Lecturer: Hung Q. Ngo
Last update: March 10, 2005

Approximation algorithms based on LP relaxation

There are two fundamental approximation algorithm design techniques based on linear programming: (a) LP-relaxation and rounding, and (b) the primal-dual method. In this lecture note, we will discuss the former.

The idea of LP-relaxation and rounding is quite simple. We first formulate an optimization problem as an *integer program* (IP), which is like a linear program (LP) with integer variables. Then, we solve the LP for an optimal solution, say $\mathbf{x}^*$. From $\mathbf{x}^*$, we construct a feasible solution $\mathbf{x}^A$ to the IP. This construction step is often called *rounding*. Rounding can be done deterministically or randomly with some probability distribution. In the latter approach is taken, we obtain the *randomized rounding* method.

Let cost$(\mathbf{x}^A)$ and cost$(\mathbf{x}^*)$ denote the objective values of $\mathbf{x}^A$ and $\mathbf{x}^*$, respectively. Let $\text{OPT}(IP)$ and $\text{OPT}(LP)$ denote the optimal values of the the IP and the LP, respectively. (Note that $\text{OPT}(LP) = \text{cost}(\mathbf{x}^*)$.) Suppose we are working on a minimization problem, then the performance ratio of this algorithm can be obtained by observing that

$$\text{cost}(\mathbf{x}^A) \leq \frac{\text{cost}(\mathbf{x}^A)}{\text{OPT}(LP)} \times \text{OPT}(LP) \leq \frac{\text{cost}(\mathbf{x}^A)}{\text{cost}(\mathbf{x}^*)} \times \text{OPT}(IP).$$

Consequently, any upper bound of the ratio $\frac{\text{cost}(\mathbf{x}^A)}{\text{cost}(\mathbf{x}^*)}$ is an approximation ratio for this algorithm. The ratio is also referred to as the *integrality gap*.

The previous paragraph describes how typically things go. However, it is possible in principle to prove approximation ratios better than the integrality grap by delving deeper into the structure of the problem at hand.

1 Linear programs and linear integer programs

A *linear integer program* is similar to a linear program with an additional requirement that variables are integers. The INTEGER PROGRAMMING problem is the problem of determining if a given integer program has a feasible solution. This problem is known to be $\mathbf{NP}$-hard. Hence, we cannot hope to solve general integer programs efficiently. However, integer programs can often be used to formulate a lot of discrete optimization problems.

One of the most well-known NP-hard problems is the following: VERTEX COVER

> Given a graph $G = (V, E)$, $|V| = n$, $|E| = m$, find a minimum vertex cover, namely a subset $C \subseteq V$ with size as small as possible such that for each edge $ij \in E$, either $i \in C$ or $j \in C$.

Let us see how we can formulate VERTEX COVER as an integer program. Suppose we are given a graph $G = (V, E)$ with n vertices and n edges. For each $i \in V = \{1, \dots, n\}$, let $x_i \in \{0, 1\}$ be a variable which is 1 if i belongs to the vertex cover, and 0 otherwise; then, the problem is equivalent to solving the following (linear) *integer program*:

$$\begin{array}{rll} \min & x_1 + x_2 + \cdots + x_n & \\ \text{subject to} & x_i + x_j \geq 1, & \forall ij \in E, \\ & x_i \in \{0, 1\}, & \forall i \in V. \end{array} \tag{1}$$

The objective function basically counts the number of vertices in the vertex cover. Each inequality $x_i + x_j \geq 1, ij \in E$ requires each edge to have at least one of its end points in the vertex cover. Actually, the formulation above is somewhat too strict. Suppose we relax it a little bit:

$$\begin{array}{ll} \min & x_1 + x_2 + \cdots + x_n \\ \text{subject to} & x_i + x_j \geq 1, \quad \forall ij \in E, \\ & x_i \geq 0, x_i \in \mathbb{Z} \quad \forall i \in V. \end{array}$$

Then, this would still be equivalent to solving the vertex cover problem, since in an optimal solution to the integer program above, none of the x_i can be more than 1 (why?).

The next problem is a generalized version of the VERTEX COVER problem.

WEIGHTED VERTEX COVER

Given a graph $G = (V, E)$, $|V| = n$, $|E| = m$, a weight function $w : V \to \mathbb{R}$. Find a vertex cover $C \subseteq V$ for which $\sum_{i \in C} w(i)$ is minimized.

Note that when $w \equiv 1$, the weighted version is the same as the non-weighted version. An equivalent linear integer program can be written as

$$\begin{array}{ll} \min & w_1x_1 + w_2x_2 + \cdots + w_nx_n \\ \text{subject to} & x_i + x_j \geq 1, \quad \forall ij \in E, \\ & x_i \in \{0, 1\}, \quad \forall i \in V. \end{array}$$

Note that if the weights were all non-negative, then we only have to require the variables to be non-negative integers, just like in the case of normal vertex cover. An integer program (IP) as above is also referred to as a 01-integer program.

The next two problems are more general versions of the VERTEX COVER and the WEIGHTED VERTEX COVER problems. Recall that we use $[n]$ to denote the set $\{1, \ldots, n\}$, for all positive integer n, and $[0]$ naturally denotes $\emptyset$.

SET COVER

Given a collection $\mathcal{S} = \{S_1, \ldots, S_n\}$ of subsets of $[m] = \{1, \ldots, m\}$. Find a sub-collection $\mathcal{C} = \{S_i \mid i \in J\}$ with as few members as possible (i.e. $|J|$ as small as possible) such that $\bigcup_{i \in J} S_i = [m]$.

