approximation factor. In the analysis that we did today we used all three of the local improvement steps to argue that the locally optimal solution is close to the global one. If we ended up not using one we could have gotten by without using it in our algorithm.

Facility location problems come in many different flavors. One particular flavor is picking out exactly k different locations while minimizing the sum of the routing costs. This has a similar local search algorithm: start with some subset of k locations and perform a series of swaps to get to a locally optimal solution. This algorithm is very general and works in a number of situations, but we can do much better than a 3-approximation for facility location using other techniques.

11.2 Linear Programming

11.2.1 Introduction

Linear Programming is one of the most widely used and general techniques for designing algorithms for NP-hard problems.

Definition 11.2.1 *A linear program is a collection of linear constraints on some real-valued variables with a linear objective function.*

Example:

Suppose we have two variables: x,y.
Maximize: $5 \cdot x + y$
Subject to : $2 \cdot x - y \leq 3$
$x - y \leq 2$
$x, y \geq 0$

To get an idea of what this looks like plot the set of pairs (x, y) that satisfy the constraints. See Figure 11.2.4 for a plot of the feasible region.

The feasible region, the region of all points that satisfy the constraints is the shaded region. Any point inside the polytope satisfies the constraints. Our goal is to find a pair that maximizes $5 \cdot x + y$.

The solution is the point $(x, y) = (7, 5)$.

■

Every point in the feasible region is a feasible point, i.e., it is a pair of values that satisfy all the constraints of the program. The feasible region is going to be a polytope which is an n-dimensional volume all faces of which are flat. The optimal solution, either a minimum or a maximum, always occurs at a corner of the polytope (feasible region). The extreme points (corners) are also called basic solutions.

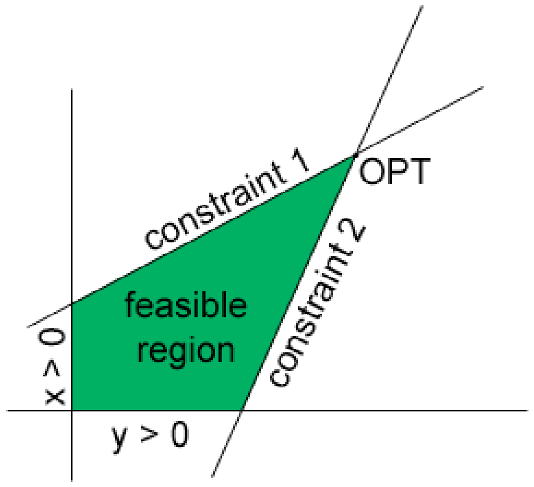

Figure 11.2.4: The feasible region in some linear programming problem.

11.2.2 Solving a Linear Program

Linear programs can be solved in polynomial time. If you have n variables and m constraints, then the linear program can be solved in time polynomial in n and m.

One way to solve a linear program is to:

1. Enumerate all extreme points of the polytope.
2. Check the values of the objective at each of the extreme points.
3. Pick the best extreme point.

This algorithm will find the optimal solution, but has a major flaw because there could be an exponential number of extreme points. If you have n variables, then the polytope is in some n-dimensional space. For example, say the polytope is an n-dimensional hypercube, so all of the variables have constraints such that $0 \leq x_i \leq 1$. There are 2^n extreme points of the hypercube, so we can't efficiently enumerate all the possible extreme points of the polytope and then check the function values on each of them.

What is typically done to find the solution is the Simplex Method. The Simplex Method starts from some extreme point of the polytope, follows the edges of the polytope from one extreme point to another doing hill climbing of some sort and then stops when it reaches a local optimum. The "average case complexity" of this algorithm is polynomial-time, but on the worst case it is exponential. There are other algorithms that are polynomial-time that follow points in the polytope until you find an optimal solution. The benefit of Linear Programming is that once you write down the constraints, there exists some polynomial-time algorithm that is going to solve the problem.

Linear Programming is interesting to us because 1. it is solvable in polynomial time and 2. a closely related problem (Integer Programming) is NP-hard. So we can take any NP-complete problem, reduce it to an instance of Integer Programming, and then try to solve the new problem using the same techniques that solve Linear Programming problems.

11.2.3 Integer Programming

Definition 11.2.2 *An Integer Program is a Linear Program in which all variables must be integers.*

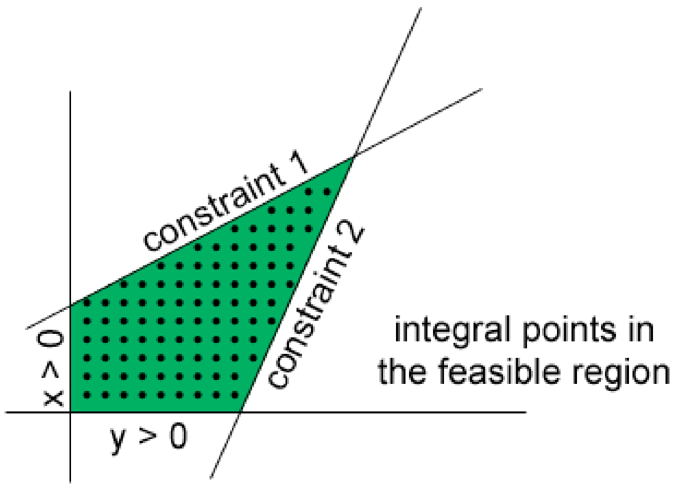

Figure 11.2.5: The feasible region of an Integer Programming problem.

In an Integer Program you are trying to optimize some linear function over some set of linear constraints with the additional constraint that all the variables must be integers. If we were to solve Example 11.2.1 subject to the constraint that x and y are integers, we would get the same optimal solution. This is not generally the case. In general, if you plot the feasible set of an Integer Program, you get some set of points defined by some constraints, and our goal is to optimize over the set of integer points inside the feasible region. A Linear Program will optimize over the larger set which contains the integer points and the real-valued points inside the feasible region. In general, the optimum of the Integer Program will be different from the optimum of the corresponding Linear Program. With an Integer Program, the list of constraint again form a polytope in some n-dimensional space, but the additional constraint is that you are only optimizing over the integer points inside this polytope. Note: The integer points in the feasible region do not necessarily lie on the boundaries and the extreme points, in particular, are not necessarily integer points. A Linear Programming algorithm is going to find an extreme point as a solution to the problem. This is not necessarily the same as the optimal solution to the Integer Program, but hopefully it is close enough.

11.2.4 Solving NP-hard Problems

In general, the way we use Linear Programming is to

1. reduce an NP-hard optimization problem to an Integer Program
2. relax the Integer Program to a Linear Program by dropping the integrality constraint
3. find the optimal fractional solution to the Linear Program
4. round the optimal fractional solution to an integral solution

Note: In general the optimal fractional solution to the Linear Program will not be an integral solution. Thus the final integral solution is not necessarily an optimal solution, but it came from an optimal fractional solution, so we hope it is close to an optimal integral solution.

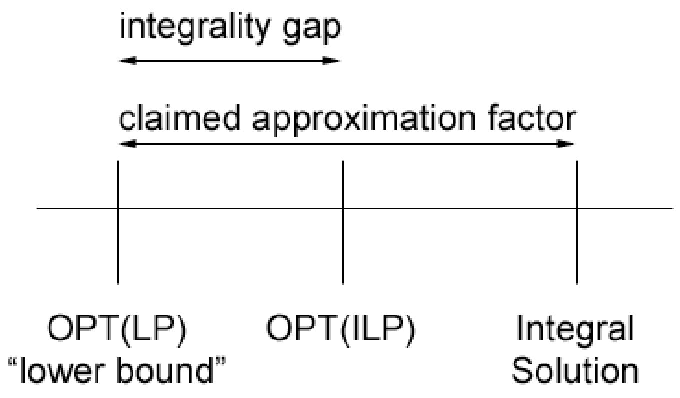

Figure 11.2.6: The range of solutions for Linear vs. Integer Programming.

In terms of the values of the solutions, the optimal value of the Integer Program which is equal to the solution of the NP-hard problem falls in between the optimal solution to the Linear Program and the integral solution determined from rounding the optimal solution to the Linear Program. See Figure 11.2.6 for an example of a minimization problem. The gap between the optimal solution to the Integer Program and the Linear Program is known as the integrality gap. If this gap is large, then we can expect a large approximation factor. If this gap is small, then the approximation factor is likely to be small. The integrality gap characterizes if we relax integrality, what is the improvement in the optimal solution. If we are using this particular algorithm for solving an NP-hard problem we cannot prove an approximation factor that is better than the integrality gap. The approximation factor is the difference between the optimal solution to the Linear Program and the integral solution that is determined from that optimal fractional solution. When we do the transformation from the NP-hard problem to an Integer Program our goal is to come up with some

Integer Program that has a low integrality gap and the right way to do the rounding step from the fractional solution to the integral solution. Note: Finding a fractional optimal solution to a Linear Program is something that is known (and can be done in polynomial-time), so we can take it as given.

The good thing about Linear Programs is that they give us a lower bound to any NP-hard problem. If we take the NP-hard problem and reduce it to an Integer Program, relax the Integer Program to a Linear Program and then solve the Linear Program, the optimal solution to the Linear Program is a lower bound to the optimal solution for the Integer Program (and thus the NP-hard problem). This is a very general way to come up with lower bounds. Coming up with a good lower bound is the most important step to coming up with a good approximation algorithm for the problem. So this is a very important technique. Most of the work involved in this technique is coming up with a good reduction to Integer Programming and a good rounding technique to approximate the optimal fractional solution.

