Optimization and Approximation	ENS Lyon
Lecture 8	
Lecturer: Alantha Newman	*November 15, 2016*

1 Using Linear Programs for Approximation Algorithms

Given a discrete optimization problem (e.g. minimum vertex cover, minimum set cover), an approximation algorithm returns an integer solution whose value is at most α times that of an optimal solution, where $\alpha \geq 1$. (For a maximization problem (e.g. maximum clique, maximum independent set), an α-approximation algorithm returns a solution whose value is at least α times that of an optimal solution, where $\alpha \leq 1$.) A general framework to design an approximation algorithm for a particular problem using linear programming can be described as follows:

a. Formulate the problem as an integer linear program.

b. Relax the integrality constraints and solve the resulting linear program in polynomial time.

c. Round the fractional solution to obtain a feasible integral solution.

d. Analyze the quality of this integral solution compared to the optimal fractional solution.

Let us consider the minimum vertex cover problem. Given a graph $G = (V, E)$, the goal is to find a minimum cardinality subset $S \subset V$ such that for all $ij \in E$, at least one vertex from the pair (i, j) belongs to the set S. We apply the above framework to this problem.

a. Formulate the following integer program for the vertex cover problem.

$$\begin{aligned} \min & \sum_{i \in V} x_i \\ \text{subject to:} \quad & x_i + x_j \geq 1, \quad \text{for all } ij \in E, \\ & x_i \in \{0, 1\}. \end{aligned} \tag{IP$_{VC}$}$$

b. Relax the integrality constraints to obtain a linear program.

$$\begin{aligned} \min & \sum_{i \in V} x_i \\ \text{subject to:} \quad & x_i + x_j \geq 1, \quad \text{for all } ij \in E, \\ & 0 \leq x_i \leq 1. \end{aligned} \tag{P$_{VC}$}$$

c. Let $\mathbf{x}^*$ denote an optimal solution for (P$_{VC}$). Let $S = \{i \in V \mid x_i^* \geq \frac{1}{2}\}$. Note that S is feasible since for each edge $ij \in E$, the constraints from (P$_{VC}$) imply that either x_i^* or x_j^* (or both) have value at least $\frac{1}{2}$.

d. How large can $|S|$ be? Each vertex $i \in V$ for which $x_i^* < \frac{1}{2}$ contributes zero to $|S|$, and each vertex for which $x_i^* \geq \frac{1}{2}$ contributes one to $|S|$. So $|S| \leq 2 \cdot \sum_{i \in V} x_i^*$. So the size of S is no more than twice that of an optimal vertex cover.

This approximation algorithm is an example of *deterministic rounding*. While the rounding step here is quite straightforward, there are many other, more sophisticated, types of rounding procedures. In this and later lectures, we will see several examples of *randomized rounding* as well as more involved deterministic rounding schemes based on extreme point structure. (We give a glimpse of how extreme point structure can be exploited in Section 7.) In a later lecture, we will also see how to use linear programs to obtain combinatorial algorithms via the primal-dual method.

1.1 Integrality Gap

A key notion related to accessing the quality of a linear programming relaxation is the concept of an *integrality gap*. Given a set $\mathcal{I}$ of instances for a minimization problem, the integrality gap is defined as:

$$\sup_{I \in \mathcal{I}} \frac{OPT(I)}{OPT_f(I)}.$$

For a maximization problem, it is defined as:

$$\inf_{I \in \mathcal{I}} \frac{OPT(I)}{OPT_f(I)}.$$

The integrality gap establishes a limit on the usefulness of a linear programming relaxation. Specifically, if our approximation algorithm is solely based on a linear program, then we cannot obtain an approximation ratio better than the integrality gap of this linear program. In the case of vertex cover, we can see that the simple deterministic rounding procedure is the best possible for the relaxation (P_{VC}), since the integrality gap of this linear program is 2.