Similar to VERTEX COVER, we use a 01-variable x_j to indicate the inclusion of S_j in the cover. For each $i \in \{1, \ldots, m\}$, we need at least one of the S_j containing i to be picked. The integer program is then

$$\begin{array}{ll} \min & x_1 + \cdots + x_n \\ \text{subject to} & \displaystyle\sum_{j : S_j \ni i} x_j \geq 1, \quad \forall i \in [m], \\ & x_j \in \{0, 1\}, \quad \forall j \in [n]. \end{array}$$

WEIGHTED SET COVER

Given a collection $\mathcal{S} = \{S_1, \ldots, S_n\}$ of subsets of $[m] = \{1, \ldots, m\}$, and a weight function $w : \mathcal{S} \to \mathbb{R}$. Find a cover $\mathcal{C} = \{S_j \mid j \in J\}$ with minimum total weight.

Trivially, we have

$$\begin{array}{ll} \min & w_1x_1 + \cdots + w_nx_n \\ \text{subject to} & \displaystyle\sum_{j : S_j \ni i} x_j \geq 1, \quad \forall i \in [m], \\ & x_j \in \{0, 1\}, \quad \forall j \in [n]. \end{array}$$

TRAVELING SALESMAN (TSP)

A salesman must visit n cities each exactly once and return to the originating city. Given the time to go from city i to city j is t_{ij}, find a tour which takes the shortest time.

This problem is slightly more difficult to formulate then the ones we have seen. Let x_{ij} be the 01 variable indicating if the salesman does go from city i to city j. Obviously, we want the salesman to go into i exactly once and to go out of j exactly once for each city i, j. This condition alone, however, does not ensure connectivity as we might form a few disjoint cycles. We also need constraints to ensure the connectivity of our tour. This could be done by checking for each non-empty set $S \subset [n]$ if there was at least one edge leaving S. In summary, we have the following equivalent linear integer program:

$$\begin{array}{lll}
\min & \sum_{i \neq j} t_{ij} x_{ij} & \\
\text{subject to} & \sum_{j:j\neq i} x_{ij} = 1, & \forall i \in [n], \\
& \sum_{i:i\neq j} x_{ij} = 1, & \forall j \in [n], \\
& \sum_{i \in S, j \notin S} x_{ij} \geq 1, & \forall S \subset [n], S \neq \emptyset \\
& \quad x_{ij} \in \{0,1\}, & \forall i,j \in [n], i \neq j.
\end{array}$$

Exercise 1 (ASSIGNMENT PROBLEM)**.** Formulate the following problem as an IP problem:

> There are n processors and n tasks. Each processor might be more suitable to perform a particular kind of task. Hence, there is a cost w_{ij} associated if processor i was to do task j. Find a one-to-one assignment of processors to tasks which minimizes the total cost.

Exercise 2 (KNAPSACK)**.** Formulate the following problem as an IP problem:

> Given n items with values $v_1, \dots, v_n$, and weights $w_1, \dots, w_n$, correspondingly. Find a subset S of items with total weight at most a given W, such that the total value of S is as large as possible.

Exercise 3 (INDEPENDENT SET)**.** Formulate the following problem as an IP problem:

> Given a graph $G = (V, E)$, a weight function $w : V \to \mathbb{R}^+$, find an independent set of maximum total weight; namely, a subset $P \subseteq V$ of vertices such that no pair of vertices in P are adjacent and the sum $\sum_{i \in P} w(i)$ is maximized.

Exercise 4. Given an $m \times n$ matrix A whose entries are either 0 or 1, and $w \in \mathbb{Z}^n, c \geq \vec{0}$. As usual, we use $\vec{0}$ to denote the all-0 vector, whose dimension is implicit in the (in)equality involved, and $\vec{1}$ to denote the all-1 vector. The (weighted) SET COVER problem has the form

$$\min \left\{ w^T x \mid Ax \geq \vec{1}, x \in \{0,1\}^n \right\}, \tag{2}$$

while the INDEPENDENT SET problem in the previous exercise has the form

$$\max \left\{ w^T x \mid Ax \leq \vec{1}, x \in \{0,1\}^n \right\}, \tag{3}$$

(That is, if you do it correctly.)

In this problem, we shall see that the converse is also true:

(i) Given an integer program of the form (2), where A is **any** 01-matrix, formulate a (weighted) SET COVER instance which is equivalent to the program.

(ii) Given an integer program of the form (3), where A is **any** 01-matrix, formulate an INDEPENDENT SET instance which is equivalent to the program.

In both questions, show the "equivalence." For example, in (i) you must show that the minimum weighted set cover of the constructed set family corresponds to an optimal solution of the integer program and vice versa.

Exercise 5 (BIN PACKING)**.** Formulate the following problem as an IP problem:

> Given a set of n items $\{1, \dots, n\}$, and their "size" $s(i) \in (0, 1]$. Find a way to partition the set of items in to a minimum number m of "bins" $B_1, \dots, B_m$, such that
>
> $$\sum_{i \in B_j} s(i) \leq 1, \quad \forall j \in [m].$$

2 Relaxation and rounding

In general, *relaxation* refers to the action of relaxing the integer requirement of a linear IP to turn it into an LP. For example, the LP corresponding to the IP (1) of VERTEX COVER is

$$\begin{array}{rl} \min & x_1 + x_2 + \dots + x_n \\ \text{subject to} & x_i + x_j \geq 1, \quad \forall ij \in E, \\ & 0 \leq x_i \leq 1, \quad \forall i \in V. \end{array} \tag{4}$$

