LP Relaxation and Rounding

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

The basic idea of LP-relaxation and rounding is quite simple. We first formulate an optimization problem as an *integer program* (IP), which is a linear program (LP) with integer variables. Then, we relax the integral constraints to turn the IP to an LP. Next, we solve the LP for an optimal solution, say $\mathbf{x}^*$, from which we construct a feasible solution $\mathbf{x}^A$ to the IP. This construction step is often done by *rounding* the coordinates of $\mathbf{x}^*$ to integers so that $\mathbf{x}^A$ is feasible. Rounding can be done deterministically or probabilistically. In the latter approach is taken, we have the so-called *randomized rounding* method. In many problems, we might need to look at the structure of an optimal solution to the LP, in order to obtain sufficient information to make rounding decision.

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

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

Consequently, any upper bound of the ratio $\frac{\text{cost}(\mathbf{x}^A)}{\text{cost}(\mathbf{x}^*)}$ is an approximation ratio for this algorithm. If this approach is taken "as is," the best approximation we can hope for is $\frac{\text{cost}(IP)}{\text{cost}(LP)}$, whose supremum is referred to as the *integrality gap*. We will have more to talk about this gap later. It is possible, in many cases, to devise algorithms whose approximation ratios are better than the integrality gap by delving deeper into the structure of the problem at hand.

1 Linear programs and linear integer programs

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

VERTEX COVER

> Given a graph $G = (V, E)$, $|V| = n$, $|E| = m$, find a minimum vertex cover, namely a minimum-size subset $C \subseteq V$ such that every edge of G has at least one end in C.

Let us see how we can formulate VERTEX COVER as an integer program. Suppose we are given a graph $G = (V, E)$ with n vertices and n edges. For each $i \in V = \{1, \ldots, n\}$, let $x_i \in \{0, 1\}$ be a

variable which is 1 if i belongs to the vertex cover, and 0 otherwise; then, the problem is equivalent to solving the following (linear) *integer program*:

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

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

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

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

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

WEIGHTED VERTEX COVER

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

An equivalent linear integer program can be written as

$$\begin{array}{ll} \min & w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \\ \text{subject to} & x_i + x_j \geq 1, \quad \forall ij \in E, \\ & x_i \in \{0,1\}, \quad \forall i \in V. \end{array}$$

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

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

SET COVER

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

WEIGHTED SET COVER

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

Similar to VERTEX COVER, we use a 01-variable x_j to indicate the inclusion of S_j in the cover. The corresponding IP is thus

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

TRAVELING SALESMAN (TSP)

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

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

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

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

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

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

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

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

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

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

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

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

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

(That is, if you do it correctly.)

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

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

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

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

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

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

2 Covering Problems

The LP corresponding to the IP (1) of VERTEX COVER is

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

Exercise 6. Show that, given any graph G, any vertex $\mathbf{x}$ of the polyhedron defined by (4) is *half-integral*, i.e. x_i is either $0, 1/2$, or 1.

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

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

You should check that $\mathbf{x}^A$ is definitely feasible for the IP. This technique of constructing a feasible solution for the IP from the LP is called *rounding*. We have just seen the second advantage of doing relaxation. The third is that an optimal value for the LP provides a lower bound for the optimal value of the IP. Following the principle described in the beginning, an upper bound of $\text{cost}(\mathbf{x}^A)/\text{cost}(\mathbf{x}^*)$ is an approximation ratio for this algorithm. It is straightforward that

$$\text{cost}(\mathbf{x}^*) = x_1^* + \cdots + x_n^* \geq \frac{1}{2}x_1^A + \cdots + \frac{1}{2}x_n^A = \frac{1}{2}Cost(x^A).$$

We thus have a 2-approximation algorithm to solve the VERTEX COVER problem. Since it is impossible, unless P $=$ NP, to have an exact polynomial time algorithm to solve VERTEX COVER, an algorithm giving a feasible solution within twice the optimal is nice to have. The exact same technique works for the WEIGHTED VERTEX COVER problem, when the weights are non-negative. Thus, we also have a 2-approximation algorithm for the WEIGHTED VERTEX COVER problem.

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

Exercise 7. Show that the integrality gap for the VERTEX COVER problem is at least $2(1 - 1/n)$, where n is the number of vertices.