Theorem 1. *The integrality gap of* (P_{VC}) *is at least* $2 - \frac{2}{n}$.

Proof. Consider the complete graph K_n on n vertices. If we set $x_i = \frac{1}{2}$ for all $i \in V$, then the value of OPT_f is $\frac{n}{2}$. However, the size of a minimum vertex cover in K_n is $n - 1$. Thus, the integrality gap is at least $\frac{n-1}{n/2} = 2 - \frac{2}{n}$. □

Finally, to complete our analysis of the simple rounding algorithm for vertex cover, we give an example that illustrates that the analysis of the approximation ratio is tight. If we consider the complete bipartite graph $K_{n,n}$, assigning each $x_i = \frac{1}{2}$ results in a solution that is optimal for (P_{VC}) and whose value equals the value of an optimal integral solution. However, our rounding algorithm will place all of the vertices in the vertex cover, resulting in a set whose size is twice as large as that of an optimal solution.

2 Hitting Set Problems

Many of the discrete optimization problems that we want to (approximately) solve can be viewed as *hitting set* problems. For example, vertex cover falls into this category. Let $\mathcal{U} = \{e_1, \ldots, e_n\}$ denote a universe of n elements, and let $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ denote a family of m sets such that $T_j \subseteq \mathcal{U}$ for all $j = 1, 2, \ldots, m$. Each element in $\mathcal{U}$ is equipped with a cost function $c : \mathcal{U} \to \mathbb{R}^+$. The goal is to find a subset $A \subseteq \mathcal{U}$ with minimum cost $c(A)$ such that $T_j \cap A \neq \emptyset$ for all $j = 1, 2, \ldots, m$. The hitting set problem can be modeled with the following linear program.

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} c(e_i) x_{e_i} \\ \text{subject to:} \quad & \sum_{e_i \in T_j} x_{e_i} \geq 1, \quad \text{for all sets } T_j, \\ & x_{e_i} \geq 0. \end{aligned} \tag{P_{HS}}$$

Suppose that each set T_j has size at most k, i.e. $|T_j| \leq k$. If we consider the solution set $S = \{e_i \in \mathcal{U} \mid x_{e_i} \geq 1/k\}$, then S is a hitting set and the cost of S is at most k times the optimal value of (P_{HS}). Note that this is a generalization of the 2-approximation algorithm we saw for the vertex cover problem in Section 1. Many of the problems we will study in this lecture and the next can be viewed as hitting set problems, e.g. maximum satisfiability, set cover, integer multicommodity flow, shortest s-t-path and minimum cost Steiner forest.

3 Probability Tools for Randomized Rounding

In the *randomized rounding* method, each variable x_i is interpreted as the probability of adding element i to the solution, i.e. of setting $x_i \to 1$ in an integral solution. We consider three examples of approximation algorithms based on this technique and we require some basic probabilistic tools to analyze these algorithms.

(Markov Inequality) If X is a random variable taking on nonnegative values, then:

$$\Pr[X \geq a] \quad \leq \quad \frac{\mathbb{E}[X]}{a} \text{ for } a > 0.$$

(Chernoff Bound) Let $X_1, X_2, \ldots, X_n$ be n independent 0-1 random variables, not necessarily identically distributed. Define $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$. Then for an upper limit U such that $U \geq \mu$ and a small positive constant $\delta > 0$:

$$\Pr[X \geq (1+\delta)U] \quad < \quad e^{-\frac{U\delta^2}{3}}.$$

4 Maximum Satisfiability

In the maximum satisfiability problem, we are given a formula in conjunctive normal form and the goal is to find a boolean assignment for the variables that satisfies the maximum number of clauses.

Input: A formula F on n variables, $\{x_1, x_2, \ldots, x_n\}$, and m clauses, $\{C_1, \ldots, C_m\}$. Each clause C_j has a nonnegative weight w_j and consists of some subset of positive and negative variables (a.k.a. literals). For example, $F = (x_1 \vee \bar{x_2}) \wedge (x_3) \wedge (x_1 \vee x_2 \vee \bar{x_3}) \wedge \ldots$. (We assume that each clause contains only one copy of each variable.)