Obviously if the LP version is infeasible, then the IP version is also infeasible. This is the first good reason to do relaxation. Now, suppose x^* is an optimal solution to the LP problem. We know that x^* can be found in polynomial time. We shall construct a feasible solution x^A to the IP problem as follows. Let

$$x_i^A = \begin{cases} 1 & \text{if } x_i^* \geq 1/2 \\ 0 & \text{if } x_i^* < 1/2. \end{cases}$$

You should check that x^A is definitely feasible for the IP. This technique of constructing a feasible solution for the IP from the LP is called *rounding*. We have just seen the second advantage of doing relaxation. The third is that an optimal value for the LP provides a lower bound for the optimal value of the IP (why?). Using this fact, one can derive the approximation ratio of the feasible solution x^A. Let $\text{OPT}(IP)$ be the optimal value for the IP instance, $\text{OPT}(LP)$ be the optimal value for the LP, and $Cost(x^A)$ be the cost of the feasible solution x^A for the IP, then we have

$$\begin{array}{rcl} \text{OPT}(IP) & \geq & \text{OPT}(LP) \\ & = & x_1^* + \dots + x_n^* \\ & \geq & \frac{1}{2}x_1^A + \dots + \frac{1}{2}x_n^A \\ & = & \frac{1}{2}Cost(x^A). \end{array}$$

In other words, the cost of the approximation x^A is at most twice the optimal. We thus have a 2-approximation algorithm to solve the VERTEX COVER problem. Since it is impossible, unless P $=$ NP, to have an exact polynomial time algorithm to solve VERTEX COVER, an algorithm giving a feasible solution within twice the optimal is quite satisfactory. The exact same technique works for the WEIGHTED VERTEX COVER problem, when the weights are non-negative. Thus, we also have a 2-approximation algorithm for the WEIGHTED VERTEX COVER problem. (Note, again, that when we say "approximation algorithm," it automatically means a polynomial-time approximation algorithm. There would be no point approximating a solution if it takes exponentially long.)

Theorem 2.1. *There is an approximation algorithm to solve the* WEIGHTED VERTEX COVER *problem with approximation ratio* 2*.*

Obviously, one would like to reduce the ratio 2 to be as close to 1 as possible. However, no approximation algorithm with a constant ratio less than 2 is known to date. It is possible that 2 is the best we can hope for. It has been shown that approximating WEIGHTED VERTEX COVER to within $10\sqrt{5} - 21$ [4].

To this end, let us attempt to use the relaxation and rounding idea to find approximation algorithms for the WEIGHTED SET COVER problem. In fact, we shall deal with the following much more general problem called the GENERAL COVER problem:

$$\begin{array}{lcccccccc} \min & c_1x_1 & + & \dots & + & c_nx_n & & & \\ \text{subject to} & a_{i1}x_1 & + & \dots & + & a_{in}x_n & \geq & b_i, & i \in [m]. \\ & & & & & & & x_j \in \{0,1\}, & \forall j \in [n], \end{array} \tag{5}$$

where a_{ij}, b_i, c_j are all non-negative integers. Since we can remove an inequality if $b_i = 0$, we can assume that $b_i > 0, \forall i \in [n]$. Moreover, if $c_j = 0$ then we can set $x_j = 1$ and remove the column corresponding to j without effecting the objective function as well as feasible solutions. Thus, we can also assume that $c_j > 0, \forall j \in [n]$. Lastly, we assume that a feasible solution exists, namely $\sum_j a_{ij} \geq b_i$, for all $i \in [m]$.

The relaxed LP version for (5) is

$$\begin{array}{lcccccccc} \min & c_1x_1 & + & \dots & + & c_nx_n & & & \\ \text{subject to} & a_{i1}x_1 & + & \dots & + & a_{in}x_n & \geq & b_i, & i \in [m]. \\ & & & & & & & 0 \leq x_j \leq 1, & \forall j \in [n], \end{array} \tag{6}$$

Let $\mathbf{x}^*$ be an optimal solution to the LP version. How would we round $\mathbf{x}^*$ to get $\mathbf{x}^A$ as in the VERTEX COVER case? Firstly, the rounding must ensure that $\mathbf{x}^A$ is feasible, namely they must satisfy each of the inequalities $a_{i1}x_1 + \cdots + a_{in}x_n \geq b_i$. Secondly, we do not want to "over-round," such as assigning everything to 1, which would give a feasible solution but it does not give a very good approximation. Consider an inequality such as

$$3x_1^* + 4x_2^* + x_3^* + 2x_4^* \geq 4, \tag{7}$$

which x^* satisfies. If we were to round some of the x_i^* up to 1, and the rest down to 0, we must pick the ones whose coefficients sum up to 4 or more; for instance, we could round x_2^* up to 1 and the rest to 0, or x_1^* and x_3^* to 1 and the rest to 0. The difficulty with this idea is that there might be an exponential number of ways to do this, and we also have to do this consistently throughout all inequalities $a_{i1}x_1 + \cdots + a_{in}x_n \geq b_i$. We cannot round x_1^* to 0 in one inequality, and to 1 in another inequality. Fortunately, some information about which x_j^* to round is contained in the actual values of the x_j^*. Consider inequality (7) again. The sum of all coefficients of the x_j^* is 10. If all x_j^* were at most $1/10$, then the left hand side is at most 1. Hence, there must be some x_j^* which are at least $1/10$. If $x_2^* \geq 1/10$, and we round it up to 1, then we'd be fine. However, if x_3^* and x_4^* are the only ones which are $\geq 1/10$, then rounding them up to 1 is not sufficient. Fortunately, that cannot happen, because if $x_1^*, x_2^* < 1/10$ and $x_3^*, x_4^* \geq 1/10$, then

$$3x_1^* + 4x_2^* + x_3^* + 2x_4^* < \frac{3}{10} + \frac{4}{10} + 1 + 2 < 4.$$

Thus, *the sum of the coefficients of the x_j^* which are at least $1/(a_{i1} + \cdots + a_{in})$ has to be at least b_i.* This is an informal proof. We will give a rigorous proff later.