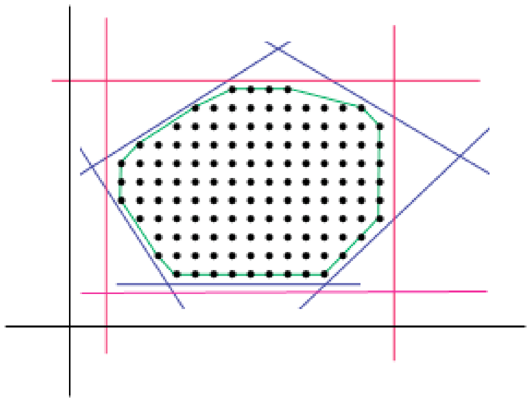

Figure 11.2.7: The possible constraints on an ILP problem.

Note: There are multiple ways to enclose a given set of integer solutions using linear constraints. See Figure 11.2.7. The smallest that encloses all feasible integer solutions without too many constraints and without enclosing any more integer points is best. It is an art to come up with the right set of linear constraints that give a low integrality gap and a small number of constraints.

11.2.5 Vertex Cover Revisited

We have already seen a factor of 2 approximation using maximum matchings for the lower bound. Today we will see another factor of 2 approximation based on Linear Programming. There is yet another factor of 2 approximation that we will see in a few lectures. For some problems many techniques work.

Given: $G = (V, E)$ with weights $w : V \to \mathbb{R}^+$

Goal: Find the minimum cost subset of vertices such that every edge is incident on some vertex in that subset.

1. Reducing Vertex Cover to an Integer Program

Let the variables be:

x_v for $v \in V$ where $x_v = 1$ if $v \in VC$ and $x_v = 0$ otherwise.

Let the constraints be:

For all $(u, v) \in E$, $x_u + x_v \geq 1$ (each edge has at least one vertex)

For all $v \in V$, $x_v \in \{0, 1\}$ (each vertex is either in the vertx cover or not)

We want to minimize $\sum_{v \in V} w_v \cdot x_v$ (the total weight of the cover)

Note that all the constraints except the integrality constraints are linear and the objective function is also linear because the w_v are constants given to us. Thus this is an Integer Linear Program.

Note: This proves that Integer Programming is NP-hard because Vertex Cover is NP-hard and we reduced Vertex Cover to Integer Programming.

2. Relax the Integer Program to a Linear Program

Now relax the integrality constraint $x_v \in \{0, 1\}$ to $x_v \in [0, 1]$. to obtain a Linear Program.

3. Find the optimal fractional solution to the Linear Program

Say x^* is the optimal fractional solution to the Linear Program.

Lemma 11.2.3 *$Val(x^*) \leq OPT$ where OPT is the optimal solution to the Integer Program.*

Example:

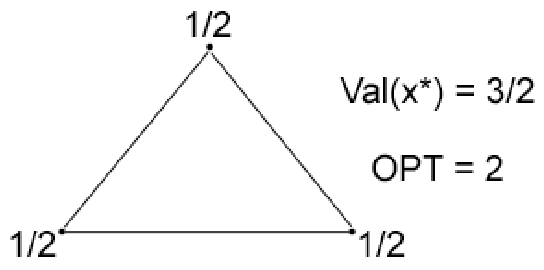

Figure 11.2.8: A possible LP soltion to the vertex cover problem.

Consider a graph that is just a triangle with $w_v = 1$ for all $v \in V$. See Figure 11.2.8. Our Linear Program is looking for fractional values at each of the vertices such that the fractional values for the vertices on each edge sum up to at least one and the sum of all of the fractional values times their vertex weight is minimized.

One fractional solution is to assign $x_v^* = \frac{1}{2}$ for all $v \in V$. Note: This fractional solution is also optimal. In this case, $Val(x^*) = \frac{3}{2}$.

The optimal solution to vertex cover is going to have to include at least two vertices. So $OPT = 2$. We can see that the Linear Program is doing better than the Integer Program.

■

4. Round the optimal fractional solution to an integral solution.

Given an optimal fractional solution, we clearly want to include all $x_v^* = 1$ in the vertex cover and we don't want to include any $x_v^* = 0$ in the vertex cover. Further we can obtain an integral solution by picking all vertices that are at least $\frac{1}{2}$. This covers all edges because for every edge $e = (u, v)$ $x_u^* + x_v^* \geq 1$ which implies that either x_u^* or x_v^* is at least $\frac{1}{2}$ and so will be picked for the vertex cover.

Algorithm: Return all vertices v with $x_v^* \geq \frac{1}{2}$.

Theorem 11.2.4 *This algorithm is a 2-approximation.*

Proof: Theorem 11.2.4 $\tilde{x_v} = 1$ if the algorithm picks v and $\tilde{x_v} = 0$ otherwise. Let x^* be the optimal fractional solution to the Linear Program. This implies that $\tilde{x_v} \leq 2 \cdot x_v$ for all $v \in V$ because some subset of the vertices was doubled and the rest were set to 0.

The algorithm's cost is $\sum_{v \in V} w \cdot \tilde{x_v}$
$$\begin{aligned} &\leq 2 \cdot \textstyle\sum_{v \in V} w_v \cdot x_v^* \\ &= 2 \cdot cost(LinearProgram) \\ &\leq 2 \cdot OPT. \end{aligned}$$

■

This proves that this is a 2-approximation, so the integrality gap is at most 2. In Example 11.2.5 the integrality gap is at least $\frac{4}{3}$ which is less than 2. For Vertex Cover with this Linear Program, the integrality gap can be arbitrarily close to 2. As an exercise, the reader should come up with an instance of vertex cover where the integrality gap is indeed very close to 2.

11.3 Next Time

Next time we will see some more examples of Linear Programming and in particular how to do the rounding step.

CS787: Advanced Algorithms

Scribe: Archit Gupta, Pavan Kuppili **Lecturer:** Shuchi Chawla
Topic: Randomized rounding, concentration bounds **Date:** 10/10/2007

12.1 Set Cover

Let $E = \{e_1, e_2, .., e_n\}$ be a set of n elements. Let $S_1, S_2, .., S_m$ be subsets of E with associated costs $c_1, .., c_m$. As explained in a previous lecture, the problem of set cover is to choose 1 or more of the m subsets such that the union of them cover every element in E, and the objective is to find the set cover with the minimum cost. In a previous lecture, we have seen a greedy solution to the set cover which gave a log n approximation. In this lecture, we will see a linear programming solution with randomized rounding to achieve an $O(logn)$ approximation.

12.1.1 ILP formulation of set cover

Let X_i be a random variable associated with each subset S_i.
$X_i = 1$, if S_i is in the solution, and 0 otherwise.

The ILP formulation:
Minimize $\sum_{i=1}^{m} c_i X_i$ s.t
$\forall e \in E, \sum_{i:e \in S_i} X_i \geq 1$ - (1)
$\forall X_i, X_i \in \{0, 1\}$ - (2)

The constraints in (1) ensure that every element is present in atleast one of the chosen subsets. The constraints in (2) just mean that every subset is either chosen or not. The objective function chooses a feasible solution with the minimum cost.

It can be noted that this problem formulation is a generalization of the vertex cover. If we map every edge to an element in E, and every vertex to one of the the subsets of E, the vertex cover is the same as the set cover problem with the additional constraint that every element e appears in exactly 2 subsets (corresponding to the 2 vertices it is incident on).

12.1.2 Relaxing to an LP problem

Instead of $X_i \in \{0, 1\}$, relax the constraint so that X_i is in the range [0, 1]. The LP problem can be solved optimally in polynomial time. Now we need to round the solution back to an ILP solution.

12.1.3 Deterministic rounding

First, let us see the approximation we get by using a deterministic rounding scheme analogous to the one we used in the vertex cover problem.

Theorem 12.1.1 *We have a deterministic rounding scheme which gives us an F-approximation* *to the solution, where F is the maximum frequency of an element.*

Proof: Let F be the maximum frequency of an element, or the maximum number of subsets an element appears in.
Pick a threshold t = $\frac{1}{F}$.
If $x_1, .., x_m$ is the solution to the LP, the ILP solution is all subsets for which $x_i \geq \frac{1}{F}$.

This results in a feasible solution because in any of the constraints in (1) we have atmost F variables on the LHS, and atleast one of them should be $\geq \frac{1}{F}$, so we will set atleast one of those variables to 1 during the rounding. So, every element will be covered. It can be noted here that the vertex cover has F = 2, and hence we used a threshold of $\frac{1}{2}$ during the previous lecture. If the values of $X_1, .., X_m$ are $x_1^{'}, .., x_m^{'}$ after rounding, $\forall i, x_i^{'} \leq Fx_i$ [since if $x_i \geq \frac{1}{F}$ we round it to 1, and 0 otherwise]
So the cost of the ILP solution $\leq$ F(cost of the LP solution). So, this deterministic rounding gives us an F approximation to the optimal solution. ■

12.1.4 Randomized rounding

Algorithm:
Step 1. Solve the LP.
Step 2 (Randomized rounding). ∀ i, pick S_i independently with probability x_i (where x_i is the value of X_i in the LP solution).
Step 3. Repeat step 2 until all elements are covered.

The intution behind the algorithm is that higher the value of x_i in the LP solution, the higher the probability of S_i being picked during the random rounding.