Exercise 8. Given a graph G and and proper k-coloring of G (i.e. a coloring using k colors such that two adjacent vertices have different colors). We want to solve the WEIGHTED VERTEX COVER on G. Devise a polynomial time $(2 - 2/k)$-approximation algorithm to solve this problem.

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

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

where a_{ij}, b_i, c_j are all non-negative integers. Since we can remove an inequality if $b_i = 0$, we can assume that $\mathbf{b} > 0$. Moreover, if $c_j = 0$ then we can set $x_j = 1$ and remove column j. Thus, we can also assume that $\mathbf{c} > 0$. Lastly, assume $\sum_j a_{ij} \geq b_i$, for all $i \in [m]$; otherwise, the system is infeasible.

The relaxed LP version for (5) is

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

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

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

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

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

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

The analysis above leads to the following rounding strategy. Let

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

and set

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

then we have an approximation ratio of f.

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

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

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

But then

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

which is a contradiction. For the performance ratio, notice that

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

□

Exercise 9. Describe the ratio f for the WEIGHTED SET COVER problem in terms of m, n and the set collection $\mathcal{S}$.

Exercise 10. Suppose we apply the rounding strategy above to the WEIGHTED SET COVER problem. However, instead of rounding the $x_j^* \geq 1/f$ up to 1, we round all positive x_j^* up to 1. Show that we would still have an f-approximation algorithm. (**Hint**: consider the primal complementary slackness condition.)

3 Minimum complete matching in bipartite graphs

In this problem, we are given a bipartite graph $G = (A \cup B, E)$, where $|A| \leq |B|$, along with a weight function $w : E \to \mathbb{Z}+$ which assigns a non-negative weight to each edge of G. The problem is to find a minimum weight perfect matching from A into B. The corresponding integer program can be formulated as follows.

$$\begin{aligned} \min \quad & \sum_{uv \in E} w_{uv} x_{uv} \\ \text{subject to} \quad & \sum_{v:uv \in E} x_{uv} = 1 \quad \forall u \in A, \\ & \sum_{u:uv \in E} x_{uv} \leq 1 \quad \forall v \in B, \\ & x_{uv} \in \{0,1\} \quad \forall uv \in E. \end{aligned} \tag{8}$$

In the corresponding LP, the integral condition is relaxed to $x_{uv} \geq 0$:

$$\begin{aligned} \min \quad & \sum_{uv \in E} w_{uv} x_{uv} \\ \text{subject to} \quad & \sum_{v:uv \in E} x_{uv} = 1 \quad \forall u \in A, \\ & \sum_{u:uv \in E} x_{uv} \leq 1 \quad \forall v \in B, \\ & x_{uv} \geq 0 \quad \forall uv \in E. \end{aligned} \tag{9}$$

Lemma 3.1. *Every vertex of the polyhedron defined by* (9) *is integral.*

Proof. Let $\mathbf{x}$ be any vertex of the polyhedron. Consider the subgraph H of G induced by taking all edges uv where $x_{uv} \in (0, 1)$.

If G' has a cycle C, then the cycle must be of even length. Suppose the cycle consists of edges u_1u_2, u_2u_3, $\ldots$, $u_{2k-1}u_{2k}$, $u_{2k}u_1$. Then, for any ϵ with sufficiently small absolute value, the vector $\mathbf{y}^{(\epsilon)}$ defined by

$$\begin{array}{rcll} y^{(\epsilon)}_{uv} & = & x_{uv} & uv \notin C \\ y^{(\epsilon)}_{u_1u_2} & = & x_{u_1u_2} + \epsilon & \\ y^{(\epsilon)}_{u_1u_2} & = & x_{u_2u_3} - \epsilon & \\ \ldots & & & \\ y^{(\epsilon)}_{u_{2k-1}u_{2k}} & = & x_{u_{2k-1}u_{2k}} + \epsilon & \\ y^{(\epsilon)}_{u_{2k}u_1} & = & x_{u_{2k}u_1} - \epsilon & \end{array}$$

is feasible. But then, $\mathbf{x} = (\mathbf{y}^{(\epsilon)} + \mathbf{y}^{(-\epsilon)})/2$, which is a contradiction.