The analysis above leads to the following rounding strategy. Let

$$f = \max_{i=1..m} \left(\sum_{j=1}^{n} a_{ij} \right),$$

and set

$$x_j^A = \begin{cases} 1 & \text{if } x_j^* \geq \frac{1}{f} \\ 0 & \text{if } x_j^* < \frac{1}{f}, \end{cases}$$

then we have an approximation ratio of f. You should map this rounding back to the rounding of the VERTEX COVER problem to see the analogy.

Theorem 2.2. *The rounding strategy above gives an approximation algorithm for the* GENERAL COVER *problem with approximation ratio at most f.*

Proof. We first show that x^A is indeed feasible for IP-GC (the integer program for the GENERAL COVER problem). Suppose x^A is not feasible, then there is some row i for which

$$a_{i1}x_1^A + \cdots + a_{in}x_n^A < b_i.$$

This is the same as

$$\sum_{j:x_j^* \geq 1/f} a_{ij} \leq b_i - 1.$$

If $\sum_{j:x_j^* < 1/f} a_{ij} > 0$, then

$$\sum_{j=1}^n a_{ij}x_j^* = \sum_{j:x_j^* \geq 1/f} a_{ij}x_j^* + \sum_{j:x_j^* < 1/f} a_{ij}x_j^* < \sum_{j:x_j^* \geq 1/f} a_{ij} + 1 \leq b_i,$$

which is a contradiction. On the other hand, when $\sum_{j:x_j^* < 1/f} a_{ij} = 0$, we get

$$\sum_{j=1}^n a_{ij}x_j^* = \sum_{j:x_j^* \geq 1/f} a_{ij}x_j^* + \sum_{j:x_j^* < 1/f} a_{ij}x_j^* = \sum_{j:x_j^* \geq 1/f} a_{ij} \leq b_i - 1,$$

which again is a contradiction to the feasibility of x^*.

For the performance ratio, notice that

$$\text{cost}(x^A) = \sum_{j=1}^n c_j x_j^A \leq \sum_{j=1}^n c_j(f x_j^*) = f \text{ OPT(LP-GC)} \leq f \text{ OPT(IP-GC)}.$$

□

Exercise 6. Describe the ratio f for the WEIGHTED SET COVER problem in terms of m, n and the set collection $\mathcal{S}$.

Exercise 7. Suppose we apply the rounding strategy above to the WEIGHTED SET COVER problem. However, instead of rounding the $x_j^* \geq 1/f$ up to 1, we round all positive x_j^* up to 1. Show that we still have an f-approximation algorithm. (**Hint**: consider the primal complementary slackness condition.)

3 Randomized Rounding

Randomized rounding refers to the strategy of rounding fractional variables randomly according to some probability distribution. We want the result to be an expectedly good solution (with some performance ratio) with high probability (say $\geq 1/2$). We can run the algorithm independently many times to increase this probability to be as large as we want.

3.1 Weighted Set Cover (WSC)

In the WSC problem, we are given a collection $\mathcal{S} = \{S_1, \dots, S_n\}$ of subsets of $[m] = \{1, \dots, m\}$, where S_j is of weight $w_j \in \mathbb{Z}^+$. The objective is to find a sub-collection $\mathcal{C} = \{S_i \mid i \in J\}$ with least total weight such that $\bigcup_{i \in J} S_i = [m]$. The corresponding integer program is

$$\begin{array}{rl} \min & w_1 x_1 + \dots + w_n x_n \\ \text{subject to} & \displaystyle\sum_{j : S_j \ni i} x_j \geq 1, \quad \forall i \in [m], \\ & x_j \in \{0, 1\}, \quad \forall j \in [n]. \end{array}$$

And, relaxation gives the following linear program:

$$\begin{array}{rl} \min & w_1 x_1 + \dots + w_n x_n \\ \text{subject to} & \displaystyle\sum_{j : S_j \ni i} x_j \geq 1, \quad \forall i \in [m], \\ & 0 \leq x_j \leq 1 \quad \forall j \in [n]. \end{array}$$

Suppose we have an optimal solution $\mathbf{x}^*$ of the LP. To obtain $\mathbf{x}^A$, a sensible rounding strategy is to round x_j^* to 1 with probability x_j^*, namely

$$\text{Prob}[x_j^A = 1] = x_j^*.$$

It follows that

$$\text{E}[\text{cost}(\mathbf{x}^A)] = \sum_{j=1}^{n} w_j x_j^* = \text{OPT}(LP).$$

What we really want is to find the probability that $\mathbf{x}^A$ is feasible and cost$(\mathbf{x}^A) \leq \rho \cdot \text{OPT}$. If this probability at least some positive constant, then ρ is an approximation ratio of this algorithm. (If the probability is small, we can run the algorithm independently a few times.) We can estimate the desired probability as follows.