Output: An assignment $x_i \in \{0, 1\}$ for each variable maximizing the total weight of the satisfied clauses.

The maximum satisfiability problem can be viewed as a hitting set problem. We can construct a set system whose sets correspond to the clauses in F as well as pairs of literals $\{x_i, \bar{x_i}\}$. A satisfying assignment corresponds to one in which at least one literal from each of these sets is included in the hitting set. We note that the maximum satisfiability problem is NP-hard since it generalizes the decision problem 3-SAT: If m is the maximum number of clauses that can be satisifed, then the formula is satisfiable. If each clause contains exactly two literals, then this problem is known as MAX-2-SAT and is NP-hard (even though the decision problem has an efficient algorithm). We now consider a simple randomized algorithm for the maximum satisfiability problem.

ALGORITHM 1

For $i = 1$ to n:

Set each variable $x_i \to 1$ with probability $\frac{1}{2}$.

The expected value of the assignment output by the algorithm is:

$$\mathbb{E}[W] \quad = \quad \sum_{j=1}^{m} w_j \cdot \Pr[C_j \text{ is satisfied}].$$

Lemma 2. *If the clause C_i has size k (i.e. k variables), then:*

$$\Pr[C_j \text{ is satisfied}] = w_j(1 - \frac{1}{2^k}).$$

Proof. $\Pr[C_j \text{ is not satisfied}] = \frac{1}{2^k}$. So $\Pr[C_j \text{ is satisfied}] = 1 - \frac{1}{2^k}$. □

If all clauses have size ≥ 2, ALGORITHM 1 has an approximation guarantee of $\frac{3}{4}$. But can we achieve an approximation ratio of $\frac{3}{4}$ when there are also unit clauses? ALGORITHM 1 has good performance when the clause lengths are long. We now consider another algorithm that performs well when the clauses are short and show that combining these two algorithms yields a $\frac{3}{4}$-approximation algorithm.

4.1 LP Relaxation for Maximum Satisfiability

For each clause C_j, we let S_j^+ denote the positive variables and S_j^- denote the negated variables. The following integer program exactly models (i.e. is equivalent to) the maximum satisfiability problem.

$$\begin{aligned} & \max \sum_{j=1}^{m} w_j z_j \\ \text{subject to: } & \sum_{i \in S_j^+} y_i + \sum_{i \in S_j^-} (1 - y_i) \geq z_j, \quad \text{for } j = 1, 2, \ldots, m, \\ & y_i, z_j \in \{0, 1\}. \end{aligned}$$

We consider the corresponding linear programming relaxation. Now the variable z_j denotes the fraction of clause C_j that is satisfied and y_i is the fractional truth value of variable x_i.

$$\begin{aligned} & \max \sum_{j=1}^{m} w_j z_j \\ \text{subject to: } & \sum_{i \in S_j^+} y_i + \sum_{i \in S_j^-} (1 - y_i) \geq z_j, \quad \text{for } j = 1, 2, \ldots, m, \\ & 0 \leq y_i \leq 1, \\ & 0 \leq z_j \leq 1. \end{aligned} \tag{$P_{max-sat}$}$$

ALGORITHM 2:

Find an optimal solution $(\mathbf{y}^*, \mathbf{z}^*)$ for $(P_{max-sat})$.
For $i = 1$ to n:

Set each variable $x_i \to 1$ with probability y_i^*.

To analyze the performance of ALGORITHM 2, we need some useful inequalities.