If G' has no cycle, consider a maximal path $u_1u_2 \ldots u_k$. Because each vertex in A of G' has degree at least 2, u_1 and u_k belongs to B. The same trick applies. □

Corollary 3.2. *To solve the minimum complete matching problem on bipartite graphs, we can just solve for the optimal vertex of the corresponding LP, which will also be an optimal solution to the IP.*

Exercise 11. Let $G = (V, E)$ be a simple graph. The *matching polytope* $M(G)$ associated with G is defined by

$$\begin{array}{rcll} \displaystyle\sum_{v\,:\,uv\in E} x_{uv} & \leq & 1 & v \in V \\ x_{uv} & \geq & 0 & uv \in E. \end{array} \tag{10}$$

Suppose G is bipartite. Show that the vertices of $M(G)$ are exactly the characteristic vectors of (not necessarily complete nor perfect) matchings of G.

The *maximum matching problem* asks us to find a matching of maximum size in a given graph G. The previous exercise shows that, we can solve the maximum matching problem by solving the linear program $\max\{\sum_{uv\in E} x_{uv} \mid \mathbf{x} \in M(G)\}$ for an optimal vertex. The dual of this program can be written as follows.

$$\begin{array}{rrcll} \min & \displaystyle\sum_{v\in V} y_v & & & \\ \text{subject to} & y_u + y_v & \geq & 1 & uv \in E \\ & y_v & \geq & 0 & v \in V. \end{array} \tag{11}$$

A 01-solution to this program corresponds to a vertex cover of G.

Exercise 12. Suppose G is bipartite. Show that the vertices of the polyhedron defined in (11) are precisely characteristic vectors of *minimal* vertex cover of G.

Exercise 13 (König's Minimax Theorem). Let G by a bipartite graph. Show that the size of a maximum matching of G is equal to the size of a minimum vertex cover.

Exercise 14 (König's Edge Coloring Theorem). Formulate the edge coloring problem for bipartite graphs as an integer linear programming problem. Use the formulation to show that the chromatic index of every bipartite graph is equal to its maximum degree.

Exercise 15 (Birkhoff-von Neumann Theorem). A non-negative square matrix $\mathbf{A}$ is doubly stochastic if each row and each column sums up to 1. Show that every stochastic matrix $\mathbf{A}$ can be written as a convex combination of permutation matrices.

4 Scheduling on unrelated parallel machines

Consider a set of n jobs $J_1, \ldots, J_n$, which are to be processed on m machines $M_1, \ldots, M_m$. In an unrelated parallel machine environment, job J_j has a processing time of $p_{ij} \in \mathbb{Z}^+$ on machine M_i. Each machine can only process one job at a time. The objective is to devise a schedule so that the maximum completion time $C_{\max}$ is minimized.

Before discussing an LP-relaxation and rouding solution to this problem, we have a few words on notations describing scheduling problems. Using the standard notations [4,5,9], each scheduling problem is described with three fields:

- The first is the machine environment. For example, 1 indicates a *single machine*, $\mathcal{P}$ means *identical parallel machines*, R means *unrelated parallel machines*, etc.
- The second specifies special constraints or conditions on the problem. For example, $p_j = 1$ means *unit processing time*, $prec$ indicates *precedence constraints* (i.e. some job has to be processed completely before another job can start), etc.
- The third is the objective function. For example, $C_{\max}$ is the maximum completion time, commonly referred to as the *makespan* of a schedule, P indicates the total processing times, $\sum C_j$ is the total completion times, L_{max} is the maximum lateness, etc.

This section is concerned with the $\mathcal{P}||C_{\max}$ problem, which was defined earlier. More specifically, our problem does not allow *preemtion*, i.e. once a job starts on a machine, it cannot be interrupted.

Let x_{ij} be a variable indicating if job j is to be processed on machine i. The following integer program is equivalent to our problem.

$$\begin{array}{lrcll} \min & t & & & \\ \text{subject to} & \sum_{i=1}^{n} x_{ij} & = & 1 & j \in [m] \\ & \sum_{j=1}^{m} p_{ij} x_{ij} & \leq & t & i \in [n] \\ & x_{ij} \in \{0,1\}, & & \forall i \in [n], \forall j \in [m]. & \end{array} \tag{12}$$

In the corresponding LP, we relax the integer constraints to be $x_{ij} \geq 0$.