$$\begin{aligned} \text{Prob}[\ \mathbf{x}^A \text{ is feasible and cost}(\mathbf{x}^A) \leq \rho \cdot \text{OPT}] &= 1 - \text{Prob}[\ \mathbf{x}^A \text{ is not feasible and cost}(\mathbf{x}^A) > \rho \cdot \text{OPT}] \\ &\geq 1 - \text{Prob}[\mathbf{x}^A \text{ is not feasible}] - \text{Prob}[\text{cost}(\mathbf{x}^A) > \rho \cdot \text{OPT}]. \end{aligned}$$

Let us first estimate the probability that $\mathbf{x}^A$ is not feasible. Consider any element $i \in [m]$, and suppose the inequality constraint corresponding to i is

$$x_{j_1} + \dots + x_{j_k} \geq 1.$$

We will refer to this as the ith constraint. Then, the probability that this constraint is not satisfied by $\mathbf{x}^A$ is

$$(1 - x_{j_1}^*) \dots (1 - x_{j_k}^*) \leq \left(\frac{k - (x_{j_1}^* + \dots + x_{j_k}^*)}{k} \right)^k \leq \left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e}.$$

Thus, Prob[$\mathbf{x}^A$ is not feasible] $\leq m/e$. This is a very bad bound since m is large. We can get a better bound by setting x_j^A to be 0 with lower probability. Let t be a number to be determined, and set $x_j^A = 0$ probability $(1 - x_j^*)^t$. (This is equivalent to running the previous strategy independently t rounds, and set $x_j^A = 0$ only when $x_j^A = 0$ in all rounds.) In this case,

$$\text{Prob}[\mathbf{x}^A \text{ does not satisfy constraint } i] \leq (1/e)^t.$$

Thus, the probability that $\mathbf{x}^A$ is not a feasible solution is at most $m(1/e)^t$. When t is large, $m(1/e)^t$ gets as small as we want.

Secondly, we estimate the probability that cost($\mathbf{x}^A$) $> \rho \cdot \text{OPT}$. In one round, we have shown that E[cost($\mathbf{x}^A$)] $= \text{OPT}(IP) \leq \text{OPT}$. Hence, with t rounds we have E[cost($\mathbf{x}^A$)] $\leq t \cdot \text{OPT}$. Markov inequality gives

$$\text{Prob}[\text{cost}(\mathbf{x}^A) > \rho \cdot \text{OPT}] < \frac{\text{E}[\text{cost}(\mathbf{x}^A)]}{\rho \cdot \text{OPT}} \leq \frac{t \cdot \text{OPT}}{\rho \cdot \text{OPT}} = \frac{t}{\rho}.$$

Remark 3.1. Let X be a random variable in $\mathbb{R}^+$, and a be a positive number, Markov inequality says that Prob$[X \geq a] \leq \frac{\text{E}[X]}{a}$.

Consequently,

$$\text{Prob}[\mathbf{x}^A \text{ is feasible and } \text{cost}(\mathbf{x}^A) \leq \rho \cdot \text{OPT}(IP)] \geq 1 - m(1/e)^t - \frac{t}{\rho}.$$

We can pick $t = \theta(\lg m)$ and $\rho = 4t$ so that $1 - m(1/e)^t - \frac{t}{\rho} \geq \frac{1}{2}$. In other words, this algorithm gives a solution with approximation ratio $\Theta(\lg m)$ with probability at least $1/2$. We can then run the algorithm a few times until the solution is feasible. The expected number of runs is 2, and the expected approximation ratio is $\Theta(\lg m)$.

Exercise 8. Suppose we run the above randomized rounding algorithm with only one round (instead of t rounds). Prove that, with positive probability the resulting $\mathbf{x}^A$ satisfies at least half of the constraints at cost at most $O(\text{OPT}(IP))$.

Exercise 9. Give a randomized rounding algorithm for the GENERAL COVER problem with approximation ratio $O(\lg m)$.

Exercise 10. Given a graph G and and proper k-coloring of G (i.e. a coloring using k colors such that two adjacent vertices have different colors). We want to solve the WEIGHTED VERTEX COVER on G. Devise a polynomial time $(2 - 2/k)$-approximation algorithm to solve this problem.

3.2 A simple randomized algorithm for MAX-3SAT

A *conjunctive normal form* (CNF) formula is a boolean formula on n variables $\mathcal{X} = \{x_1, \dots, x_n\}$ consisting of m *clauses* $C_1, \dots, C_m$. Each clause is a subset of *literals*, which are variables and negations of variables. A clause can be viewed as the sum (or the OR) of the literals. A clause is satisfied by a truth assignment $a : \mathcal{X} \to \{\text{TRUE}, \text{FALSE}\}$ if one of the literals in the clause is TRUE.

Consider integers $k \geq 2$. A *k-CNF formula* is a CNF formula in which each clause is of size at most k. An *Ek-CNF formula* is a CNF formula in which each clause is of size exactly k.

Given a CNF formula φ, the MAX-SAT problem is to find a truth assignment satisfying the maximum number of clauses in φ. If φ is of the form X-CNF, for X $\in \{k, \text{E}k\}$, then we get the corresponding MAX-XSAT problems.

Exercise 11. Show that the problem of deciding if a 2-CNF formula is satisfiable is in P, but MAX-2SAT is NP-Hard (i.e. its decision version is **NP**-complete).

Exercise 12. State the decision version of MAX-E3SAT and show that it is **NP**-complete.

Theorem 3.2. *There exists an $8/7$-approximation algorithm for* MAX-E3SAT.

