College of Computer & Information Science — Fall 2007
Northeastern University — Handout 1
CS G399: Algorithmic Power Tools I — 7 September 2007

Lecture Outline:

- Course outline
- Approximation algorithms
- Set cover
- Greedy algorithm
- Introduction to LP and LP duality
- Set cover analysis using LP duality

1 Approximation algorithms

A major focus of this course is on the study of hard optimization problems. Many of these problems fall in the class of problems referred to as NP-complete. It is widely believed that no NP-complete problem can be solved optimally in polynomial time. We can relax our requirements in the following three ways:

- **Super-polynomial time algorithms:** Instead of requiring poly-time algorithms, we may demand a slightly super-polynomial time algorithm. Indeed, for certain problems such as Knapsack, we can obtain efficient slightly super-polynomial time algorithms that solve the probem optimally. It turns out, however, that the fastest known algorithms for almost all of the interesting NP-complete problems is exponential time.
- **Focus on a subset of the instances:** We can also relax our objective by focusing on a subset of the instances, rather than demanding that we solve every instance optimally. The trouble with this approach is to define the particular subset of instances we are interested in. While it may be possible to define the notion of a "random instance" in some cases and analyze the expected running-time of algorithms by probabilistic means, this is difficult to justify for most problems.
- **Approximation algorithms:** A third relaxation, which is the one we will discuss during part of this course, is to allow for approximate solutions. That is, rather than solving optimally for every instance, we will demand *polynomial-time* algorithms that solve *near-optimally* for *every instance*.

What does "near-optimal" mean? One notion of approximation is that of an *absolute performance guarantee*, in which the value of the solution returned by the approximation algorithm differs from the optimal value by an absolute constant. Let Π be an optimization problem and let I be an instance of Π. Given an algorithm $\mathcal{A}$ for Π, let $\mathcal{A}(I)$ denote both the solution as well as the value of the solution returned by $\mathcal{A}$ on instance I. Also, let $\mathrm{OPT}(I)$ denote the optimal value for instance I.

Definition 1. An approximation algorithm $\mathcal{A}$ for problem Π has an **absolute performance guarantee** of k if for the following condition holds for all instances I of Π:

$$|\mathcal{A}(I) - \mathrm{OPT}(I)| \leq k.$$

$\square$

While this notion is useful for certain problems such as the minimum-degree spanning tree problem, it turns out to be inappropriate for most NP-complete problems. The notion of approximation that is most widely used is that of a *relative performance guarantee*, which we now define.

Definition 2. An approximation algorithm $\mathcal{A}$ for problem Π has an **approximation ratio** of r if the following condition holds for all instances I of Π:

$$\frac{\mathcal{A}(I)}{\mathrm{OPT}I} = r.$$

2 Set cover

The set cover problem plays a central role in the study of approximation algorithms. It has numerous applications in diverse areas and several other fundamental optimization problems are natural generalizations of set cover. It is also a very useful problem to demonstrate several fundamental techniques developed in the area of approximation algorithms.

Problem 1. Given a universe $\mathcal{U} = \{e_1, \ldots, e_n\}$ of n elements, a collection $\mathcal{S}$ of m subsets of $\mathcal{U}$, and a cost function $c : \mathcal{S} \to Q^+$, the set cover problem seeks a minimum-cost subset of $\mathcal{S}$ that covers all elements of $\mathcal{U}$.

The set cover problem is NP-complete, so we seek good approximation algorithms. The first algorithm that comes to mind for the set cover problem is the following greedy algorithm.

Repeat the following step until all elements of $\mathcal{U}$ are covered: select a set $S \in \mathcal{S}$ that has the least ratio of cost to the number of uncovered elements in S.

It is relatively easy to see that the greedy algorithm is not optimal. How far from optimal can it be?

First Analysis.

Divide the algorithm's progress into phases, starting from phase 0. In the ith phase, the number of uncovered elements is in $[n/2^i, n/2^{i+1})$. Therefore, the cost-effectiveness of sets selected in the ith phase is always at most $OPT/(n/2^{i+1})$. Since the number of elements covered in the ith phase is at most $n/2^i$, the cost of the sets added in the ith phase is at most $2OPT$. The total number of phases is at most $\lg n$. Therefore, the total cost of the greedy solution is at most $2 \lg nOPT$.