Exercise 16. Give an example showing that the integrality gap for (12) is at least m.

Exercise 17. Show that a non-empty polyhedron $P = \{\mathbf{x} \mid \mathbf{A}_1\mathbf{x} = \mathbf{b}_1, \mathbf{A}_2\mathbf{x} \leq \mathbf{b}_2, \mathbf{x} \geq \mathbf{0}\}$ always has a vertex. Moreover, suppose k is the total number of rows of $\mathbf{A}_1$ and $\mathbf{A}_2$. Show that a vertex $\mathbf{x}^*$ of P has at most m positive components.

Exercise 17 shows that the polyhedron defined by

$$\begin{array}{rcll} \sum_{i=1}^{n} x_{ij} & = & 1 & j \in [m] \\ \sum_{j=1}^{m} p_{ij} x_{ij} & \leq & t & i \in [n] \\ x_{ij} & \geq & 0 & \forall i \in [n], \forall j \in [m] \end{array}$$

always has a vertex. (Recall $p_{ij} \in \mathbb{Z}^+$.) In particular, a vertex $(\mathbf{x}^*, t^*)$ has at most $m + n$ positive variables. This means that the number of fractional (i.e. non-integral) x^*_{ij} is at most $m + n - 1$. We intepret the value of x^*_{ij} to mean "a fraction x^*_{ij} of job j is assigned to machine i." An easy counting argument implies that the number of jobs fractionally assigned is at most $m - 1$. This suggests the following algorithm.

APPROX-SCHEDULING-1

1: Use integral variables x^*_{ij} of the optimal vertex $(\mathbf{x}^*, t^*)$ to assign jobs to machines. After this step, there is a set U of at most $m-1$ jobs not yet scheduled.
2: Schedule optimally the jobs in U by looking at all $O(m^n)$ possibilities.

Let $\text{OPT}(C_{\max})$ be the optimal makespan. Then, $t^* \leq \text{OPT}(C_{\max})$. This means that the makespan of the integrally assigned jobs (in step 1) is at most $\text{OPT}(C_{\max})$. Since the jobs in U are optimally scheduled, their makespan is at most $\text{OPT}(C_{\max})$ also. Thus, algorithm APPROX-SCHEDULING-1 has approximation ratio 2. Unfortunately, the running time is exponential if the number of machines is not a constant.

Fortunately, the previous algorithm suggests the following idea. Suppose we can find a schedule in which the integral schedule's makespan is at most T, and the fractional variables x_{ij} all correspond to $p_{ij} \leq T$, which we can some how match to machines in a one-to-one manner, then the final schedule's makespan is at most $2T$.

For each positive integer T, let $S_T = \{ij \mid p_{ij} \leq T\}$, and consider the following polyhedron called $P(T)$:

$$\begin{aligned} \textstyle\sum_{i\,:\,ij \in S_T} x_{ij} &= 1 \quad j \in [m] \\ \textstyle\sum_{j\,:\,ij \in S_T} p_{ij} x_{ij} &\leq T \quad i \in [n] \\ x_{ij} &\geq 0 \quad \forall ij \in S_T. \end{aligned}$$

Let $\alpha = \max_j \min_i p_{ij}$, and β be the makespan of the *greedy schedule* which assigns each job to a machine with minimum processing time. For any $T \in [\alpha, \beta]$:

- if $P(T)$ is empty, then there is no (integral) schedule of makespan at most T. Note that, the converse is not true: if $P(T)$ is not empty, then it is not necessary that the is an integral schedule of makespan at most T. However, we will show that if $P(T)$ is not empty, then there is an integral schedule of makespan at most $2T$.
- if $P(T)$ is not empty, then $P(T')$ is not empty for all $T' \geq T$.

Let T^* be the smallest (integral) value of T in the interval $[\alpha, \beta]$ for which $P(T)$ is not empty. Clearly we can find T^* with a simple binary search in the range $[\alpha, \beta]$. Moreover, $T^* \leq \text{OPT}(C_{\max})$ because at $T = \text{OPT}(C_{\max})$ even the corresponding IP is feasible.