Proof. Let φ be an E3-CNF formula with m clauses $C_1, \dots, C_m$. Let S_φ be the random variable counting the number of satisfied clauses of φ by randomly setting x_i independently to be TRUE with probability $1/2$. Since the probability that a clause C_j is satisfied is $7/8$, by linearity of expectation $\text{E}[S_\varphi] = 7m/8$. This number clearly is within a factor $7/8$ of the optimal value. Hence, this simple randomized algorithm achieves (expected) approximation ratio $8/7$. We can derandomize this algorithm by a method known as *conditional expectation*. The basic idea is as follows.

Consider a fixed $k \in [n]$. Let $a_1, \dots, a_k \in \{\text{TRUE}, \text{FALSE}\}$ be k boolean values. Let φ' be a formula obtained by setting $x_i = a_i,\ i \leq j$, and discarding all c clauses that are already satisfied. Then, it is easy to see that

$$\mathrm{E}[S_\varphi \mid x_i = a_i, 1 \leq i \leq k] = \mathrm{E}[S_{\varphi'}] + c.$$

Hence, given $a_1, \dots, a_k$ we can easily compute $\mathrm{E}[S_\varphi \mid x_i = a_i, 1 \leq i \leq k]$ in polynomial time.

Now, for $k \geq 1$, notice that

$$\begin{aligned} & \mathrm{E}[S_\varphi \mid x_i = a_i, 1 \leq i \leq k-1] \\ = \ & \frac{1}{2}\mathrm{E}[S_\varphi \mid x_i = a_i, 1 \leq i \leq k-1,\ x_k = \text{TRUE}] + \frac{1}{2}\mathrm{E}[S_\varphi \mid x_i = a_i, 1 \leq i \leq k-1,\ x_k = \text{FALSE}] \end{aligned}$$

The larger of the two expectations on the right hand side is at least $\mathrm{E}[S_\varphi \mid x_i = a_i, 1 \leq i \leq k-1]$. Hence, we can set x_i to be TRUE or FALSE one by one, following the path that leads to the larger expectation, to eventually get a truth assignment which satisfies as many clauses as $\mathrm{E}[S_\varphi] = 7m/8$. □

3.3 Randomized rounding and MAX-SAT

3.3.1 The straightforward randomized algorithm

Consider the WEIGHTED MAX-SAT problem in which a formula ϕ consists of m clauses $C_1, \dots, C_m$ weighted $w_1, \dots, w_m \in \mathbb{Z}^+$. Let $x_1, \dots, x_n$ be the variables and l_j denote the length of clause C_j. Suppose we follow the previous section and set each variable to be TRUE with probability $1/2$, and derandomized this algorithm, then what is the approximation ratio?

Let I_j denote the random variable indicating the event $\{C_j$ is satisfied$\}$, i.e.

$$I_j := \begin{cases} 1 & \text{if } C_j \text{ is satisfied} \\ 0 & \text{otherwise.} \end{cases}$$

Let S_ϕ be the cost (weight) of a random assignment and $\text{OPT}(\phi)$ be the cost of an optimal assignment, then $S_\phi = w_j I_j$. We have

$$\mathrm{E}[S_\phi] = \sum_{j=1}^m w_j \,\text{Prob}[I_j = 1] = \sum_{j=1}^m w_j (1 - (1/2)^{l_j}) \geq \frac{1}{2}\sum_{j=1}^m w_j \geq \frac{1}{2}\text{OPT}(\phi).$$

In other words, with derandomization using the method of conditional expectation, we can get a (deterministic) approximation algorithm for MAX-SAT with approximation ratio 2.

Exercise 13. Consider the following algorithm for MAX-SAT: let τ be any truth assignment and τ' be its complement, i.e. $\tau'(x_i)$ is the negation of $\tau(x_i)$. Compute the cost of both τ and τ', then output the better assignment. Show that this is a 2-approximation algorithm.

Exercise 14. Let $\mathbb{F}_2 = \{0, 1\}$. Arithmetics over $\mathbb{F}_2$ is done modulo 2. Consider a system of m linear equations on n variables over $\mathbb{F}_2$. The LINEAR EQUATIONS OVER $\mathbb{F}_2$ problem is the problem of finding an assignment to variables that satisfies as many equations as possible. Give a randomized algorithm for this problem with approximation ratio 2, then derandomize it using the method of conditional expectation.

3.3.2 A randomized algorithm with a biased coin

The approximation ratio 2 as done above is not nearly as good as $8/7$ we had for MAX-3SAT. Perhaps this is due to the fact that MAX-SAT is not as symmetrical as MAX-3SAT. Thus, our "rounding probability" should not be $1/2$. This observation suggest us to set each variable to TRUE with some probability q to be determined. Due to symmetry (of a variable and its negation), we only need to consider $q \geq 1/2$ (thus $q \geq 1-q$).

Let n_j and p_j be the number of negated variables and non-negated variables in clause C_j, then

$$\mathrm{E}[S_\phi] = \sum_{j=1}^{m} w_j (1 - q^{n_j}(1-q)^{p_j}).$$

To get a good approximation ratio, we want all the $q^{n_j}(1-q)^{p_j}$ to be as small as possible. This product is large for small clauses, especially the clauses with only one single literal. Let us consider them first.

- If singleton clauses contain no negations of variables, then it is easy to see that $q^{n_j}(1-q)^{p_j} \leq \max\{1-q, q^2\}$, for all j. To minimize the $\max$, we pick q such that $1-q = q^2$, i.e. $q \approx 0.618$. In this case, we have
$$\mathrm{E}[S_\phi] \geq \frac{1}{q}\text{OPT}(\phi).$$
(Note that this is slightly better than the ratio 2.)