Fact 3. [**Arithmetic-geometric mean inequality**] *For any nonnegative $a_1, \ldots, a_k$,*

$$\left(\prod_{i=1}^{k} a_i \right)^{\frac{1}{k}} \quad \leq \quad \frac{1}{k} \sum_{i=1}^{k} a_i.$$

Fact 4. *If a function $f(x)$ is concave on the interval $[0,1]$ (that is, $f''(x) \leq 0$ on $[0,1]$), and $f(0) = a$ and $f(1) = b + a$, then $f(x) \geq bx + a$ for $x \in [0,1]$.*

Theorem 5. ALGORITHM 2 *is a $(1 - \frac{1}{e})$-approximation for maximum satisfiability.*

Proof. What is the probability that clause C_j is satisfied by the assignment given by ALGORITHM 2? Let ℓ_j denote the length of clause C_j.

$$\begin{aligned}
\Pr[C_j \text{ is not satisfied}] &= \prod_{i \in S_j^+} (1 - y_i^*) \prod_{i \in S_j^-} y_i^* \\
&\leq \left[\frac{1}{\ell_j} \left(\sum_{i \in S_j^+} (1 - y_i^*) + \sum_{i \in S_j^-} y_i^* \right) \right]^{\ell_j} \qquad (1) \\
&= \left[1 + \frac{1}{\ell_j} \left(-\ell_j + \sum_{i \in S_j^+} (1 - y_i^*) + \sum_{i \in S_j^-} y_i^* \right) \right]^{\ell_j} \\
&= \left[1 - \frac{1}{\ell_j} \left(\sum_{i \in S_j^+} y_i^* + \sum_{i \in S_j^-} (1 - y_i^*) \right) \right]^{\ell_j} \\
&\leq \left[1 - \frac{1}{\ell_j} z_j^* \right]^{\ell_j}.
\end{aligned}$$

Line (1) follows from Fact 3. Thus, we have:

$$\Pr[C_j \text{ is not satisfied}] \leq \left[1 - \frac{1}{\ell_j} z_j^* \right]^{\ell_j}.$$

This implies:

$$\begin{aligned}
\Pr[C_j \text{ is satisfied}] &\geq 1 - \left[1 - \frac{1}{\ell_j} z_j^* \right]^{\ell_j} \\
&\geq \left[1 - \left(1 - \frac{1}{\ell_j} \right)^{\ell_j} \right] z_j^*. \qquad (2)
\end{aligned}$$

Line (2) follows from Fact 4. So we can compute the expected value of the solution:

$$\begin{aligned}
\mathbb{E}[W] &= \sum_{j=1}^{m} w_j \cdot \Pr[C_j \text{ is satisfied}] \\
&\geq \sum_{j=1}^{m} w_j \cdot z_j^* \left[1 - \left(1 - \frac{1}{\ell_j} \right)^{\ell_j} \right] \\
&\geq \min_{k \geq 1} \left[1 - \left(1 - \frac{1}{k} \right)^{k} \right] \sum_{j=1}^{m} w_j \cdot z_j^* \\
&\geq \min_{k \geq 1} \left[1 - \left(1 - \frac{1}{k} \right)^{k} \right] \cdot OPT.
\end{aligned}$$

Using the fact that $1 + x \leq e^x$ for all x, we have:

$$\min_{k \geq 1} \left[1 - \left(1 - \frac{1}{k} \right)^k \right] \quad \geq \quad \left(1 - \frac{1}{e} \right),$$

which implies that $\mathbb{E}[W] \geq (1 - \frac{1}{e}) \cdot OPT$. □

Now we combine both algorithms. Namely, we run both algorithms and output the assignment that satisfies the most clauses. To analyze this, we consider the expected value of an assignment if we run each algorithm with probability $\frac{1}{2}$. (Note that we can run ALGORITHM 1 with probability $\alpha \geq 0$ and ALGORITHM 2 with probability $\beta \geq 0$ as long as $\alpha + \beta = 1$.) Let W_1 and W_2 denote the outputs of ALGORITHM 1 and ALGORITHM 2, respectively.