Theorem 12.1.2 *With a probability* $(1 - \frac{1}{n})$*, we get a 2logn approximation.*

Proof:

Lemma 12.1.3 *The expected cost in each iteration of step 2 is the cost of the LP solution.*

Proof: Let Y_i be a random variable.
$Y_i = c_i$ if S_i is picked, and 0 otherwise.
Let Y = $\sum_{i=1}^{n} Y_i$,
E[Y] = $\sum_{i=1}^{n} E[Y_i]$ [by linearity of expectation]
= $\sum_{i=1}^{n} c_i x_i$
= cost of the LP solution. ■
This is a nice result as the expected cost of the ILP solution is exactly equal to the cost of the LP solution.

Lemma 12.1.4 *The number of iterations of step 2 is 2logn with a high probability (whp).*

Proof: Fix some element e $\in$ E.
$\mathbf{Pr}$[e is not covered in any one execution of step] = $\prod_{i:e\in S_i}\mathbf{Pr}[S_i$ is not picked] (since the subsets are picked independently)
$= \prod_{i:e\in S_i}(1-x_i)$
$\leq \prod_{i:e\in S_i} e^{-x_i}$ (using the fact that if $x\in[0,1], (1-x)\leq e^{-x}$)
$= e^{-\sum_{i:e\in S_i} x_i}$
$\leq \frac{1}{e}$ (since the LP solution satisfies constraint (1) which means $\sum_{i:e\in S_i} x_i \geq 1$)

If we execute step 2 (2logn) times,
$\mathbf{Pr}$[element e is still uncovered] $\leq \frac{1}{e^{2logn}}$ (since each execution is independent)
$= \frac{1}{n^2}$.
By the union bound, $\mathbf{Pr}[\exists$ an uncovered element after 2logn executions] $\leq \frac{1}{n}$
Hence with a high probability, the algorithm will terminate in 2log n iterations of step 2. ■

The total expected cost is just the expected cost of each iteration multiplied by the number of iterations.
So E[cost] = (2log n).Cost(LP solution) with a probability $(1-\frac{1}{n})$ (using Lemmas 12.1.3 and 12.1.4).
■

12.2 Concentration Bounds

In analyzing randomized techniques for rounding LP solutions, it is useful to determine how close to its expectation a random variable is going to turn out to be. This can be done using concentration bounds: We look at the probability that given a certain random variable X, the probability that X lies in a particular range of values (Say, the deviation from the expectation value). For instance, we want $\mathbf{Pr}[X \geq \lambda]$ for some value of λ. Note that if a random variable has small variance, or (as is often the case with randomized rounding algorithms) is a sum of many independent random variables, then we can give good bounds on the probability that it is much larger or much smaller than its mean.

12.2.1 Markov's Inequality

Given a random variable $X \geq 0$, we have,

$$\mathbf{Pr}[X \geq \lambda] \leq \frac{\mathbf{E}[X]}{\lambda} \tag{12.2.1}$$

Also, given some $f : X \rightarrow \Re^+ \cup \{0\}$,

$$\mathbf{Pr}[f(X) \geq f(\lambda)] \leq \frac{\mathbf{E}[f(X)]}{f(\lambda)} \tag{12.2.2}$$

If the above function f is monotonically increasing then in addition to (12.2.2), the following also holds:

$$\mathbf{Pr}[X \geq \lambda] = \mathbf{Pr}[f(X) \geq f(\lambda)] \leq \frac{\mathbf{E}[f(X)]}{f(\lambda)} \tag{12.2.3}$$

12.2.2 Chebyshev's Inequality

We now study a tighter bound called the Chebyshev's bound.
Let $f(X) = (X - \mathbf{E}[X])^2$. Note that f is an increasing function if $X > \mathbf{E}[X]$.
We have,

$$\mathbf{Pr}[|X - \mathbf{E}[X]| \leq \lambda] = \mathbf{Pr}\left[(X - \mathbf{E}[X])^2 \leq \lambda^2\right] \tag{12.2.4}$$

$$= \frac{\mathbf{E}\left[(X - \mathbf{E}[X])^2\right]}{\lambda^2} = \frac{\sigma^2(X)}{\lambda^2} \tag{12.2.5}$$

That is, the deviation of X from $\mathbf{E}[X]$ is a function of its variance($\sigma(X)$). If the variance is small, then we have a tight bound.

Also not that with probability p, $X \in \mathbf{E}[X] \pm \sqrt{\frac{1}{p}}\sigma(X)$

12.2.3 Chernoff's bound

We present Chernoff's bound which is tighter compared to the Chebyshev and Markov bounds.

Let random variables X_1, X_2, ..., X_n be independent and identically distributed random variables, where $X_i \in [0, 1]$. Note that the class of indicator random variables lie in this category.

Let $X = \sum_{i=1}^{n} X_i$. Also $\mathbf{E}[X] = \sum_{i=1}^{n} \mathbf{E}[X_i] = \mu$

For any $\delta > 0$,

$$\mathbf{Pr}[X \geq (1 + \delta)\mu] \leq \left(\frac{e^{\delta}}{(1 + \delta)^{1+\delta}}\right)^{\mu} \tag{12.2.6}$$

It can also be shown for $\delta \geq 0$,

$$\mathbf{Pr}[X \geq (1 + \delta)\mu] \leq e^{-\left(\frac{\delta^2}{2+\delta}\right)\mu} \tag{12.2.7}$$

Note that the above probability resembles a gaussian/bell curve. When $0 \leq \delta \leq 1$

$$\mathbf{Pr}[X \geq (1 + \delta)\mu] \leq e^{-\frac{\delta^2}{3}\mu} \tag{12.2.8}$$

$$\mathbf{Pr}[X \leq (1 - \delta)\mu] \leq e^{-\frac{\delta^2}{2}\mu} \tag{12.2.9}$$

Example: Coin tossing to show utility of bounds. We toss n independent unbiased coins. We want to find a t such that: $\mathbf{Pr}[\text{No. of times we get heads in } n \text{ tosses} \geq t] \leq 1/n$.

We use indicator random variables to solve this: If the ith coin toss is heads, let $X_i = 1$, otherwise $X_i = 0$. Let $X = \sum_{i=1}^{n} X_i$. Here X is the total number of heads we get in n independent tosses. Note that $\mu = \mathbf{E}[X] = \sum_{i=1}^{n} \mathbf{E}[X_i] = n/2$, since $\mathbf{E}[X_i] = 1/2$ (The probability of getting a heads).

To get bounds on $\mathbf{Pr}[X \geq t]$, we first apply Markov's inequality:

$$\mathbf{Pr}[X \geq t] \leq \frac{\mathbf{E}[X]}{t} = \frac{n/2}{t}$$

$$\mathbf{Pr}[X \geq t] = 1/n \Rightarrow \frac{n/2}{t} = 1/n \Rightarrow t = n^2/2$$

But, we don't do any more than n coin tosses, so this bound is not useful. Note that Markov's bound is weak.

Applying Chebyshev's inequality:

$$\mathbf{Pr}[X \geq t] \leq \mathbf{Pr}[|X - \mu| \geq t - \mu] \leq \frac{\sigma^2(X)}{(t-\mu)^2}$$

Evaluating $\sigma(X)$ where $X_i \in \{0, 1\}$,

$$\mathbf{E}[X_i] = 1/2$$

$$\sigma^2(X_i) = \mathbf{E}\big[(X_i - \mathbf{E}[X_i])^2\big] = \frac{1}{2}(\frac{1}{4}) + \frac{1}{2}(\frac{1}{4}) = 1/4$$

Since this is a sum of independent random variables, the variances can be summed together:

$$\sigma^2(X) = \sum_{i=1}^{n} \sigma^2(X_i) = n/4$$

Evaluating t,

$$\mathbf{Pr}[X \geq t] \leq \frac{\sigma^2(X)}{(t-\mu)^2} = 1/n$$

$$\Rightarrow t = \mu + n/2 = n$$

Again this bound is quite weak.

Applying Chernoff's bound:

$$\mathbf{Pr}[X \geq (1+\delta)\mu] \leq e^{-\frac{\delta^2}{3}\mu} = e^{-\frac{\delta^2}{6}n}$$

Taking,

$$e^{-\frac{\delta^2}{6}n} = 1/n$$

we get,

$$\delta = \sqrt{\frac{6logn}{n}}$$

$$\Rightarrow t = (1+\delta)\mu = n/2 + \sqrt{\frac{3nlogn}{2}}$$

This bound is much nicer than what we obtained earlier, showing us this gradual increase in tightness of the bounds across Markov, Chebyshev and Chernoff. ■

CS787: Advanced Algorithms

Scribe: Archit Gupta, Pavan Kuppili **Lecturer:** Shuchi Chawla
Topic: Randomized rounding (contd.), LP duality. **Date:** 10/15/2007

13.1 Routing to minimize congestion

We are given a graph $G = (V, E)$ and k "commodities". Each commodity i has a source s_i and a destination t_i. The goal is to find an $(s_i,\, t_i)$ path in the graph G for every commodity i, while minimizing the maximum congestion along any edge. The latter can be written formally as
Minimize congestion $C = max_e|\{i : P_i \ni e\}|$, where P_i is the $(s_i,\, t_i)$ path of commodity i and e is an edge.

It can be noted that this problem is similar to a network flow problem. But it is not a network flow problem because for each commodity i, we need to have a single path between s_i and t_i. We cannot have the flow splitting along multiple branches.

This problem is an NP-hard problem. So we will solve it by formulating it as an ILP problem, relaxing it to an LP problem, solving the LP, and rounding the solution to an ILP solution.