Exercise 17 tells us that $P(T^*)$ is pointed, and every vertex $\mathbf{x}^*$ has at most $m+n$ positive coordinates. This implies that the number of fractionally assigned jobs (according to $\mathbf{x}^*$) is at most m. Let $G = (A, B; E)$ be a bipartite graph defined as follows: A is the set of fractionally assigned jobs, B is the set of machines to which some fractional jobs were assigned, and $ij \in E$ iff x^*_{ij} is fractional. We will show that there is a complete matching from A into B. The final schedule assign integrally set jobs according to x^*_{ij}, then assign each job in A to its matched machine in B.

The complete algorithm can be summarized as follows.

APPROX-SCHEDULING-2
1: Use binary search to find the least value $T^* \in [\alpha, \beta]$ for which $P(T^*)$ is not empty.
2: Find a vertex $\mathbf{x}^*$ of $P(T^*)$.
3: Construct the bipartite graph $G = (A, B; E)$ as described above. Find a complete matching from A into B.
4: Assign integrally set jobs to corresponding machines
5: For fractionally set jobs (in A), assign them according to the complete matching just found.

Theorem 4.1. *Algorithm* APPROX-SCHEDULING-2 *has approximation ratio* 2.

Proof. It remains to show that there is a complete matching from A to B in the bipartite graph G. Consider any subset of fractionally assigned jobs $S \subseteq A$, we only need to check Hall's maching condition

that $|\Gamma(S)| \geq |S|$, where $\Gamma(S)$ is the set of neighbors of S. Let $H = (S, \Gamma(S); E(H))$ be the subgraph of G induced by S and $\Gamma(S)$. Consider the polyhedron P which is defined in the same way as $P(T^*)$ restricting to variables corresponding to edges in $E(H)$. Let $\mathbf{y}^*$ be $\mathbf{x}^*$ restricted to H. Then, $\mathbf{y}^*$ must be a vertex of P, otherwise $\mathbf{x}^*$ is not a vertex of $P(T^*)$. Thus, the number of positive components of $\mathbf{y}^*$ is at most $|S| + |\Gamma(S)|$ (Exercise 17). But, the number positive components of $\mathbf{y}^*$ is exactly $|E(H)|$. Moreover, each fractionally assigned job must be assigned to at least 2 machines. Consequently, $2|S| \leq |E(H)| \leq |S| + |\Gamma(S)|$. □

Exercise 18. A graph with n vertices is called a 1-*tree* if it has at most n edges. Show that each connected component of the bipartite graph $G = (A, B; E)$ above is a 1-tree, whose leaf nodes are all in B. Using this fact, construct a complete matching from A into B in G.

Exercise 19. Suppose $p_{ij} = p_j, \forall i$ (i.e. the machines are identical). Does algorithm APPROX-SCHEDULING-2 have approximation ratio better than 2?

Exercise 20. Suppose $P(T)$ is not empty. Show that it has a vertex whose corresponding "fractional bipartite graph" G is a forest.

Exercise 21. Give a PTAS for the problem of minimizing makespan on uniform parallel machines. In this problem, there is a speed s_i associated with each machine M_i. The processing time for job j is p_j/s_i.

5 Filtering and rounding

In this section, we illustrate the technique of *filtering and rounding* by presenting an approximation algorithm for the metric uncapacitated facility location problem. Filtering and rounding typically goes as follows:

- Formulate the optimization problem as an IP.
- Formulate the corresponding LP.
- Use an optimal solution to the LP to construct a "filtered" version of the optimization problem; this steps often involves setting some integer variables to be zero in the IP, in effect creating a restricted version of the IP. The filtered problem has optimal (integral) cost within about $(1 + \epsilon)$ of the LP.
- Finally, a rounding step produces a good integral solution to the filtered problem.