$$\begin{aligned} \mathbb{E}\left[\max\{W_1, W_2\}\right] &\geq \mathbb{E}\left[\frac{W_1}{2} + \frac{W_2}{2}\right] = \mathbb{E}\left[\frac{W_1}{2}\right] + \mathbb{E}\left[\frac{W_2}{2}\right] \\ &\geq \sum_{j=1}^{m} w_j \cdot z_j^* \left[\frac{1}{2}\left(1 - \left(1 - \frac{1}{\ell_j} \right)^{\ell_j} \right) + \frac{1}{2}\left(1 - \frac{1}{2^{\ell_j}} \right) \right]. \end{aligned}$$

Proving the following claim proves that the best of the two algorithms is a $\frac{3}{4}$-approximation algorithm.

Claim 6.

$$\left[\frac{1}{2}\left(1 - \left(1 - \frac{1}{\ell_j} \right)^{\ell_j} \right) + \frac{1}{2}\left(1 - \frac{1}{2^{\ell_j}} \right) \right] \quad \geq \quad \frac{3}{4}.$$

Proof. Note that ℓ_j is always an integer with value at least one. For $\ell_j = 1$, we can see that the claim is true. Similarly for $\ell_j = 2$. For $\ell_j \geq 3$, the value is actually at least .753. □

Remark: Finally, we note that there is a randomized algorithm achieving an approximation ratio of $\frac{3}{4}$ for the maximum satisfiability problem that does *not* use a linear program [PS11]. There is also a deterministic rounding algorithm that achieves an approximation guarantee of $\frac{3}{4}$ and *does* use a linear program [VZ11]. However, it is not known whether or not there is a $\frac{3}{4}$-approximation algorithm that is both combinatorial and deterministic.

5 Set Cover

We now look at another algorithm based on randomized rounding of a solution to a linear program. In the *set cover* problem we are given a universe of elements and a family of sets on these elements. The goal is to find a minimum cardinality (or cost) collection of sets such that each element occurs in at least one of the chosen sets.

Input: A universe of n elements, $U = \{e_1, e_2, \ldots, e_n\}$, and a family of m sets, $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$, such that $T_j \subseteq U$ for all $j = 1, 2, \ldots, m$. There is a cost function $c : \mathcal{T} \to \mathbb{R}^+$ indicated by $c_j = c(T_j)$.
Output: A minimum cost subset of $\mathcal{S} \subseteq \mathcal{T}$ such that for each $e_i \in U$, there is some set $T_j \in \mathcal{S}$ such that $e_i \in T_j$. The cost of $\mathcal{S}$ is indicated by $c(\mathcal{S}) = \sum_{T_j \in \mathcal{S}} c(T_j)$.

The set cover problem can be viewed as a hitting set problem, since for each element, there is an associated collection of sets and we must choose at least one of these sets. We can model this problem with the following integer program. Let $y_j \in \{0, 1\}$ denote whether or not set T_j is included in the set

cover

$$\begin{aligned} \min & \sum_{T_j \in \mathcal{T}} c_j y_j \\ \text{subject to:} & \sum_{j: e \in T_j} y_j \geq 1, \quad \text{for all } e \in U \\ & y_j \in \{0, 1\}. \end{aligned}$$

Now let us consider the linear programming relaxation in which y_j now denotes the fractional amount of set T_j included in the optimal fractional solution. We can assume that $y_j \leq 1$ since otherwise we can show the solution is not a minimum cost solution.

$$\begin{aligned} \min & \sum_{T_j \in \mathcal{T}} c_j y_j \\ \text{subject to:} & \sum_{j: e \in T_j} y_j \geq 1, \quad \text{for all } e \in U \\ & y_j \geq 0. \end{aligned} \tag{$P_{set-cover}$}$$

If each element is in at most f sets, then $\mathcal{S} = \{T_j \mid y_j \geq \frac{1}{f}\}$ is a feasible set cover and choosing this set cover is f-approximation algorithm.