Second Analysis.

Define the price of an element to be the average cost at which it is covered. Let $e_1, e_2, \ldots, e_n$ denote the elements in the order they were covered.

Lemma 1. *The price of element e_i is at most $OPT/(n-i+1)$.*

Proof. At the instant element e_i is covered, there are at least $n-i+1$ uncovered elements. Therefore, there exists a set in the optimal solution that has a cost-effectiveness of at most $OPT/(n-i+1)$. Since e_i is covered in this step, its price is at most $OPT/(n-i+1)$. □

Theorem 1. *The approximation ratio of the greedy algorithm is at most H_n.*

Proof. The cost of a selected set is simply the sum of the prices of the new elements it covers. Thus, the cost of the greedy solution is simply the sum of the prices of all the elements, which, according to Lemma 1, is upper bounded by

$$OPT \sum_{i=1}^{n} \frac{1}{n-i+1} = OPT \cdot H_n.$$

□

3 Linear programming

Linear programming is the problem of optimizing a linear objective function subject to linear (in)equality constraints. It has been extensively studied the second half of the 20th century and has played a major role in the design of algorithms (both exact and approximation) for combinatorial optimization problems. There are several equivalent forms of writing a linear program. In the following, we consider minimization problems.

3.1 General Form

This is the general form of the linear programming problem. The c_i's can be interpreted as costs. In this case, the objective is to minimize the total cost subject to the linear constraints.

$$\text{Minimize} \sum_{i=1}^{n} c_i x_i \qquad \text{(objective function)} \tag{1}$$

$$\begin{array}{rrcll} \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j & \geq & b_i & , i = 1, \ldots, m_1 & \text{(inequality)} \\ & \sum_{j=1}^{n} a_{ij} x_j & = & b_i & , i = m_1 + 1, \ldots, m_1 + m_2 & \text{(equality)} \\ & x_j & \geq & 0 & , j = 1, \ldots, n_1 & \text{(non} - \text{negativity)} \\ & x_j & \lesseqgtr & 0 & , j = n_1 + 1, \ldots, n & \text{(unconstrained)} \end{array}$$

The general form of the LP can be written more compactly in matrix notation.

$$\text{Minimize } \mathbf{c}^T \mathbf{x} \tag{2}$$

$$\begin{array}{rrcl} \text{subject to} & \mathbf{A}\mathbf{x}_1 & \geq & \mathbf{b} \\ & \mathbf{A}'\mathbf{x}_2 & = & \mathbf{b}' \\ & \mathbf{x}_1 & \geq & 0 \\ & \mathbf{x}_2 & <=> & 0 \end{array}$$

where

c and **x** are n x 1 vectors,

A is an m_1 x n matrix,

A$'$ is an m_2 x n matrix,

b is an m_1 x 1 vector,

b$'$ is an m_2 x 1 vector,

$\mathbf{x}_1$ is an n_1 x 1 vector,

$\mathbf{x}_2$ is an n_2 x 1 vector, and

$\mathbf{x} = \left\{ \begin{array}{c} \mathbf{x}_1 \\ \mathbf{x}_2 \end{array} \right\}$.

3.2 Standard Form

Here the constraints take the form of linear inequality constraints, plus non-negativity constraints on the independent variables.

$$\text{Minimize} \sum_{i=1}^{n} c_i x_i \qquad \text{(objective function)} \tag{3}$$

$$\begin{array}{lrcll} \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j & \geq & b_i & , i = 1, \ldots, m \quad \text{(inequality)} \\ & x_j & \geq & 0 & , j = 1, \ldots, n \quad \text{(non} - \text{negativity)} \end{array}$$

Once more, the standard form of the LP can be written more compactly in matrix notation.

$$\text{Minimize } \mathbf{c}^T \mathbf{x} \tag{4}$$

$$\begin{array}{lrcl} \text{subject to} & \mathbf{A}\mathbf{x} & \geq & \mathbf{b} \\ & \mathbf{x} & \geq & 0 \end{array}$$

where

c and **x** are n x 1 vectors,

A is an m x n matrix, and

b is an m x 1 vector.