13.1.1 ILP formulation with exponential number of variables and constraints

Let $i = 1, \ldots, k$ be the k commodities.
Let P_i be the set of all paths from s_i to t_i.
We have a variable x_P for every $P \in P_i$, $\forall i$

Minimze t s.t

$$\sum_{P \in P_i} x_P = 1, \forall i \tag{13.1.1}$$

$$\sum_i \sum_{P \in P_i, e \in P} x_P \leq t, \forall e \in E \tag{13.1.2}$$

$$t \geq 0 \tag{13.1.3}$$

$$x_P \in \{0, 1\} \forall P \tag{13.1.4}$$

Note that the actual objective function which is in a min-max form is not linear. So, we employ a trick of introducing a new variable t. We introduce the constraint (13.1.2) that congestion on any edge $\leq$ t, and so all we need to do now is minimze t, which is equivalent to minimizing the maximum congestion. The constaint 13.1.1 makes sure that we select exactly one path for each commodity.

The problem with this ILP formulation is that we can have an exponential number of paths between s_i and t_i (for example, in a completely connected graph). So even if we relax it to an LP problem, we will still have an exponential number of variables and constraints. We would like a formulation with a polynomial (polynominal in $|V|$, $|E|$, and k) number of variables and constraints. Next we consider an alternative ILP formulation.

13.1.2 ILP formulation with polynomial number of variables and constraints

Rather than looking at all the paths, we look at an edge granularity and see if a commodity is routed along any edge.
We have variables $x_{e,i}$, $\forall e \in E, i = 1, \ldots, k$
$x_{e,i} = 1$, if $e \in P_i$, and 0 otherwise. [P_i is the chosen (s_i, t_i) path of commodity i]
The ILP formulation is

Minimize t s.t

$$\sum_{e \in \delta^+(t_i)} x_{e,i} = \sum_{e \in \delta^-(s_i)} x_{e,i} = 1 \forall i \tag{13.1.5}$$

$$\forall i, \forall v \neq s_i, t_i \sum_{e \in \delta^+(v)} x_{e,i} = \sum_{e \in \delta^-(v)} x_{e,i} \tag{13.1.6}$$

$$\sum_i x_{e,i} \leq t \forall e \in E \tag{13.1.7}$$

$$t \geq 0 \tag{13.1.8}$$

$$x_{e,i} \in \{0,1\} \tag{13.1.9}$$

In the above formulation, δ^+ indicates the flow coming into a vertex, and δ^- indicates the flow going out of a vertex. Constraint 13.1.5 makes sure that we have a unit flow being routed out of every s_i, and a unit flow being routed into every t_i. Constraint 13.1.6 is like flow conservation at every other vertex. Congestion along an edge is just the number of commodities being routed along that edge, and constraint 13.1.7 ensures that $t \geq$ congestion along any edge. The objective function is to minimze t.

The problem can be relaxed to an LP problem by letting $x_{e,i} \in [0,1]$. It can be noted that this formulation (let's call it edge formulation) is equivalent to the formulation in Section 13.1.1 (let's call it path formulation). From the LP solution to this edge formulation, one can obtain the LP solution to the path formulation. (This is left as an exercise to the reader). Now, we have a fractional solution $\{x_P\}$ for each variable x_P. We also know that the solution to the LP $t \leq C^*$, where C^* is the optimal solution to the ILP. Now, we need to round the LP solution to an ILP solution such that we achieve a reasonable approximation to t, the LP solution.

13.1.3 Deterministic rounding

First, we can note that a deterministic rounding doesn't help us get a good approximation. For example, let us look at a deterministic strategy. A reasonable strategy would be to look at all $P \in P_i$, and round the one with highest value to 1, and others to 0. Say if P_i has n paths, it might be the case that in the LP solution, all the n paths have a weight of $\frac{1}{n}$. So while rounding, the value of a path can increase by a factor of n. Now the optimal congestion t on an edge is caused by some paths, and if the value of each of these paths increase by a factor of n during the rounding, we can only get an n factor approximation. Since n can be an exponential number, this is not a very useful approximation factor. Likewise it can be seen that other deterministic rounding strategies do not help in getting a good approximation factor.

13.1.4 Randomized rounding

For every commodity i, consider the probability distribution on P_i given by $\{x_P\}_{P\in P_i}$, and pick one of the P_i's according to this probability distribution. For example, say we have three paths with values 0.5, 0.3, and 0.2. Get a random number between 0 and 1. If the number is between 0 and 0.5, pick the first path. If the number is between 0.5 and 0.8, pick the second path, and if the number is between 0.8 and 1, pick the third path.

For any edge e, $\mathbf{Pr}[\text{commodity } i \text{ is routed along } e] = x_{e,i}$.
Let X_e be a random variable , and let X_e = number of commodities i with $P_i \ni e$. In other words, X_e is a random variable which indicates the level of congestion along an edge e. To get the $\mathbf{E}[X_e]$, we can use indicator random varibales.
Let $X_{e,i} = 1$, if $P_i \ni e$, and 0 otherwise.
$X_e = \sum_i X_{e,i}$
$\Rightarrow \mathbf{E}[X_e] = \sum_i \mathbf{E}[X_{e,i}]$ [By linearity of expectation]
$= \sum_i x_{e,i} \le t$
This means that the expected value of congestion along any edge $\le t$, the solution to the LP problem. Now if we can show that for any edge, $\mathbf{Pr}[X_e \ge \lambda t] \le \frac{1}{n^3}$, the by the union bound, $\mathbf{Pr}[\exists$ edge e s.t $X_e \ge \lambda t] \le \frac{|E|}{n^3} \le \frac{1}{n}$, and we can get a λ approximation with a high probability.

$$X_e = \sum_i X_{e,i}$$

$$\mathbf{E}[X_e] = \mu \le t$$

$$\mathbf{Pr}[X_e > \lambda t] \le \mathbf{Pr}[X_e > \lambda\mu]$$

Using Chernoff's bounds:

$$\mathbf{Pr}[X_e > \lambda t] \le \left(\frac{e^{\lambda-1}}{\lambda^\lambda}\right)^\mu$$

$$\approx \lambda^{-\lambda\mu}$$

$$\approx \lambda^{-\lambda}$$

(Taking $\mu \approx 1$, atleast edge picked once in expectation.) Now,

$$\lambda^\lambda = n^3 \Rightarrow \lambda = O\left(\frac{logn}{loglogn}\right)$$

. We have with high probability, λ approximation.

13.2 LP Duality

The motivation behind using an LP dual is they provide lower bounds on LP solutions. For instance, consider the following LP problem.
Minimize $7x + 3y$, such that

$$x + y \geq 2$$

$$3x + y \geq 4$$

$$x, y \geq 0$$

Say, by inspection, we get a solution $x = 1, y = 1$, with an objective function value of 10. We can show that this is the optimal solution in the following way.
Proof: If we multiply the constraints with some values and add them such that the coeffiecients of x and y are < 7 and 3, respectively, we can get a lower bound on the solution. For instance, if we add the two constraints, we get $4x + 2y \geq 6$, and since $x, y \geq 0$, we have $7x + 3y > 4x + 2y \geq 6$. So, 6 is a lower bound to the optimal solution. Likewise, if we multiply the first constraint with 1, and the second constraint with 2, we have

$$7x + 3y = (x + y) + 2(3x + y) \geq 2 + 2(4) = 10$$

Hence, 10 is a lower bound on the solution, and so $(x = 1, y = 1)$ with an objective function value of 10 is an optimal solution. ■
Similar to the above example, LP duality is a general way of obtaining lower bounds on the LP solution. The idea is we multiply each constraint with a multiplier and add them, such that the sum of coefficients of any variable is $\leq$ the coefficient of the variable in the objective function. This gives us a lower bound on the LP solution. We want to choose the multipliers such that the lower bound is maximized (giving us the tightest possible lower bound).

13.2.1 Converting a primal problem to the dual form

To convert the primimal to the dual, we do the following.
1. For each constraint in the primal, we have a corresponding variable in the dual. (this variable is like the multiplier).
2. For each variable in the primal, we have a corresponding constraint in the dual. These constraints say that when we multiply the primal constraints with the dual variables and add them, the sum of coefficients of any primal variable should be less than or equal to the coefficient of the variable in the primal objective function.
3. The dual objective function is to maximize the sum of products of right hand sides of primal constraints and the corresponding dual variables. (This is maximizing the lower bound on the primal LP solutions).
The following is a more concrete example showing the primal to dual conversion.
A linear programming problem is of the form:

Minimize $\sum_i c_i x_i$, such that,

$$\sum_i A_{ij} x_i \geq b_j \forall j$$

$$x_i \geq 0 \forall i$$

We call this the primal LP. This LP can be expressed in the matrix form as:

Minimize $c^T x$, such that,

$$Ax \geq b$$

$$x \geq 0$$

The corresponding dual problem is: Maximize $\sum_j b_j y_j$, such that,

$$\sum_j A_{ij} y_j \leq c_i \forall i$$

$$y_j \geq 0 \forall j$$

Expressed in matrix form, the dual problem is, Maximize $b^T y$, such that

$$A^T y \leq c$$

$$y \geq 0$$

Note that the dual of a dual LP is the original primal LP.

