truth assignment τ is output as the solution.

To get a bound on this algorithm, let us consider an arbitrary clause c. Without loss of generality we may assume that all literals in c appear nonnegated. This is because, we could substitute a nonnegated literal in place of every negated literal in c, and make the corresponding change visible in all other clauses without affecting the problem. Assume that $c = (x_1 \vee \cdots \vee x_k)$. Now we know that $\sum_{i=1}^k y_i^* \geq z_c^*$. Using this and the fact that c is not satisfied iff τ assigns the value false to all literals in c we get:

$$\mathbf{Pr}[c\text{ is satisfied}] = 1 - \prod_{i=1}^{k} (1 - y_i^*)] \geq 1 - (1 - \frac{z_c^*}{k})^k$$

The inequality used is simply that the arithmetic mean of a set of nonnegative numbers is at least as large as their geometric mean

The function $g(z) = 1 - (1 - z/k)^k$ is a concave function with $g(0) = 0$ and $g(1) = 1 - (1 - 1/k)^k$. Let us call $1 - (1 - 1/k)^k$ as β_k. Within any range $[a, b]$, the value of the concave function at a point z such that $a \leq z \leq b$ is at least as large as the value of z on the line joining a and b. Now since, $\beta_k \geq 1 - 1/e$, we get the result that for $0 \leq z \leq 1, g(z) \geq z(1 - 1/e)$. Using this inequality

$$\mathbf{E}[\text{\# of clauses satisfied}] \geq (1 - \frac{1}{e}) \sum z_c^* = (1 - \frac{1}{e}) OPT_f \geq (1 - \frac{1}{e}) OPT$$

Thus, this is a $1 - 1/e$ factor algorithm for MAX-SAT.

The naive method and the LP-rounding based methods can together be combined to give a 3/4 factor algorithm. The new algorithm is very simple. Use an unbiased coin to decide which of the two algorithms to run. And then, run it. For the analysis, let us call the result of the coin flip is b, and assume that we run the naive algorithm when $b = 0$ and the LP-rounding based algorithm when $b = 1$. Now, the expectation of satisfiability of a clause c of size k is

$$\mathbf{E}[c \text{ is satisfied}] = \frac{1}{2}\mathbf{E}[c \text{ is satisfied}|b = 0] + \frac{1}{2}\mathbf{E}[c \text{ is satisfied}|b = 1] = \frac{1}{2}(\alpha_k + \beta_k z_c^*) \geq \frac{1}{2}(\alpha_k + \beta_k) z_c^*$$

Now, $\alpha_1 + \beta_1 = \alpha_2 + \beta_2 = 3/2$, and for $k \geq 3, \alpha_k + \beta_k \geq 7/8$. Thus, the probability that a clause is satisified by the truth assignment given by this algorithm is $\geq 3/4$. The bound for the algorithm follows in a straightforward manner.

Interestingly, both these algorithms can be derandomized by considering a self-reducibility tree for the formula f. Each internal node in such a tree at level i corresponds to a truth assignment to the variables $x_1, \ldots, x_i$. Every node is labelled by the conditional expectation of the number of satisfied clauses given the corresponding truth assignment at the node i.e $\mathbf{E}[\text{\# of clauses}|x_1 = a_1, \ldots, x_i = a_i]$. Given such a labelling, the derandomized version of the algorithm moves from the root to the leaves of the tree, always taking the child with a higher expectation. Since the conditional expectation of a node is a convex combination of the conditional expectations of its two children, the algorithm keeps improving its solution as it moves down the tree.

The derandomized version of the 3/4 factor algorithm follows likewise. All we need to do is to run the derandomized versions of both the algorithms, and output the better of the two assignments.

3 Uniform Metric Labeling

The metric labeling problem [2] is a general classification framework that encapsulates a number of standard classification problems. The goal of this framework is to capture classification problems in which pairwise relationships exist among the objects to be classified.

To state the problem more precisely, consider a set P of n objects that we wish to classify, and a set L of k possible labels. A labeling of P over L is a function $f : P \rightarrow L$. There is a cost of labeling an object which is captured by a non-negative cost function $c(p, a)$ for assigning label a to object p. In addition, we know which objects are *related* and store this information as a graph G over the vertex set P, with the edges connecting related objects. Associated with each edge e

is a non-negative weight w_e corresponding to the *strength of the relation*. Similarly, there is also a notion of similarity amongst labels captured as a distance $d(.,.)$ on the set L of labels. If a labels a and b are assigned to objects p and q and e is the edge between p and q, then we pay a cost of $w_e d(a,b)$ in addition to the cost of assigning the labels to the objects.

Given this framework, the total cost of a labeling f is

$$\sum_{p\in P} c(p, f(p)) + \sum_{e=(p,q)\in E} w_e d(f(p), f(q))$$

The labeling problem is to find the labeling of minimum total cost. The metric labeling problem is the labeling problem with the additional assumption that $d(.,.)$ is a metric on the set L of labels. The uniform metric labeling problem is the even more constrained version in which the metric is uniform i.e for labels a and b, $d(a,b) = 0$ if $a = b$ and 1 otherwise.