5.1 Randomized Rounding for Set Cover

We consider the following rounding algorithm.

> RANDOMIZED-ROUNDING SET COVER 1
>
> Find an optimal solution ($\mathbf{y}^*$) for ($P_{set-cover}$).
> For $j = 1$ to m:
>
> Include each set T_j in the solution set $\mathcal{S}$ with probability y_j^*.

The cost of this solution is exactly $\sum_{T_j \in \mathcal{T}} c_j y_j$. However, this solution may not be feasible. In other words, there may be some elements in U that are covered by any set in $\mathcal{S}$. However, we can prove that each element has a constant probability of being covered by some set.

Lemma 7. RANDOMIZED-ROUNDING SET COVER 1 *covers each element with probability at least* $(1 - \frac{1}{e})$.

Proof. Consider element e_i which belongs to k sets, each of which is selected by the algorithm with probability $p_1, p_2, \ldots, p_k$. By the constraints from ($P_{set-cover}$), we have:

$$\sum_{j=1}^{k} p_j \quad \geq \quad 1.$$

Next, we have:

$$\begin{aligned} \Pr[e_i \text{ is not covered}] &= \prod_{j=1}^{k} (1 - p_j) \\ &\leq \prod_{j=1}^{k} e^{-p_j} = e^{-\sum p_j} \leq \frac{1}{e}. \end{aligned}$$

So, $\Pr[e_i \text{ is covered}] \geq 1 - \frac{1}{e}$. □

Now we modify our algorithm: we include each set with probability p_j and repeat this $2\log n$ times (we can abort if set T_j is chosen).

> RANDOMIZED-ROUNDING SET COVER 2
>
> Find an optimal solution $(\mathbf{y}^*)$ for $(P_{set-cover})$.
> For $j = 1$ to m:
>
> Repeat $2\log n$ times: Include each set T_j in the solution set $\mathcal{S}$ with probability y_j^*.

Lemma 8. RANDOMIZED-ROUNDING SET COVER 2 *covers all elements with probability at least* $1 - \frac{1}{n}$.

Proof.

$$\begin{aligned}
\Pr[e_i \text{ is not covered}] &= \prod_{j:e_i \in T_j} (1 - y_j^*)^{2\log n} \\
&\leq \prod_{j:e_i \in T_j} e^{-y_j^* 2\log n} \\
&= e^{-2\log n \sum_{j:e_i \in T_j} y_j^*} \\
&\leq \left(\frac{1}{e}\right)^{2\log n} = \frac{1}{n^2}.
\end{aligned}$$

By union bound, we can show:

$$\Pr[\text{exists } e_i \text{ that is not covered}] \leq \frac{1}{n^2} \cdot n = \frac{1}{n}. \qquad \text{(Bad event I)}$$

So we have:

$$\Pr[\text{all elements are covered}] \geq 1 - \frac{1}{n}.$$

□

We are not done, because there is some probability that even if all elements are covered, the cost exceeds the expected cost. Let $OPT_f = \sum_{T_j \in \mathcal{T}} c_j y_j^*$. By Markov's inequality, we have:

$$\Pr[c(\mathcal{S}) > 4(OPT_f \cdot 2\log n)] \leq \frac{1}{4}. \qquad \text{(Bad event II)}$$

By union bound the probability that either bad events I or II can occur is at most $\frac{1}{4} + \frac{1}{n} < \frac{1}{3}$ for sufficiently large n. Thus, with a constant probability, we find a solution that is (i) a valid set cover and (ii) costs at most $8\log n \cdot OPT$. We can conclude that RANDOMIZED-ROUNDING SET COVER 2 is a $O(\log n)$-approximation algorithm. There are instances for the set cover problem exhibiting an integrality gap of $\Omega(\log n)$.