- If there is no i such that both x_i and $\bar{x}_i$ are clauses, then by swapping labels of some x_i and $\bar{x}_i$, we can obtain the same bound.

- The situation comes down to the case when there are x_i such that both x_i and $\bar{x}_i$ are clauses. Firstly, note that two clauses of the form $\{x_i\}$ (or of the form $\{\bar{x}_i\}$) can be combined into one (whose weight is the total weight). Consequently, we can assume that x_i (and $\bar{x}_i$) does not appear in two singleton clauses. Secondly, if x_i and $\bar{x}_i$ are both clauses, we can assume that the weight of the x_i-clause is at least the weight of the $\bar{x}_i$-clause, otherwise we swap x_i and $\bar{x}_i$. Thirdly, assume the rest of the singleton clauses contain only non-negated variables. Define
$$N = \{j \mid C_j = \{\bar{x}_i\}, \text{ for some } i\}.$$
Then,
$$\text{OPT}(\phi) \leq \sum_{j=1}^{m} w_j - \sum_{j \in N} w_j.$$
And,
$$\mathrm{E}[S_\phi] = \sum_{j \notin N} w_j(1 - q^{n_j}(1-q)^{p_j}) + \sum_{j \in N} w_j(1-q) \geq q \sum_{j=1}^{m} w_j - q \sum_{j \in N} w_j \geq q \cdot \text{OPT}(\phi).$$

3.3.3 A randomized algorithm with different biased coins based on linear programming

The above randomized algorithms do not deal well with small-size clauses. In this section, we borrow the idea from the randomized-algorithm for SET COVER, using linear programming to determine the rounding probability of each variable.

An integer program for MAX-SAT can be obtained by considering the following 01-variables: (a) $y_i = 1$ iff $x_i = \text{TRUE}$; and (b) $z_j = 1$ iff C_j is satisfied. We then have the following integer program

$$\begin{array}{rll} \max & w_1z_1 + \cdots + w_mz_n & \\ \text{subject to} & \displaystyle\sum_{i:x_i\in C_j} y_i + \sum_{i:\bar{x}_i\in C_j}(1-y_i) \geq z_j, & \forall j \in [m], \\ & y_i, z_j \in \{0,1\}, & \forall i \in [n], j \in [m] \end{array}$$

and its relaxed version

$$\begin{array}{rll} \max & w_1z_1 + \cdots + w_nz_n & \\ \text{subject to} & \displaystyle\sum_{i:x_i\in C_j} y_i + \sum_{i:\bar{x}_i\in C_j}(1-y_i) \geq z_j, & \forall j \in [m], \\ & 0 \leq y_i \leq 1 & \forall i \in [n], \\ & 0 \leq z_j \leq 1 & \forall j \in [m]. \end{array}$$

Mimicking the rounding strategy for SET COVER, we obtain an optimal solution (y^*, z^*) for the linear program, and round $x_i = \text{TRUE}$ with probability y_i^*. Then,

$$\begin{aligned}
\mathrm{E}[S_\phi] &= \sum_{j=1}^m w_j \left(1 - \prod_{i:x_i\in C_j}(1-y_i^*)\prod_{i:\bar{x}_i\in C_j} y_i^*\right) \\
&\geq \sum_{j=1}^m w_j \left(1 - \left[\frac{\sum_{i:x_i\in C_j}(1-y_i^*) + \sum_{i:\bar{x}_i\in C_j} y_i^*}{l_j}\right]^{l_j}\right) \\
&= \sum_{j=1}^m w_j \left(1 - \left[\frac{l_j - \left(\sum_{i:x_i\in C_j} y_i^* + \sum_{i:\bar{x}_i\in C_j}(1-y_i^*)\right)}{l_j}\right]^{l_j}\right) \\
&\geq \sum_{j=1}^m w_j \left(1 - \left[1 - \frac{z_j^*}{l_j}\right]^{l_j}\right) \\
&\geq \sum_{j=1}^m w_j \left(1 - \left[1 - \frac{1}{l_j}\right]^{l_j}\right) z_j^* \\
&\geq \min_j \left(1 - \left[1 - \frac{1}{l_j}\right]^{l_j}\right) \sum_{j=1}^m w_j z_j^* \\
&\geq \left(1 - \frac{1}{e}\right) \text{OPT}(\phi).
\end{aligned}$$

(We have used the fact that the function $f(x) = (1-(1-x/l_j)^{l_j}$ is concave when $x \in [0,1]$, thus it lies above the segment through the end points.) We have just proved

Theorem 3.3. *The LP-based randomized rounding algorithm above has approximation ratio* $e/(e-1)$.

Note that $e/(e-1) \approx 1.58$, while $1/q \approx 1/0.618 \approx 1.62$. Thus, this new algorithm is slightly better than the one with a biased coin.

Exercise 15. Describe how to use the method of conditional expectation to derandomize the algorithm above.

Exercise 16. Let $g(y)$ be any function such that $1 - 4^{-y} \le g(y) \le 4^{y-1}, \forall y \in [0,1]$. Suppose we set each $x_i = \text{TRUE}$ with probability $g(y_i^*)$, where (y^*, z^*) is an optimal solution to the linear program. Show that this strategy gives a $4/3$-approximation algorithm for MAX-SAT.