The uniform metric labeling problem can be expressed by an integer program. Define nonnegative variables x_{pa} for each object p and label a with the condition that $x_{pa} = 1$ iff $f(p) = a$, and 0 otherwise. The label cost of an object is then $\sum_{a\in L} c(p,a)x_{pa}$. Since the labeling is uniform, the distance between labels assigned to 2 objects p and q is $\frac{1}{2}\sum_{a\in L}|x_{pa} - x_{qa}|$. This can be captured in an integer program by using a variable z_{ea} to express the absolute value $|x_{pa} - x_{qa}|$ for $a \in L$ and $e = (p,q)$. With this terminology is place, the integer program is

$$Min \sum_{e\in E} w_e z_e + \sum_{p\in P, a\in L} c(p,a)x_{pa} \quad s.t \begin{cases} \sum_{a\in L} x_{pa} = 1 & \forall p \in P \\ z_e = \frac{1}{2}\sum_{a\in L} z_{ea} & \forall e \in E \\ z_{ea} \geq x_{pa} - x_{qa} & \forall e = (p,q), a \in L \\ z_{ea} \geq x_{qa} - x_{pa} & \forall e = (p,q), a \in L \\ x_{pa} \in 0,1 & \forall p \in P, a \in L \end{cases}$$

The linear programming relaxation of the above integer program is created by changing the last set of conditions (integrality constraints) to $x_{pa} \geq 0$. In what follows, the solution to the linear program is called x^* with x^*_{pa} being the fractional labeling corresponding to an object p and label a. Similarly z^*_e and z^*_{ea} denote the parts of the solution corresponding to the z_e and z_{ea} variables.

A simple randomized rounding scheme using the fractional labels as probabilities does poorly since it involves assigning labels independently to objects and hence ignores the pairwise relationship constraints encapsulated in the graph. In their paper [2], Kleinberg and Tardos introduce a randomized rounding scheme which proceeds in phases and gives a bound of 2 with respect to the optimal solution. The algorithm proceeds in phases. Initially all vertices are unlabeled. In each phase, we pick a label a uniformly from the set of labels, and a real number $\alpha \in [0,1]$. For each object p that is currently unlabeled, we give it the label a if $\alpha \leq x^*_{pa}$. The phases are iterated until each vertex has a label assignment. The following lemmas let us prove the bound on this procedure.

Lemma 1. *Consider a particular phase, and an object p that is yet to be assigned a label at the start of the phase. The probability that p is assigned label a in this phase is x^*_{pa}/k and the probability that p is assigned any label in this phase is $1/k$. Also, the probability that p is assigned label a at some point in the run of the algorithm is x^*_{pa}.*

Proof. In each phase, while object p has no label assignment, the probability that it is assigned to label a is $(1/k)x^*_{pa}$, which is proportional to x^*_{pa}. Therefore over the whole run of the algorithm, the probability that object p is assigned label a is x^*_{pa}. It also follows that the probability that an object p is assigned $a$ny label in a phase is the $1/k$ (by summing probabilities over all labels).

We say that an edge $e = (p, q)$ is *separated* by a phase if before the phase both p and q were unassigned, and exactly one of p and q is assigned in this phase. An edge $e = (p, q)$is *split* by the process if p and q receive different labels. Note that if e is split, then e must have been separated in some phase, but the converse does not hold. With this terminology in place, we state the other lemma

Lemma 2. *For an edge $e = (p, q)$, the probability that the edge is separated by a given phase, assuming neither end is assigned before the phase, is $2z_e{}^*/k = \sum_{a\in L}|x^*_{pa} - x^*_{qa}|/k$. The probability that the two objects p and q are split by the process is at most $2z^*_e$.*

Proof. Consider a particular phase, and assume that objects p and q are both unassigned at the beginning of the phase. If label a is selected, then the probability that exactly one of p and q is assigned the label is $|x^*_{pa} - x^*_{qa}|$. Summing over probabilities, the probability that $e = (p, q)$ is separated by this phase is

$$\frac{1}{k}\sum_{a\in L}|x^*_{pa} - x^*_{qa}| = \frac{\sum_{a\in L} z^*_{ea}}{k} = \frac{2z^*_e}{k}$$

If p and q are both assigned a label in this phase, then the edge (p, q) is not split by the process. The conditional probability that p and q are separated in this phase, given that at least one of p and q is assigned by this phase is

$$\frac{\mathbf{Pr}[p, q \text{ are separated by this phase}]}{\mathbf{Pr}[\text{at least one of } p, q \text{ is assigned a label by this phase}]} \le \frac{2z^*_e/k}{1/k} = 2z^*_e$$

The inequality comes from the fact that the denominator is bound by the probability that object p is assigned, which is $1/k$ as shown in Lemma 1. Since this conditional probability holds for any phase that starts with both p and q unassigned, the probability that an edge $e = (p, q)$ is split by the algorithm is at most $2z^*_e$.

Lemma 1 tells us that the probability that object p is assigned label a is x^*_{pa}, and hence the expected contribution of p to the objective function is $\sum_{a\in L} c_{pa}x^*_{pa}$. Lemma 2 tells us that the probability that an edge $e = (p, q)$ is split is bound by $2z^*_e$ and so the expected contribution of the edge to the cost is $2w_e z^*_e$. Hence, the expected cost of a labeling x made by the algorithm is

$$\mathbf{E}[\sum_{e\in E} w_e z_e + \sum_{p\in P, a\in L} c(p, a)x_{pa}] \le 2\sum_{e\in E} w_e z^*_e + \sum_{p\in P, a\in L} c(p, a)x^*_{pa} \le 2 \,.\, OPT_f$$

Hence, this algorithm is a 2-approximation algorithm for the uniform metric labeling problem.

References

[1] Vijay V. Vazirani. *Approximation algorithms.* Springer-Verlag New York, Inc., 2001.

[2] Jon M. Kleinberg and Éva Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields. *J. ACM*, 49(5):616–639, 2002.