Theorem 13.2.1 Weak LP duality theorem *If x is any primal feasible solution and y is any dual feasible solution, then $Val_P(x) \geq Val_D(y)$*

Proof:

$$\sum_i c_i x_i \geq \sum_i \left(\sum_j A_{ij} y_j \right) x_i$$

$$= \sum_j \left(\sum_i A_{ij} x_i \right) y_j$$

$$\geq \sum_j b_j y_j$$

■

So, the weak duality thoerem says that any dual feasible solution is a lower bound to the primal optimal solution. This is a particularly nice result in the context of approximation algorithms. In the previous lectures, we were solving the primal LP exactly and using the LP solution as a lower bound to the optimal ILP solution. By using the weak duality theorem, instead of solving the LP exactly to obtain a lower bound on the optimal value of a problem, we can (more easily) use any dual feasible solution to obtain a lower bound.

Theorem 13.2.2 Strong LP duality theorem *If the primal has an optimal solution x^* and the dual has an optimal solution y^*, then $c^T x^* = b^T y^*$, i.e, the primal and the dual have the same optimal objective function value.*

In general, if the primal is infeasible (there is no feasible point which satisfies all the constraints), the dual is unbounded (the optimal objective function value is unbounded). Similarly, if the dual is infeasible, the primal is unbounded. However, if both the primal and dual are feasible (have atleast one feasible point), the strong LP duality theorem says that the optimal solutions to the primal and the dual have the exact same objective function value.

CS787: Advanced Algorithms

Scribe: Amanda Burton, Leah Kluegel **Lecturer:** Shuchi Chawla
Topic: Primal-Dual Algorithms **Date:** 10-17-07

14.1 Last Time

We finished our discussion of randomized rounding and began talking about LP Duality.

14.2 Constructing a Dual

Suppose we have the following primal LP.

min $\sum_i c_i x_i$ *s.t.*

$\sum_i A_{ij} x_i \geq b_j \quad \forall j = 1, ..., m$

$x_i \geq 0$

In this LP we are trying to minimize the cost function subject to some constraints. In considering this canonical LP for a minimization problem, lets look at the following related LP.

max $\sum_j b_j y_j$ *s.t.*

$\sum_j A_{ij} y_j \leq c_i \quad \forall i = 1, ..., n$

$y_j \geq 0$

Here we have a variable y_j for every constraint in the primal LP. The objective function is a linear combination of the b_j multiplied by the y_j. To get the constraints of the new LP, if we multiply each of the constraints of the primal LP by the multiplier y_j, then the coefficients of every x_i must sum up to no more than c_i. In this way we can construct a dual LP from a primal LP.

14.3 LP Duality Theorems

Duality gives us two important theorems to use in solving LPs.

Theorem 14.3.1 ***(Weak LP Duality Theorem)*** *If x is any feasible solution to the primal and y is any feasible solution to the dual, then $Val_P(x) \geq Val_D(y)$.*

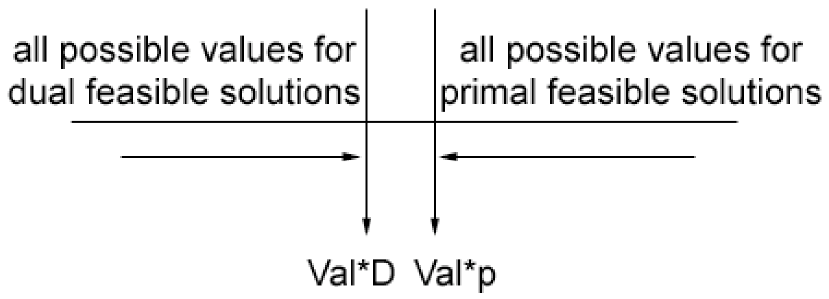
all possible values for dual feasible solutions
all possible values for primal feasible solutions
Val*D
Val*p

Figure 14.3.1: Primal vs. Dual in the Weak Duality Theorem.

On the number line we see that all possible values for dual feasible solutions lie to the left of all possible values for primal feasible solutions.

Last time we saw an example in which the optimal value for the dual was exactly equal to some value for the primal. This introduces the question: was it a coincidence that this was the case? The following theorem claims that no, it was not a coincidence. In fact, this is always the case.

Theorem 14.3.2 ***(Strong LP Duality Theorem)*** *When P and D have non-empty feasible regions, then their optimal values are equal, i.e.* $Val_P^* = Val_D^*$.

On the number line we see that the maximum value for dual feasible solutions is equivalent to the minimum value for primal feasible solutions.

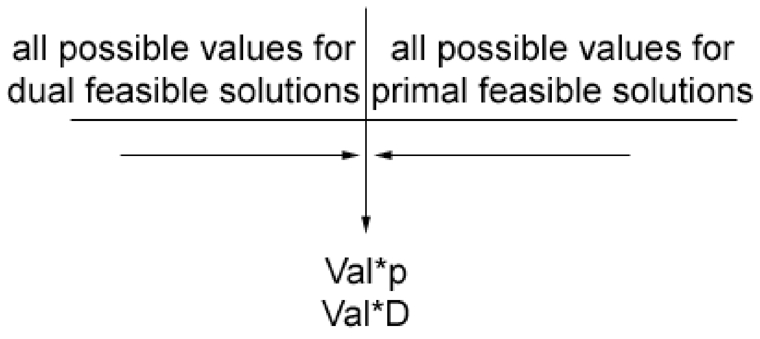
all possible values for dual feasible solutions
all possible values for primal feasible solutions
Val*p
Val*D

Figure 14.3.2: Primal vs. Dual in the Strong Duality Theorem.

It could happen that there is some LP with no solution that satisfies all of the constraints. In this case the feasible region would be empty. In this case the LP's dual will be unbounded in the sense that you can achieve any possible value in the dual.

The Strong LP Duality Theorem has a fairly simple geometric proof that will not be shown here due to time constraints. Recall from last time the proof of the Weak LP Duality Theorem.

Proof: (Theorem 14.3.1)

Start with some feasible solution to the dual LP, say y. Let y's objective function value be $Val_D(y)$. Let x be a feasible solution to the primal LP with objective function value $Val_P(x)$. Since y is a feasible solution for the dual and x is a feasible solution to the primal we have

$$\begin{aligned} Val_D(y) &= \textstyle\sum_j b_j y_j \\ &\le \textstyle\sum_j (\sum_i A_{ij} x_i) y_j \\ &= \textstyle\sum_i \sum_j A_{ij} x_i y_j \\ &= \textstyle\sum_i (\sum_j A_{ij} y_j) x_i \\ &\le \textstyle\sum_i c_i x_i \\ &= Val_P(x) \end{aligned}$$

■

So just rewriting the equations shows us that the value of the dual is no more than the value of the primal. What happens if these values are exactly equal to one another? This happens when x and y are optimal solutions for the primal and the dual respectively. What can we deduce from this?

First, both of the inequalities in the proof of 14.3.1 must both be equalities. If these are equal we have

$$\sum_j b_j y_j = \sum_j (\sum_i A_{ij} x_i) y_j \tag{14.3.1}$$

and

$$\sum_i (\sum_j A_{ij} y_j) x_i = \sum_i c_i x_i \tag{14.3.2}$$

Then each term on the left hand side of Equation 14.3.1 must equal the corresponding term on the right hand side of Equation 14.3.1 and each term on the left hand side of Equation 14.3.2 must equal the corresponding term on the right hand side of Equation 14.3.2. This is the case if, in Equation 14.3.1, $b_j = \sum_i A_{ij} x_i$ and, in Equation 14.3.2, $c_i = \sum_j A_{ij} y_j$. What if $b_j \neq \sum_i A_{ij} x_i$ or $c_i \neq \sum_j A_{ij} y_j$, can we still have the equalities in Equation 14.3.1 and Equation 14.3.2? Equation 14.3.1 can if $y_j = 0$ and Equation 14.3.2 can if $x_i = 0$.

If x and y are primal feasible and dual feasible, respectively, such that $Val_D(y) = Val_P(x)$, then they must satisfy the following:

1. $\forall j$: either $y_j = 0$ or $b_j = \sum_i A_{ij} x_i$.
2. $\forall i$: either $x_i = 0$ or $c_i = \sum_j A_{ij} y_j$.

If $b_j = \sum_i A_{ij} x_i$, we say that the j^{th} constraint in the primal is tight. Condition 1 is called dual complementary slackness (DCS). Similarly, if $c_i = \sum_j A_{ij} y_j$, we say the the i^{th} constraint in the dual is tight. Condition 2 is called primal complementary slackness (PCS).

For a nicer view of the correspondence between the primal and the dual consider the following way of viewing this property.

Primal LP	**Dual LP**
min $\sum_i c_i x_i$ *s.t.*	max $\sum_j b_j y_j$ *s.t.*
$\sum_i A_{i1} x_i \geq b_1$	$y_1 \geq 0$
$\sum_i A_{i2} x_i \geq b_2$	$y_2 \geq 0$
$\vdots$	$\vdots$
$\sum_i A_{im} x_i \geq b_m$	$y_m \geq 0$
$x_1 \geq 0$	$\sum_j A_{1j} y_j \geq c_1$
$x_2 \geq 0$	$\sum_j A_{2j} y_j \geq c_2$
$\vdots$	$\vdots$
$x_n \geq 0$	$\sum_j A_{nj} y_j \geq c_n$

Here we have associated regular constraints in the primal to non-negativity constraints in the dual and non-negativity constraints in the primal to regular constraints in the dual along the rows of this table. Notice that the primal has m regular constraints and n non-negativity constraints on it and the dual has a variable for each of the m regular constraints and a regular constraint for each of the n non-negativity constraints. Therefore, there is a one-to-one correspondence between constraints in the primal and constraints in the dual. If x and y are optimal solutions for the primal and the dual, then for each row in this table, either the left side or the right side of the table must be tight, i.e. an equality rather than an inequality.