FACILITY LOCATION is a fundamental optimization problem appearing in various contexts. In the metric uncapacitated version of the problem, we are given a set F of "facilities," and a set C of "clients." The cost of opening facility $i \in F$ is f_i, and the cost of assigning client j to facility i is d_{ij}. The costs d_{ij} satisfy the triangle inequality. (You can think of the facilities and clients as points on a plane, for instance. The cost d_{ij} is the "distance" between facility i and client j.) The Metric uncapacitated facility location problem is to find a subset $O \subseteq F$ of facilities to be open and an assignment $a : C \to O$ assigning every client j to some facility $a(j)$ to minimize the objective function

$$\sum_{i \in O} f_i + \sum_{j \in C} d_{a(j),j}.$$

The first part of the objective function is the cost of opening the facilities O; the second part is the "service assignment cost."

Designate a variable x_i indicating if facility i is open and y_{ij} indicating if client j is assigned to facility i, we get the following integer program:

$$\begin{array}{lrcll} \min & \displaystyle\sum_{i\in F} f_i x_i + \sum_{i\in F, j\in C} d_{ij} y_{ij} & & & \\ \text{subject to} & \displaystyle\sum_{i\in F} y_{ij} & = & 1 & j \in C, \\ & y_{ij} & \leq & x_i & i \in F,\ j \in C, \\ & x_i, y_{ij} \in \{0,1\}, & & & i \in F,\ j \in C. \end{array} \tag{13}$$

Relaxing this integer program gives the following linear program

$$\begin{array}{lrcll} \min & \displaystyle\sum_{i\in F} f_i x_i + \sum_{i\in F, j\in C} d_{ij} y_{ij} & & & \\ \text{subject to} & \displaystyle\sum_{i\in F} y_{ij} & = & 1 & j \in C, \\ & y_{ij} & \leq & x_i & i \in F,\ j \in C, \\ & x_i, y_{ij} \in \{0,1\}, & & & i \in F,\ j \in C. \end{array} \tag{14}$$

The objective function of the LP has two parts: the open facility part $F(\mathbf{x}) = \sum_{i\in F} f_i x_i$, and the assignment part $A(\mathbf{y}) = \sum_{i,j} d_{ij} y_{ij}$. Let $(\mathbf{x}^*, \mathbf{y}^*)$ be an optimal solution to the LP. As usual, we interpret a fractional x_i^* as a partially open facility, and a fractional y_{ij}^* as a partial assignment of client j to facility i.

The filtering step. The candidate facilities to assign client j to are the facilities i with $y_{ij} > 0$. However, the corresponding assignment costs d_{ij} might be too large. Fix a parameter $\epsilon > 0$ to be determined later. We first try to filter out, for each client j, the facilities that are more than $(1+\epsilon)$ of the optimal cost. Let the current optimal (fractional) assignment cost for client j be

$$A_j^* = \sum_{i\in F} d_{ij} y_{ij}^*.$$

Define the "good candidate facility set" for j by

$$C_j = \{i \mid y_{ij}^* > 0,\ d_{ij} \leq (1+\epsilon) A_j^*\}.$$

Note that $C_j \neq \emptyset$. We next construct (using filtering) a feasible solution $(\mathbf{x}', \mathbf{y}')$ to the LP so that two conditions hold: (a) the cost of $(\mathbf{x}', \mathbf{y}')$ is not too far from the cost of $(\mathbf{x}^*, \mathbf{y}^*)$, and (b) $y'_{ij} > 0$ implies $d_{ij} \leq (1+\epsilon) A_j^*$ (in other words, set $y'_{ij} = 0$ whenever $d_{ij} > (1+\epsilon) A_j^*$). Because we have to keep $\sum_i y_{ij} = 1$, we must change the positive y_{ij}^* for which $d_{ij} \leq (1+\epsilon) A_j^*$. A simple rescaling works. Define

$$y'_{ij} = \begin{cases} \frac{y_{ij}^*}{\sum_{i\in C_j} y_{ij}^*} & i \in C_j \\ 0 & o.w. \end{cases}$$

Note that

$$A_j^* > \sum_{i\notin C_j} d_{ij} y_{ij}^* > (1+\epsilon) A_j^* \sum_{i\notin C_j} y_{ij}^* = (1+\epsilon) A_j^* \left(1 - \sum_{i\in C_j} y_{ij}^*\right).$$