3.3 Slack Form

Here the constraints take the form of linear equality constraints, plus non-negativity constraints on the independent variables.

$$\text{Minimize} \sum_{i=1}^{n} c_i x_i \qquad \text{(objective function)} \tag{5}$$

$$\begin{array}{lrcll} \text{subject to} & \sum_{j=1}^{n} a_{ij}x_j & = & b_i & ,i=1,\ldots,m \quad \text{(equality)} \\ & x_j & \geq & 0 & ,j=1,\ldots,n \quad \text{(non}-\text{negativity)} \end{array}$$

The standard form of the LP can also be written more compactly in matrix notation.

$$\text{Minimize } \mathbf{c}^T\mathbf{x} \tag{6}$$

$$\begin{array}{lrcl} \text{subject to} & \mathbf{Ax} & = & \mathbf{b} \\ & \mathbf{x} & \geq & 0 \end{array}$$

where

c and **x** are n x 1 vectors,

A is an m x n matrix, and

b is an m x 1 vector.

Definition 3. If x satisfies $\mathbf{Ax} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$, then x is **feasible**.

3.4 Constraint Conversion

It is possible to convert between equality and inequality constraints by the following method.

1. To convert a less than or equal inequality constraint,

$$Ax \leq b$$

to an equality constraint, add the vector of slack variables, s.

$$Ax + s = b,\ s \geq 0$$

2. To convert a greater than or equal inequality constraint,

$$Ax \geq b$$

to an equality constraint, add the vector of surplus variables, t.

$$Ax - t = b,\ t \geq 0$$

3. Finally, to convert an equality constraint,

$$Ax = b$$

to an inequality constraint, add two inequality constraints.

$$Ax \leq b,\ -Ax \leq -b$$

4 The Geometry of LP

Definition 4. A **polytope** is the analogue in n-dimensional space of point, segment, polygon, and polyhedron in spaces of dimensions 0, 1, 2, and 3. A **convex polytope** in n-space is the convex span of a finite set of points that do not all lie in the same hyperplane; thus a convex polytope is a bounded convex subset enclosed by a finite number of hyperplanes.

Definition 5. A set is **convex** if it contains the line segment joining any two of its points; in a *vector space*, a set such that $rx + (1-r)y$ is in the set for $0 < r < 1$, if x and y are in the set.

Definition 6. A point x is a **vertex** of $\mathcal{P}$ if $\nexists y \neq 0$ such that $x + y,\ x - y \in \mathcal{P}$.

Lemma 2. *Let $\mathcal{P}$ denote a convex polytope in R^n and let $x \in \mathcal{P}$. The following three statements are equivalent.*

1. *There does not exist $y \neq 0$ such that $x + y, x - y \in \mathcal{P}$.*
2. *There does not exist $y, z \in \mathcal{P}$ such that $y, z \neq x$ and $x = \alpha y + (1-\alpha)z$ for some $\alpha \in [0,1]$.*
3. *There exist n linearly independent inequalities in $\mathcal{P}$ that are satisfied with equality.*

Proof. The equivalence between the first two statements is easy. Supposing that statement 2 is false, we will show that statement 1 is false. Suppose there exist $y, z \in \mathcal{P}$ such that $y, z \neq x$ and $x = \alpha y + (1-\alpha)z$ for some $\alpha \in (0,1)$. Without loss of generality, we may assume that $\alpha \leq 1/2$. We now show that both $x - (x - y)$ and $x + (x - y)$ belong to $\mathcal{P}$, showing that statement 1 is false. The first part is immediate. For the second part, we obtain

$$2x - y = 2\alpha y + (2 - 2\alpha)z - y = (2\alpha - 1)y + (2 - 2\alpha)z = \beta y + (1 - \beta)z,$$

where $\beta = 2\alpha - 1$. Since $\alpha \leq 1/2$, $\beta \in [0,1]$. Byt convexity of $\mathcal{P}$, $2x - y$ is in $\mathcal{P}$.