From the Strong LP Duality Theorem we have

Corollary 14.3.3 (x^*, y^*) *are primal and dual optimal solutions respectively if and only if they satisfy DCS and PCS.*

14.4 Simple Example

Consider the following minimization LP.

$$\begin{aligned}
\min\ & x_1 + 3x_2 + 4x_3 + x_5 \ \ s.t. \\
& 5x_1 + 2x_4 \geq 1 \\
& 4x_2 + 3x_3 + x_4 + x_5 \geq 2 \\
& x_1 + x_3 + x_5 \geq 7 \\
& x_1 \geq 0 \\
& \vdots \\
& x_5 \geq 0
\end{aligned}$$

We want to compute the dual of this LP. We want to introduce a variable for each regular constraint in the primal, so we will have variables y_1, y_2, and y_3 in the dual. We want to introduce a constraint for each variable in the primal, so we will have 5 constraints in the dual. Each constraint in the dual is written as a function of the variables y_1, y_2, and y_3 in such a way that the coefficients of x_i sum up to no more than the coefficient of x_i in the objective function of the primal. Finally, we determine the objective function of the dual by letting the right hand sides of the constraints in the primal be the coefficients of the y_j. This gives us the following dual

$$\begin{aligned}
\max\ & y_1 + 2y_2 + 7y_3 \ \ s.t. \\
& 5y_1 + y_3 \leq 1 \\
& 4y_2 \leq 3 \\
& 3y_2 + y_3 \leq 4 \\
& 2y_1 + y_2 \leq 0 \\
& y_2 + y_3 \leq 1 \\
& y_1 \geq 0 \\
& y_2 \geq 0 \\
& y_3 \geq 0
\end{aligned}$$

In matrix form this transposes the coefficient matrix

x_1	x_2	x_3	x_4	x_5
5	0	0	2	0
0	4	3	1	1
1	0	1	0	1

into a coefficient matrix by transposing the first column of coefficients of the primal into the first row of coefficients of the dual, the second column into the second row, and so on.

y_1	y_2	y_3
5	0	1
0	4	0
0	3	1
2	1	0
0	1	1

14.5 The Power of LP-Duality

We would now like to use LP-Duality for designing algorithms. The basic idea behind why we might want to use LP-Duality to design algorithms is to solve a problem more quickly than using Linear Programming on its own. When we need to solve an algorithm with Linear Programming, we relax some Integer Program so that the corresponding LP gives us a good lower bound, solve the LP for some optimal linear solution, and round it off. It is possible to solve Linear Programs in polynomial time for its optimal solution, but the known algorithms still run fairly slowly in practice.

Also, we sometimes want to reduce some very large Integer Program and relax it to some exponential sized Linear Program with a small integrality gap, so that you get a good approximation. In which case, you cannot solve the linear program to optimality in polynomial time. LP-duality lets us take these programs, and instead of finding the optimal point of the LP exactly, we use some feasible solution to the Dual LP to get some lower bound on the optimal solution and often times that suffices. In fact, sometimes this process gives us a way to design a purely combinatorial for approximating a problem.

Instead of writing down the LP and solving the LP, we simultaneously construct feasible solutions to both the Primal and Dual LPs in such a way that these are within a small factor of each other, and then construct an integer solution using those. This is much faster and much simpler than solving the LP to optimality. Sometimes, this can even be used even for LPs of exponential size.

Another nice thing about LP-duality, is that it is such a non-trivial relationship between two seemingly different LPs, that it often exposes very beautiful min-max kinds of theorems about combinatorial objects. One such theorem is the Max-flow Min-cut Theorem, and we will shortly prove that using LP-Duality.

14.6 Max-flow Min-cut

Max-flow Min-cut is one of the most widely stated examples of LP-duality, when in fact there are many other examples of the theorems of that kind that arise from this particular relationship. For this example, though, we will talk about Max-flow Min-cut.

To explore this relationship, we would first like to create some LP that solves a Max-flow problem, find its Dual, and then see how the Dual relates to the Min-cut problem.

For the Max-flow problem, are given a graph $G = (V, E)$ with source s, sink t, and some capacities on edges c_e.

14.6.1 Max-flow LP

First we define the variables of our Max-flow LP:

x_e : amount of flow on edge e

Next we define the constraints on the variables:

$x_e \leq c_e \ \ \forall e$ - the flow on x_e does not exceed the capacity on e

$x_e \geq 0 \ \ \forall e$ - the flow on e is non-negative

$\sum_{e \in \delta^+(v)} x_e = \sum_{e \in \delta^-(v)} x_e \ \ \forall v \neq s, t$ - the flow entering v equals the flow leaving v

Subject to these constraints, our objective function maximizes the amount of flow from s to t, by summing over all the flow entering t or all the flow leaving s:

max $\sum_{e \in \delta^-(s)} x_e$

14.6.2 Alternate Max-flow LP

There exists another equivalent way of writing the Max-flow LP in terms of the flow on paths:

x_p : amount of flow on an s-t path p

$\mathcal{P}$: set if all paths from s-t

This LP has the constraints on the variables:

$x_p \geq 0 \ \ \forall p \in \mathcal{P}$ - the flow on p is non-negative

$\sum_{p \ni e} x_p \leq c_e \ \ \forall e$ - the flow on e is no larger than the edge's capacity

Subject to these constraints, our objective function maximizes the amount of flow on the x_p paths:

max $\sum_{p \in \mathcal{P}} x_p$

This is an equivalent LP, and the Primal we are going to talk about, because this one will be easier to work with when constructing the Dual.

14.6.3 The Dual to the Max-flow LP

First, we must define our variables for the Dual, where each variable in the Dual corresponds to a constraint in the Primal. Since the Primal contains one non-trivial constraint for every edge, the Dual must contain one variable for every edge in the Primal:

y_e : variable for edge $e \in E$

The Dual LP must have one constraint for every variable. Like we did in the Simple Example, for each variable in the Primal, we look at the coefficient that you get when you multiply the Primal constraints by the new variables, y_e, and sum over the number of constraints e. We want to find the coefficient of any fixed variable, and write the constraint coresponding to that variable. For any fixed path p, we analyze the coefficients using the constraints on x_p:

$\sum_e \sum_{p \ni e} x_p y_e \leq \sum_e c_e y_e$

$\sum_p x_p \sum_{e \in p} y_e \leq \sum_e c_e y_e$

From this analysis, we get the constraints on the variable y_e:

$y_e \geq 0 \ \ \forall e$

$\sum_{e \in p} y_e \leq 1 \ \ \forall p$

Subject to these constraints, our objective function is

min $\sum_e y_e c_e$

So far, we have mechanically applied a procedure to a Primal LP and determined its Dual. When you start from a combinatorial problem and obtain its Dual, it is a good question to ask what these variables and constraints actually mean. Do they have a nice combinatorical meaning? Usually, if you start from some combinatorial optimization problem, then its Dual does turn out to have a nice combinatorial meaning.

So what is the meaning of this Dual? It is saying that it wants to assign some value to every edge, such that looking at any s-t path in the graph, the sum total of the values on the edges in that path should be at least one. In order to understand what kind of solution this LP is asking for, think of an integral solution to the same LP. Say that we require $y_e \in \{0, 1\}$. What, then, is a integral feasible solution to this program? If $y_e \in \{0, 1\} \;\; \forall e$, then we are picking some subset of the edges, specifically the subset of edges where $y_e = 1$. What properties should those edges satisfy? It should be that for all s-t paths in the graph, the collection of edges that we pick should have at least one edge from that path, and these edges form a cut in the graph G. If we were to minimize the solution over all the cuts in the graph, this would give us a Min-cut.

Fact 14.6.1 *all integral feasible solutions to the Dual form the set of all s-t cuts.*

Because the Dual is minimizing over some values, any feasible solution to the Dual is going to be greater than or equal to any feasible solution to the Primal, by the Weak Duality Theorem.

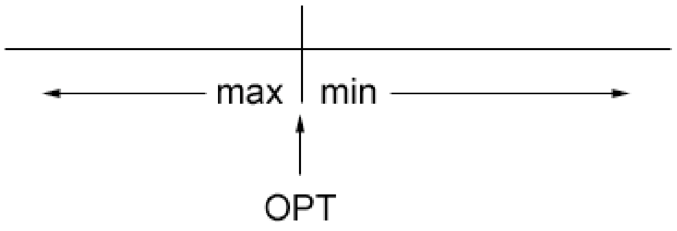

Figure 14.6.3: Ranges of minimization vs. maximazation feasible solutions.

From this we determine the following corollary:

Corollary 14.6.2 *Min-cut $\geq$ Max-flow*

This result is unsurprising, though determining that Min-cut = Max-flow is still non-trivial. Thankfully, the Strong Duality Theorem tells us:

Corollary 14.6.3 *Min-fractional-cut = Max-flow*

The optimal value of the Dual will be exactly equal to the optimal value of the Primal. So far, we do not know if the optimal solution to the Dual is an integral solution or not. If the solution to the Dual is integral, than it will prove the Max-flow Min-cut Theorem, but if it turns out to be a fractional solution, than we cannot determine the best solution to the problem. The Strong Duality Theorem gives us this weaker version of Max-flow Min-cut, that any fractional Min-cut is equal to the Max-flow of the graph. A fractional Min-cut is definted by this LP, where we assign some fractional value to edges, so that the total value of a path is less than or equal to 1. In order to get the Max-flow Min-cut Theorem from this weaker version, we need to show that for every fractional solution there exists integral solution that is no less optimal. We will give a sketch of the proof of this idea.