In other words,

$$\frac{1}{\sum_{i\in C_j} y_{ij}^*} < \frac{1+\epsilon}{\epsilon}. \tag{15}$$

This tells us that y'_{ij} is not more than a factor of $(1+\epsilon)/\epsilon$ from y^*_{ij}. Thus, define

$$x'_i = \min\{1, x^*_i(1+\epsilon)/\epsilon\},$$

and we have the "filtered" solution $(\mathbf{x}', \mathbf{y}')$ for the LP. Moreover,

$$\begin{aligned} F(\mathbf{x}') = \sum_i f_i x'_i \leq \sum_i f_i(1+1/\epsilon)x^*_i &= (1+1/\epsilon)F(\mathbf{x}^*) \\ A(\mathbf{y}') = \sum_j \sum_i d_{ij} y'_{ij} \leq \sum_j (1+\epsilon)A^*_j \sum_i y'_{ij} &\leq (1+\epsilon)A(\mathbf{y}^*) \end{aligned}$$

The rounding step. Now that we have a good LP solution, we will use it to devise a greedy rounding procedure. Choose a client j with smallest assignment cost A^*_j. Then, open the facility $a(j)$ in C_j with the least opening cost. Note that

$$\sum_{i \in C_j} f_i x'_i \geq f_{a(j)} \sum_{i \in C_j} x'_i \geq f_{a(j)} \sum_{i \in C_j} y'_{ij} = f_{a(j)}.$$

Hence, the cost of opening facility $a(j)$ is at most the fractional cost of facilities covering j. We will not open any other facility in C_j to keep the facility opening cost low. For any j' with $C_{j'} \cap C_j \neq \emptyset$, assign j' to facility $a(j)$ also, then repeat the process.

Thus, the facility opening cost is not increased, but we have increased the assignment cost from j' to $a(j)$. Fortunately, the triangle inequality ensures that this increase is not by much. Let i be any facility in $C_{j'} \cap C_j$ (facility i could be the same as $a(j)$, which does not matter). We have

$$d_{a(j)j'} \leq d_{ij'} + d_{ij} + d_{a(j)j} \leq (1+\epsilon)A^*_{j'} + (1+\epsilon)A^*_j + (1+\epsilon)A^*_j \leq 3(1+\epsilon)A^*_{j'}.$$

Consequently, the facility opening cost is at most $(1+1/\epsilon)F(\mathbf{x}^*)$, and the assignment cost is at most

$$3(1+\epsilon)\sum_j A^*_j = 3(1+\epsilon)A(\mathbf{y}^*).$$

In total, the final cost is at most

$$(1+1/\epsilon)F(\mathbf{x}^*) + 3(1+\epsilon)A(\mathbf{y}^*) \leq \max\{3(1+\epsilon), (1+1/\epsilon)\}\text{OPT}.$$

Thus, the approximation ratio of this algorithm is $\max\{3(1+\epsilon), (1+1/\epsilon)\}$. To minimize this maximum, we choose $\epsilon = 1/3$, which yields an approximation ratio of 4.

Exercise 22 (k-MEDIAN using filtering and rounding). The k-MEDIAN problem can be defined as follows. Given a set F of facilities and a set C of clients. As in the facility location problem, there is a service cost d_{ij} if client j is assigned to facility i. We are also given an integer k, where $k \leq |F|$. There is no facility opening cost. We want to open a subset O of at most k facilities, then assign each client to the "nearest" facility in O, so as to minimize the total assignment cost. Formally, the objective function is

$$\sum_{j \in C} \min_{i \in O} d_{ij}.$$

(i) Formulate this problem as a linear integer program. Call the optimal cost for this program $\text{OPT}(IP)$.

(ii) Recall the SET COVER problem, which has a corresponding IP and a corresponding LP. Each feasible solution to the LP is called a *fractional set cover*. Suppose we have an efficient algorithm which, given a fractional set cover, returns a set cover with cost at most f times the cost of the

fractional set cover. A rounding procedure applied to the fractional set cover is an example of such algorithm. (Note that, f is actually a function of the input size, but this fact is not that important for our problem.)

Suppose that the d_{ij} may not satisfy the triangle inequality. For any $\epsilon > 0$, use the filtering and rounding method to devise an approximation algorithm for the k-median problem, where the number of open facilities is at most $(1+\epsilon)kf$, and the assignment cost is at most $(1+1/\epsilon)\text{OPT}(IP)$.