3.3.4 The "best-of-two" algorithm

Note that the rounding algorithm in the previous section works fairly well if clauses are of small sizes. For instance, if $l_j \le 2$ for all j, then the approximation ratio would have been $1/(1-(1-1/2)^2) = 4/3$. On the other hand, the straightforward randomized algorithm works better when clauses are large. It just makes sense to now combine the two: run both algorithms and report the better assignment. Let S_ϕ^1 and S_ϕ^2 (which are random variables) denote the corresponding costs. Then, it is easy to see the following

$$\begin{aligned}
\mathrm{E}[\max\{S_\phi^1, S_\phi^2\}] &\ge \mathrm{E}[(S_\phi^1 + S_\phi^2)/2] \\
&\ge \sum_{j=1}^m w_j \left(\frac{1}{2}\left(1 - \frac{1}{2^{l_j}}\right) + \frac{1}{2}\left(1 - \left[1 - \frac{1}{l_j}\right]^{l_j}\right) z_j^* \right) \\
&\ge \frac{3}{4}\sum_{j=1}^m w_j z_j^* \\
&\ge \frac{3}{4}\text{OPT}(\phi).
\end{aligned}$$

Thus, the BEST-OF-TWO algorithm has performance ratio $4/3$.

Historical Notes

Texts on Linear Programming are numerous, of which I recommend [3] and [15]. For Integer Programming, [18] and [15] are suggested. Recent books on approximation algorithms include [2, 8, 13, 17]. For linear algebra, see [9, 16]. See [1, 14] for randomized algorithms, derandomization and the probabilistic methods.

The $8/7$-approximation algorithm for MAX-E3SAT follows the line of Yannakakis [19], who gave the first $4/3$-approximation for MAX-SAT. A 2-approximation for MAX-SAT was given in the seminal early work of Johnson [10]. Johnson's algorithm can also be interpreted as a derandomized algorithm, mostly the same as the one we presented. The LP-based randomized algorithm and the best-of-two algorithm for MAX-SAT are due to Goemans and Williamson [6]. The algorithm with a biased coin is due to Lieberherr and Specker [12].

Later, Karloff and Zwick [11] gave an $8/7$-approximation algorithm for MAX-3SAT based on semidefinite programming. This approximation ratio is optimal as shown by Håstad [7]. The conditional expectation method was implicit in Erdős and Selfridge [5].

References

[1] N. ALON AND J. H. SPENCER, *The probabilistic method*, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley-Interscience [John Wiley & Sons], New York, second ed., 2000. With an appendix on the life and work of Paul Erdős.

[2] G. AUSIELLO, P. CRESCENZI, G. GAMBOSI, V. KANN, A. MARCHETTI-SPACCAMELA, AND M. PROTASI, *Complexity and approximation*, Springer-Verlag, Berlin, 1999. Combinatorial optimization problems and their approximability properties, With 1 CD-ROM (Windows and UNIX).

[3] V. CHVÁTAL, *Linear programming*, A Series of Books in the Mathematical Sciences, W. H. Freeman and Company, New York, 1983.

[4] I. DINUR AND S. SAFRA, *The importance of being biased*, in STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, ACM Press, 2002, pp. 33–42.

[5] P. ERDŐS AND J. L. SELFRIDGE, *On a combinatorial game*, J. Combinatorial Theory Ser. A, 14 (1973), pp. 298–301.

[6] M. X. GOEMANS AND D. P. WILLIAMSON, *New $\frac{3}{4}$-approximation algorithms for the maximum satisfiability problem*, SIAM J. Discrete Math., 7 (1994), pp. 656–666.

[7] J. HÅSTAD, *Some optimal inapproximability results*, in STOC '97 (El Paso, TX), ACM, New York, 1999, pp. 1–10 (electronic).

[8] D. S. HOCHBAUM, ed., *Approximation Algorithms for NP Hard Problems*, PWS Publishing Company, Boston, MA, 1997.

[9] R. A. HORN AND C. R. JOHNSON, *Matrix analysis*, Cambridge University Press, Cambridge, 1985.

[10] D. S. JOHNSON, *Approximation algorithms for combinatorial problems*, J. Comput. System Sci., 9 (1974), pp. 256–278. Fifth Annual ACM Symposium on the Theory of Computing (Austin, Tex., 1973).

[11] H. KARLOFF AND U. ZWICK, *A 7/8-approximation algorithm for MAX 3SAT?*, in Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, Miami Beach, FL, USA, IEEE Press, 1997.

[12] K. J. LIEBERHERR AND E. SPECKER, *Complexity of partial satisfaction*, J. Assoc. Comput. Mach., 28 (1981), pp. 411–421.

[13] E. W. MAYR AND H. J. PRÖMEL, eds., *Lectures on proof verification and approximation algorithms*, vol. 1367 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1998. Papers from the Workshop on Proof Verification and Approximation Algorithms held at Schloß Dagstuhl, April 21–25, 1997.

[14] R. MOTWANI AND P. RAGHAVAN, *Randomized algorithms*, Cambridge University Press, Cambridge, 1995.

[15] A. SCHRIJVER, *Theory of linear and integer programming*, Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons Ltd., Chichester, 1986. A Wiley-Interscience Publication.

[16] G. STRANG, *Linear algebra and its applications*, Academic Press [Harcourt Brace Jovanovich Publishers], New York, second ed., 1980.

[17] V. V. VAZIRANI, *Approximation algorithms*, Springer-Verlag, Berlin, 2001.

[18] L. A. WOLSEY, *Integer programming*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons Inc., New York, 1998. A Wiley-Interscience Publication.

[19] M. YANNAKAKIS, *On the approximation of maximum satisfiability*, J. Algorithms, 17 (1994), pp. 475–502. Third Annual ACM-SIAM Symposium on Discrete Algorithms (Orlando, FL, 1992).