Now we suppose that statement 1 is false. Then, there exists $y \neq 0$ such that $x + y$ and $x - y$ are both in $\mathcal{P}$. Since $x = 1/2(x + y) + 1/2(x - y)$, we obtain that statement 2 is also false.

We now prove that statement 1 is equivalent to statement 3. Suppose there does not exist $y \neq 0$ such that $x+y$ and $x-y$ are both in $\mathcal{P}$. Consider the set of constraints that are tight at x. Suppose A' and b' are the submatrices corresponding to these constraints. Then, we have $A'x = b'$. If the rank of this matrix is n, then we are done since x is a vertex. Otherwise, the rank of A' is less than n. We therefore obtain that there exists $y \neq 0$ such that $A'y = 0$. Clearly, $x + \lambda y$ satisfies $A'x = b'$ for any scalar λ. By setting $\lambda > 0$ small enough, all the remaining inequalities can be satisfied for both $x + \lambda y$ and $x - \lambda y$.

Suppose there exist n linearly independent constraints that are tight. Let A' and b' be the appropriate sub-matrices corresponding to these constraints. Then, we have $A'x = b'$. If there exists $y \neq 0$ such that $x + y, x - y \in \mathcal{P}$, then $A'x + A'y \geq b'$ and $A'x - A'y \geq b'$, which implies that $A'y = 0$. But this can be only true if A' is linearly dependent, arriving at a contradiction. □

Theorem 2. *If the optimal solution to an LP in standard form is bounded (from below), then given any $x \in \mathcal{P}$, $\exists$ a vertex v such that $c^T v \leq c^T x$.*

Corollary 1. *If the LP is bounded, then $\exists$ a vertex that yields the optimal solution.*

Proof: (of theorem) We are given $x \in \mathcal{P}$. If x is a vertex, we are done. Otherwise, $\exists y \neq 0$ such that $x+y \in \mathcal{P}$ and $x-y \in \mathcal{P}$. By feasibility of $x+y$ and $x-y$, $A(x+y) \geq b$, $A(x-y) \geq b$. Since $x+y \geq 0$ and $x-y \geq 0$, we obtain that $y_j = 0$ whenever $x_j = 0$.

Consider the set of constraints that are tight at x. Suppose A' and b' are the submatrices corresponding to these constraints. Then, we have $A'x = b'$. Since x is not a vertex, the rank of A' is less than n. We therefore obtain that there exists $y \neq 0$ such that $A'y = 0$. Clearly, $x + \lambda y$ satisfies $A'x = b'$ for any scalar λ. By setting $\lambda > 0$ small enough, all the remaining inequalities can be satisfied for both $x + \lambda y$ and $x - \lambda y$. For each constraint, we determine that λ value that will make the constraint tight.

Without loss of generality, choose y such that $c^T y \leq 0$. Then $c^T(x+\lambda y) = c^T x + \lambda c^T y \leq c^T x$.

Case 1. there exists a λ value that is greater than 0.
We need to ensure that $x + \lambda y \in \mathcal{P}$ to maintain feasibility. We know that whenever $x_i = 0$, we have $y_i = 0$. Hence we select the largest such λ value. It is clear that $x + \lambda y \in \mathcal{P}$, and $x + \lambda y$ is tight in one more constraint than x.

Case 2. all the lambda values are nonpositive.
In this case, if $c^T y = 0$, then we can negate every component of y and switch to case 1 (note that $y \neq 0$). It remains to consider the subcase $c^T y < 0$. Clearly, $x + \lambda y \in \mathcal{P}$ for all $\lambda > 0$. Hence, $c^T(x + \lambda y) = c^T x + \lambda c^T y$. Then λ can be chosen arbitrarily large, and the objective function is unbounded. Contradiction.