Theorem 14.6.4 *There exists an integral optimal solution to the Min-cut LP.*

Proof: To prove this, we will start with any fractional feasible solution to this LP, and derive from it an integral solution that is no worse than the fractional solution. Assume some fractional solution to the LP, and think of the values on each edge as lengths. Then the LP is assigning a length to every edge, with the property that any s-t path is going to have a length of at least one. This means that the distance from s to t is at least one.

We are going to lay out the graph in such a way that these y_e's exactly represent the lengths of the edges.

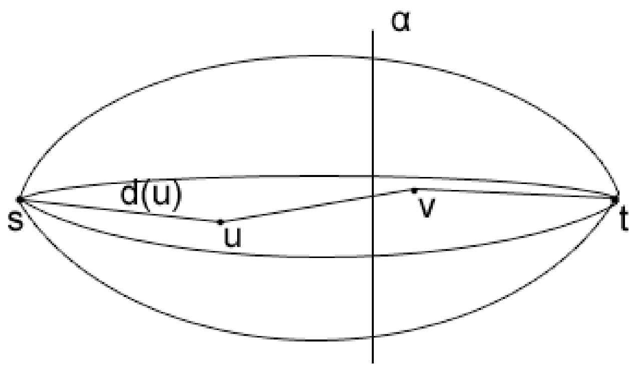

Figure 14.6.4: Distances y_e's from s to t.

Then, we will assign the distance to any point v from the source s as the minimum length of an s-v path. For any point v, we can look at the length of all paths from s to v, and the shortest such path gives me the distance from s to v, $d(v)$, with lengths according to the y_e's. From this we get the distances from s to all the points in the path, and we also know that $d(t) \geq 1$.

Then, we pick some value $\alpha \in [0, 1]$ uniformly at random, and look at all of the vertices that are within distance α of s. This will create a cut, because there are some number of edges crossing over between the set of vertices within distance α those with distances greater than α.

Suppose that we pick α uniformly at random from this range, than if we look at any particular edge in the graph, say (u, v), than the probability that the edge (u, v) is cut is no larger than its length. Why is this? The probability that (u, v) was cut is the probability that $d(u) \leq \alpha$ and $d(v) > \alpha$ (assuming $d(u) \leq d(v)$).

As we draw concentric circles of the distance α from s, then Figure 14.6.5 shows that the difference in the radii of these circles is no more than the distance from u to v. The only way that the edge can be cut, is if we chose an α somewhere between these two concentric circles. So the probability that (u, v) is cut is the difference in the radii, which is no more than the length of (u, v):

$$Pr\left[(u, v)\ is\ cut\right] \leq y_{u,v}$$

Then, the expectation of the size of the cut will be:

$$E\left[size\ of\ cut\right] \leq \sum_{e \in E} c_e Pr\left[e\ is\ cut\right]$$

$$E\left[size\ of\ cut\right] \leq \sum_{e \in E} c_e y_e$$

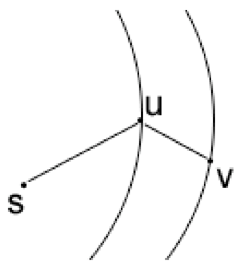

Figure 14.6.5: Distance from u to v vs. possible distances on α.

This expectation represents the size of the cut. If an edge e is in the cut, it contributes c_e to the size of the cut, and 0 otherwise. So we can assign indicator random variables to every edge that takes on a value 1 if it is in the cut, and 0 if it is not. So then the size of the cut is the weighted sum of the indicator variables. Then its expectation is the sum of its expected values of its variables, so it will turn out to be just the cost of each edge times the probability that the edge is in the cut, which is summed over all the edges.

As $\sum_{e \in E} c_e y_e$ is the value of the fractional solution, the expected value of the integer cut is no greater than that of the fractional solution. If we are picking a cut with some probability and the expected value of that cut is small, then there has to be at least one cut in the graph which has small value. So, there exists one integral cut with value at most the value of y.

∎

14.7 Next Time

We will see more applications of LP-duality next time, as well as an algorithm based on LP-duality.

CS787: Advanced Algorithms

Scribe: Mayank Maheshwari, Dan Rosendorf **Lecturer:** Shuchi Chawla
Topic: Primal-Dual Algorithms **Date:** October 19 2007

15.1 Introduction

In the last lecture, we talked about Weak LP-Duality and Strong LP-Duality Theorem and gave the concept of Dual Complementary Slackness (DCS) and Primal Complementary Slackness (PCS). We also analyzed the problem of finding maximum flow in a graph by formulating its LP.

In this lecture, we will develop a 2-approximation for Vertex Cover using LP and introduce the Basic Primal-Dual Algorithm. As an example, we analyze the primal-dual algorithm for vertex cover and later on in the lecture, give a brief glimpse into a 2-player zero-sum game and show how the pay-offs to players can be maximized using LP-Duality.

15.2 Vertex Cover

We will develop a 2-approximation for the problem of weighted vertex cover. So for this problem:

Given: A graph $G(V, E)$ with weight on vertex v as w_v.

Goal: To find a subset $V' \subseteq V$ such that each edge $e \in E$ has an end point in V' and $\sum_{v \in V'} w_v$ is minimized.

The Linear Program relaxation for the vertex cover problem can be formulated as:

The variables for this LP will be x_v for each vertex v. So our objective function is

$$min \sum_v x_v w_v$$

subject to the constraints that

$$x_u + x_v \geq 1 \qquad \forall (u, v) \in E$$

$$x_v \geq 0 \qquad \forall v \in V$$

The Dual for this LP can be written with variables for each edge $e \in E$ as maximizing its objective function:

$$max \sum_{e \in E} y_e$$

subject to the constraints

$$\sum_{v:(u,v) \in E} y_{uv} \leq w_u \qquad \forall u \in V$$

$$y_e \geq 0 \qquad \forall e \in E$$

Now to make things simpler, let us see how the unweighted case for vertex cover can be formulated in the form of a Linear Program. We have the objective function as:

$$max \sum_{e \in E} y_e$$

such that

$$\sum_{e \text{ incident on } u} y_e \leq 1 \qquad \forall u \in V$$

$$y_e \geq 0 \qquad \forall e \in E$$

These constraints in the linear program correspond to finding a matching in the graph G and so the objective function becomes finding a maximum matching in the graph. Hence, this is called the *Matching LP*.

Now remember from the previous lecture where we defined Dual and Primal Complementary Slackness. These conditions follow from the Strong Duality Theorem.

Corollary 15.2.1 (Strong Duality) *x and y are optimal for Primal and Dual LP respectively iff they satisfy:*

1. *Primal Complementary Slackness (PCS) i.e. $\forall i$, either $x_i = 0$ or $\sum_j A_{ij} y_j = c_i$.*
2. *Dual Complementary Slackness (DCS) i.e. $\forall j$, either $y_j = 0$ or $\sum_i A_{ij} x_i = b_j$.*

15.3 Basic Primal-Dual Algorithm

1. Start with $x = 0$ (variables of primal LP) and $y = 0$ (variables of dual LP) . The conditions that:
   - y is feasible for Dual LP.
   - Primal Complementary Slackness is satisfied.

   are invariants and hence, hold for the algorithm. But the condition that:

   - Dual Complementary Slackness is satisfied.

   might not hold at the beginning of algorithm. x does not satisfy the primal LP as yet.
2. *Raise* some of the y_j's, either simultaneously or one-by-one.
3. Whenever a dual constraint becomes tight, *freeze* values of corresponding y's and raise value of corresponding x.
4. Repeat from *Step 2* until all the constraints become tight.

Now let us consider the primal-dual algorithm for our earlier example of vertex cover.

15.3.1 Primal-Dual Algorithm for Vertex Cover

1. Start with $x = 0$ and $y = 0$.
2. Pick any edge e for which y_e is not frozen yet.
3. Raise the value of y_e until some vertex constraint v goes tight.
4. Freeze all y_e's for edges incident on v. Raise x_v to 1.
5. Repeat until all y_e's are frozen.

Let us see an example of how this algorithm works on an instance of the vertex cover problem. We consider the following graph:

Example: Given below is a graph with weights assigned to vertices as shown in the figure and we start with assigning $y_e = 0$ for all edges $e \in E$.