6 Integer Multicommodity Flows

Given a graph $G = (V, E)$, and k pairs $(s_i, t_i) \in V$ for $i = 1, 2, \ldots, k$, the goal of the *integer multicommodity flow* problem is to find a simple path from s_it to t_i for all $i = 1, 2, \ldots, k$ so as to minimize the maximum number of paths using any particular edge. In other words, the objective is to minimize the maximum edge congestion.

Let P_i denote the set of all simple paths from s_i to t_i for $i = 1, 2, \ldots, k$. For path p, let p also denote the set of edges in p. Let $x_p \in \{0, 1\}$ be a variable indicating whether or not path p is chosen or not. The total number of paths using an edge e is given by:

$$\sum_{p:e\in p} x_p \quad \leq \quad C.$$

Then C is an upper bound on the congestion of edge e. For each (s_i, t_i) pair, we need to choose a path:

$$\sum_{p\in P_i} x_p \quad = \quad 1.$$

These constraints lead to the following linear programming relaxation.

$$\begin{aligned} &\min\ C \\ &\sum_{p\in P_i} x_p = 1, \quad \text{for all } i = 1, \ldots, k, \\ &\sum_{p:e\in p} x_p \leq C, \quad \text{for all } e \in E, \\ &x_p \geq 0. \end{aligned} \tag{P^1_{flow}}$$

This linear program may have an exponential number of variables. Consider another linear programming relaxation with a polynomial number of variables. In this relaxation, x_{ie} represents the number of paths using edge e.

$$\begin{aligned} &\min\ C \\ &\sum_{e\in\delta^+(v)} x_{ie} = \sum_{e\in\delta^-(v)} x_{ie}, \quad \text{for all } i = 1, \ldots, k \text{ and } v \neq s_i, t_i, \\ &\sum_{e\in\delta^-(s_i)} x_{ie} = \sum_{e\in\delta^+(t_i)} x_{ie} = 1, \quad \text{for all } i = 1, \ldots, k, \\ &\sum_{i=1}^{k} x_{ie} \leq C, \quad \text{for all } e \in E, \\ &x_{ie} \geq 0. \end{aligned} \tag{P^2_{flow}}$$

It can be shown that relaxations (P^1_{flow}) and (P^2_{flow}) are equivalent in the sense that an optimal solution for (P^1_{flow}) can be converted to an optimal solution for (P^2_{flow}) and vice versa.

6.1 Randomized Rounding for Multicommodity Flow

We will analyze the following rounding procedure for the multicommodity flow problem.

> Randomized-Rounding Multicommodity Flow
>
> Find an optimal solution $(C^*, \mathbf{x}^*)$ for (P^1_{flow}).
> For $i = 1$ to k:
>
> Choose one path $p \in P_i$, such that each $p \in P_i$ is chosen with probability x_p.

Theorem 9. *With high probability, the total number of paths using an edge e is at most $O(\log n \cdot C)$.*

Proof. For each edge $e \in E$, define a random variable X_e^i where:

$$\begin{aligned} X_e^i &= 1 \text{ if } s_i \to t_i \text{ path uses edge } e, \text{ or} \\ X_e^i &= 0 \text{ otherwise.} \end{aligned}$$

The number of edges using edge e is denoted by the random variable Y_e, where

$$Y_e = \sum_{i=1}^{k} x_e^i.$$

Our goal is to show that with high probability, the value of Y_e is not much greater than C^*. The expectation of Y_e is at most C^*:

$$\mathbb{E}[Y_e] = \sum_{i=1}^{k} \sum_{\substack{p \in P_i: \\ e \in p}} x_p^* = \sum_{p: e \in p} x_p^* \leq C^*.$$