Case 1 can happen at most n times, since otherwise x is a vertex. By induction on the number of linearly independent tight constraints at x, we obtain a vertex v. $\square$

4.1 LP duality

Consider the following LP.

$$\begin{array}{rll} \text{minimize} & 3x_1 + 2x_2 + 8x_3 & \\ \text{s.t.} & x_1 - x_2 + 2x_3 & \geq 5 \\ & x_1 + 2x_2 + 4x_3 & \geq 10 \\ & x_1, x_2, x_3 & \geq 0 \end{array}$$

Let z^* denote the optimum value of the above linear program. One question we can ask is how small can z^* be? If we sum up the first two inequalities, we obtain

$$2x_1 + x_2 + 6x_3 \geq 15.$$

Since x_1, x_2, x_3 are all nonnegative, it follows that z^* is at least 15. We can obtain even a better lower bound if we sum twice the first inequality with the second inequality to obtain

$$3x_1 + 8x_3 \geq 20.$$

What is the best lower bound we can get by taking linear combinations of the constraints? We can formulate this problem by the following LP.

$$\begin{array}{rll} \text{maximize} & 5y_1 + 10y_2 & \\ \text{s.t.} & y_1 + y_2 & \leq 3 \\ & -y_1 + 2y_2 & \leq 2 \\ & 2y_1 + 4y_2 & \leq 8 \\ & y_1, y_2 & \geq 0 \end{array}$$

The above LP is referred to as the dual of the first LP, which is referred to as the primal. In general, the dual of an LP of the form

$$\min c^T x : Ax \geq b, x \geq 0,$$

is the LP

$$\max b^T y : A^T y \leq c, y \geq 0,$$

Theorem 3. *If x and y are feasible solutions for the following primal and dual LPs:*

$$\min c^T x : Ax \geq b, x \geq 0,$$
$$\max b^T y : A^T y \leq c, y \geq 0,$$

then $c^T x \geq b^T y$.

Proof: We have $c^T x \geq (A^T y)^T x = y^T Ax$. On the other hand, $b^T y \leq (Ax)^T y = (y^T (Ax))^T = y^T Ax$. $\square$

5 Set cover via dual fitting

As we have seen above, the dual of an LP provides a useful lower bound on the value of the optimum solution to the primal minimization problem. For many of the combinatorial optimization problems, this provides a very effective technique for analyzing combinatorial algorithms. Here, we analyze the greedy algorithm for set cover using weak duality and a technique referred to as dual fitting.

Informally, the approach is as follows. From the greedy solution, we also obtain a solution for the dual, with at least the same objective value. But this solution may not be dual-feasible. If scaled, the solution becomes dual feasible. The scaling factor yields the approximation ratio.

The LP for set cover is

$$\begin{array}{rll} \text{minimize} & \sum_{S \in \mathcal{S}} c(S) x_S & \\ \text{s.t.} & \sum_{S : e \in S} x_S \geq 1 & \forall e \in \mathcal{U} \\ & x_S \geq 0 & \end{array}$$

The dual of the LP is

$$\begin{array}{rll} \text{maximize} & \sum_{e \in \mathcal{U}} y_e & \\ \text{s.t.} & \sum_{e : e \in S} y_e \leq c(S) & \forall S \in \mathcal{S} \\ & y_e \geq 0 & \end{array}$$

One informal interpretation of the dual is that it is aiming to pack as many elements as possible subject to the constraint that no set is "overstuffed" beyond its cost. The primal LP above is a covering LP while the dual LP is a packing LP.

By weak duality, we know that $\mathrm{OPT} \geq \mathrm{OPT}_p \geq \mathrm{OPT}_d$, which is at least the cost of any feasible dual solution. So, if we are able to construct from the greedy algorithm's solution a dual feasible solution whose cost is less than that of the greedy solution by at most a small factor, then we can establish a good bound on the approximation ratio.

Recall the definition of price of e. Define y_e as follows.

$$y_e = \frac{\text{price of } e}{H_n}.$$

Lemma 3. *The vector y is a feasible solution for the dual LP.*

Proof: Consider any set S with, say, k elements. Label the elements e_1 to e_k in the order they were covered. When e_i is covered, the price of e_i is at most $c(S)/(k-i+1)$. Therefore, we have

$$y_{e_i} \leq \frac{c(S)}{H_n(k-i+1)}$$

Summing y_{e_i} over all i yields at most $c(S)H_k/H_n \leq c(S)$. □

The cost of the set cover picked by the greedy algorithm equals the sum of the prices of all the elements, which is at most H_n times the sum of y_e's. Since y is dual feasible, we obtain that the cost of the greedy algorithm is at most H_n times the optimal.