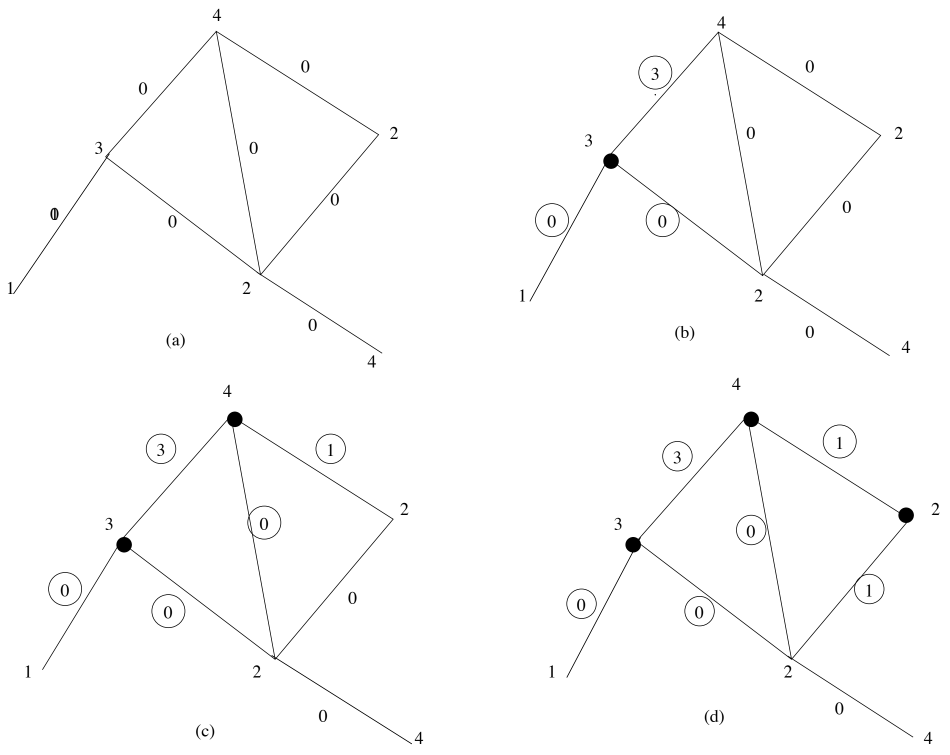

Fig 1: The primal-dual algorithm for vertex cover.

So the algorithm proceeds as shown in the figure above. In steps (a)-(d), an edge is picked for which y_e is not frozen and the value of y_e is raised until the corresponding vertex constraint goes tight. All the edges incident on that vertex are then frozen and value of x_v is raised to 1.

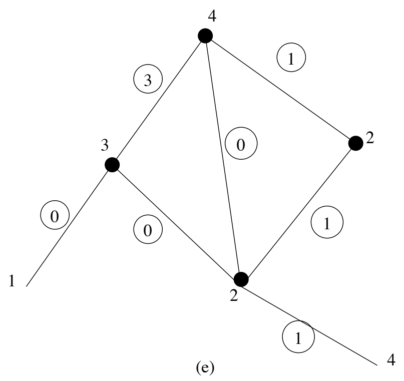

Fig 1: The vertex cover for the example graph.

When all the y_e's get frozen, the algorithm terminates. So the *Value of Primal=11* and the *Value of Dual=6*.

Hence,

$$Val_p(x) \leq 2Val_d(y)$$

and so we get the 2-approximation for our instance of vertex cover.

Lemma 15.3.1 *When the algorithm ends, x is a feasible solution for the primal and y is a feasible solution for the dual.*

Proof: That y is a feasible solution is obvious since at every step we make sure that y is a feasible solution, so it is feasible at the last when the algorithm ends. To see that x is a feasible solution we proceed by contradiction. Let us suppose it is not a feasible solution. Then there is a constraint $x_u + x_v \geq 1$ which is violated. That means both x_v and x_u are zero and thus it must be possible to raise the value of the edge between them since neither of them has a tight bound. This is a contradiction with the fact that all y_e's are frozen. ■

Lemma 15.3.2 *x and y satisfy PCS.*

Proof: This is obvious since at every step we make sure that x and y satisfy PCS. ■

Definition 15.3.3 *Let x and y be feasible solutions to the primal and dual respectively. Then we say that x and y satisfy α-approximate DCS if $\forall j, (y_j \neq 0) \rightarrow (\sum_i A_{ij}x_i \leq \alpha b_j)$.*

Lemma 15.3.4 *x and y satisfy 2-approximate DCS.*

Proof: This follows from the fact that x_v is either 0 or 1 for any v so $x_u + x_v \leq 2$ for any $e = (u, v)$. ■

The next lemma shows us why we would want an α-approximation for DCS.

Lemma 15.3.5 *Suppose x and y satisfy PCS are feasible for Primal and Dual respectively and α-approximate DCS then $Val_P(x) \leq \alpha Val_D(y)$.*

Proof: To prove this we only need to write out the sums. We have $Val_P(x) = \sum_i c_i x_i$ now since we know that x,y satisfy PCS we have that $\sum_i c_i x_i = \sum_i(\sum_j A_{ij} y_j) x_i$ by reordering the summation we get $\sum_j(\sum_i A_{ij}x_i)y_j \leq \sum_j \alpha b_j y_j = \alpha \sum_j b_j y_j = \alpha Val_D(y)$ where the $\leq$ follows from α-approximate DCS. ■

The last two lemmas then directly yield the desired result which is that our algorithm is a 2-approximation for Vertex cover.

We should also note that we never used anywhere in our analysis what order we choose our edges in or how exactly we raise the values of y_e's. A different approach might be to raise the values of all our y_e's simultaneously until some condition becomes tight. Using this approach with the graph we had earlier would give a different result. We would start by raising the values of all edges to $\frac{1}{2}$ at which point the vertex at the bottom of the diamond would become full and it's tightness condition would be met. We freeze all the edges adjecent to that vertex add it to our vertex cover and continue raising the values of the other y_e's. The next condition to be met will be the left bottom most node when we raise the value of the incoming edge to 1. After that we continue rasing the values of the remaining unfrozen edges until $1\frac{1}{2}$ when both the left and right edge of the diamond become full and we freeze all the remaining edges. The run of this algorithm with the circled numbers denoting frozen edge capacities and the emptied nodes denoting nodes in the graph cover can be seen in the following figures.

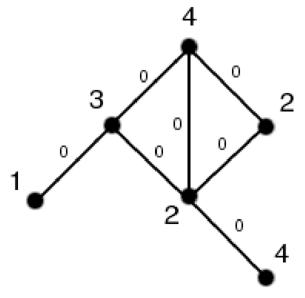

Figure 15.3.1: Step 0

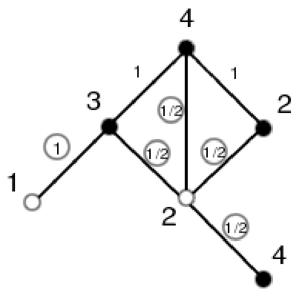

Figure 15.3.3: Step 2

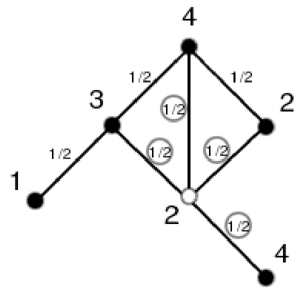

Figure 15.3.2: Step 1

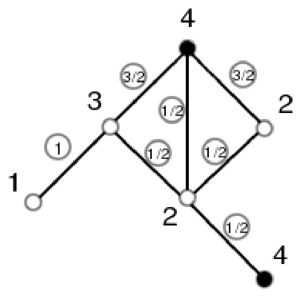

Figure 15.3.4: Step 3

Note that while this approach gives a better result in this case than the first one we used, it is not that case in general. In fact there are no known polynomial time algorithms that give a result of

$(2-\epsilon)$-approximation for any ϵ small constant.

15.4 Minimax principle

Next we will look at an application of LP-duality in the theory of games. In particular we will prove the Minimax theorem using LP-duality. First though we will need to explain some terms. By a 2-player zero sum game, we mean a protocol in which 2 players choose strategies in turn and given two strategies x and y, we have a valuation function $f(x,y)$ which tells us what the payoff for the first player is. Since it is a zero sum game, the payoff for the second player is exactly $-f(x,y)$. We can view such a game as a matrix of payoffs for one of the players.

As an example take the game of Rock-paper-scissors, where the payoff is one for the winning party or 0 if there is a tie. The matrix of winnings for player one will then be the following:

$$\begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$$

Where A_{ij} corresponds to the payoff for player one if player one picks the i-th element and player two the j-th element of the sequence (Rock, Paper, Scissors). We will henceforth refer to player number two as the column player and player number one as the row player. If the row player goes first, he obviously wants to minimize the possible gain of the column player. So the payoff to the row player in that case will be $min_j(max_i(A_{ij}))$. On the other hand if the column player goes first, we get the payoff being $max_i(min_j(A_{ij})$. It is clearly the case that $max_i(min_j(A_{ij})) \leq min_j(max_i(A_{ij}))$.

The Minimax theorem states that if we allow the players to choose probability distributions instead of a given column or row then equality holds or slightly more formally:

Theorem 15.4.1 *If x and y are probability vectors then $max_y(min_x(y^TAx) = min_x(max_y(y^TAx))$.*

Proof: We will only give a proof sketch. Notice that once we have chosen a strategy, if our opponent wants to minimize his loss, he will always just pick the one row which gives the best result, not a distribution. For $max_y(min_x(y^TAx))$, we wish to find a distribution over the rows such that whatever column gets picked, we get at least a payoff of t such that t is maximal. This can be formulated in terms of an LP-problem as maximizing t given the following set of constraints:

$$\forall j \quad \sum_i y_i A_{ij} \geq t$$

and

$$\sum_i y_i = 1$$

, which can be changed into a more standard form by writing the equations as

$$\forall j \quad t - \sum_i y_i A_{ij} \leq 0$$

and relaxing the second condition as

$$\sum_j y_j \leq 1.$$

On the other hand $min_x(max_y(y^T Ax))$ can be thought of as trying to minimize the loss and can thus be rewritten in terms of an LP-problem in a similar fashion as minimizing s given that

$$\sum_j x_j \leq 1$$

and

$$\forall i \quad s - \sum_j A_{ij} x_j \leq 0$$

.

In other words, we are trying to minimize the loss, no matter which column our opponent chooses. We leave it as an exercise to prove that the second problem is a dual of the first problem and thus by the Strong duality theorem, the proof is concluded. ■