For a fixed edge e, the random variables $\{X_e^i\}$ are independent. Thus, we can apply the Chernoff Bound from Section 3. Let $\delta = 1$ and let $U = 12 \log n$. Then we have:

$$\Pr[Y_e \geq 24 \cdot \log n] < e^{-\frac{12 \log n}{3}} < \frac{1}{n^4}.$$

Applying the union bound over all (at most n^2) edges, we have:

$$\Pr[\text{Congestion } \geq 24 \cdot \log n] = \Pr[\exists e : Y_e > 24 \cdot \log n] \leq \frac{1}{n^2}.$$

We conclude that:

$$\Pr[\text{Congestion } < 24 \cdot \log n] \geq 1 - \frac{1}{n^2}.$$

□

So with high probability, the RANDOMIZED-ROUNDING MULTICOMMODITY FLOW algorithm will output a feasible solution with edge congestion no more than $O(\log n)$ times the optimal congestion.

7 Extreme Point Structure

Many deterministic rounding algorithms crucially use the extreme point structure of an optimal solution for a linear program. For example, we can show that an extreme point for the vertex cover linear program (P_{VC}) is *half-integral*. That is, each vertex has a value $v_i \in \{0, 1/2, 1\}$. This property can be used to obtain an algorithm for the vertex cover problem that has an approximation ratio less than 2 for graphs that can be colored with few colors.

Theorem 10. *Given a k-coloring of a graph G, we can find a vertex cover which is at most $(2 - \frac{2}{k})$ times an optimal solution.*

Proof. Given an optimal extreme point solution $\mathbf{x}^*$ to (P_{VC}), consider the sets:

$$V_0 = \{v \mid x_v = 0\}, \quad V_{\frac{1}{2}} = \{v \mid x_v = \frac{1}{2}\}, \quad V_1 = \{v \mid x_v = 1\}.$$

Then there will be no edges between any pair of vertices in V_0, or between a vertex in V_0 and $V_{\frac{1}{2}}$. Furthermore, we know that $V_{\frac{1}{2}}$ can be colored with k colors, which means that there are k independent

sets in $V_{\frac{1}{2}}$. Let I be the independent set of $V_{\frac{1}{2}}$ that has the maximum cardinality (or maximum sum of weights in the weighted case). Rounding x_i down for every $i \in I$, and rounding up x_i for every $i \notin I$, will yield a feasible solution of cost at most:

$$2 \cdot \sum_{v \in V_{\frac{1}{2}} \setminus I} x_v + |V_1| \;\leq\; (1 - \frac{1}{k}) \cdot |V_{\frac{1}{2}}| + |V_1|.$$

The solution $\mathbf{x}^*$ has (optimal) value:

$$\sum_{v \in V} x_v^* \;\;=\;\; \frac{1}{2}|V_{\frac{1}{2}}| + |V_1|.$$

Thus, our solution is at most $(2 - \frac{2}{k})$ times the value of an optimal solution. □

We will see other, more elaborate, examples of how to use extreme point structure for designing approximation algorithms in future lectures.

References

[GW94] Michel X. Goemans and David P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7(4):656–666, 1994.

[PS11] Matthias Poloczek and Georg Schnitger. Randomized variants of Johnson's algorithm for MAX SAT. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 656–663. SIAM, 2011.

[RT87] Prabhakar Raghavan and Clark D. Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[Vaz13] Vijay V. Vazirani. *Approximation Algorithms*. Springer Science & Business Media, 2013.

[VZ11] Anke Van Zuylen. Simpler 3/4-approximation algorithms for MAX SAT. In *International Workshop on Approximation and Online Algorithms*, pages 188–197. Springer, 2011.

[WS11] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

These lecture notes are based on Chapters 1 and 5 of [WS11], Chapter 4 of [Vaz13], lecture notes by Stéphan Thomassé from previous versions of the same course and lecture notes from a course on approximation algorithms at EPFL (`http://theory.epfl.ch/osven/courses/Approx13`). The original presentations of the algorithms for maximimum satisfiability and integer multicommodity flow can be found in [GW94] and [RT87], respectively.