(iii) Now, assume the d_{ij} satisfy the triangle inequality. For any $\epsilon > 0$, use the filtering and rounding method to devise an approximation algorithm for the k-median problem, where the number of open facilities is at most $(1 + 1/\epsilon)k$, and the assignment cost is at most $3(1 + \epsilon)\text{OPT}(IP)$.

6 Multiway node cuts

In the MULTIWAY NODE CUT problem, we are given a vertex-weight graph $G = (V, E)$ with weight function $w : V \to \mathbb{Z}^+$, and an independent set of *terminals* $T \subset G$. The objective is to find a subset of $V - T$ whose removal disconnect the terminals from each other. Let $\mathcal{P}$ be the set of all paths connecting the terminals, then an equivalent ILP is

$$\begin{aligned} \min \quad & \sum_{v \in V-T} w_v x_v \\ \text{subject to} \quad & \sum_{v \in P \setminus T} x_v \geq 1, \quad \forall P \in \mathcal{P} \\ & x_v \in \{0, 1\} \quad \forall v \in V. \end{aligned} \tag{16}$$

The corresponding LP is

$$\begin{aligned} \min \quad & \sum_{v \in V-T} w_v x_v \\ \text{subject to} \quad & \sum_{v \in P \setminus T} x_v \geq 1, \quad \forall P \in \mathcal{P} \\ & x_v \geq 0 \quad \forall v \in V. \end{aligned} \tag{17}$$

A separation oracle for this program can be described as follows. To check if $\mathbf{x}$ is feasible, construct a directed graph D by turning each edge uv of G into two edges (u, v) and (v, u) of D. Assign a weight of x_v to edge (u, v) and a weight of x_v to edge (v, u). (For convenience, we set $x_v = 0$ if $v \in T$.) Then, find all shortest paths among all pairs of terminals in D. If one such shortest path has length < 1, then we have found a separating hyperplane. Otherwise the solution is feasible.

Consequently, we can find an optimal solution to the LP efficiently. Let $\mathbf{x}^*$ be an optimal solution to the LP. If we round all positive x_v^* up to 1, certainly we get a feasible solution to the ILP, but the approximation ratio may be too large. For instance, if the minimum positive x_v^* is $1/\rho$, then our ratio is ρ. (It seems difficult to prove a ratio less than ρ in this case.) On the other hand, if we round some positive x_v^* down to 0, the integral solution might not be feasible. Thus, before rounding we will attempt to construct another optimal solution $\mathbf{z}^*$ from $\mathbf{x}^*$ where the positive z_v^* are not too small. To do so, we will try to explore the structure of $\mathbf{x}^*$ using the notion of duality. The dual of the LP reads

$$\begin{aligned} \max \quad & \sum_{P \in \mathcal{P}} y_P \\ \text{subject to} \quad & \sum_{P: v \in P} y_P \leq c_v, \quad \forall v \in V - T \\ & y_P \geq 0 \quad \forall P \in \mathcal{P}. \end{aligned} \tag{18}$$

Let $\mathbf{y}^*$ be an optimal solution to the dual LP. The dual program can be interpreted as routing commodities between all pairs of terminals, subject to the vertex capacity constraints.

The **primal complementary slackness condition** reads: *for each* $v \in V - T$, *if* $x_v^* > 0$ *then* $\sum_{P: v \in P} y_P^* = c_v$. Thus, this condition is saying that vertex v is *saturated* whenever $x_v^* > 0$.

The **dual complementary slackness condition** reads: *for each* $P \in \mathcal{P}$, *if* $y_P^* > 0$ *then* $\sum_{v \in P \setminus T} x_v^* = 1$. We will interpret $\sum_{v \in P \setminus T} x_v^*$ as the "length" of the path P. Thus, this condition is saying that P is of length 1 whenever $y_P^* > 0$.

Our objective is to construct from $\mathbf{x}^*$ a feasible solution $\mathbf{z}^*$ whose smallest positive entry is as large as possible. Preferably $\mathbf{z}^*$ has the same cost as $\mathbf{x}^*$. In order to do this, we first need to know roughly which x_v^* are positive.

For each terminal t_i, let S_i be the set of all nodes reachable from t_i within distance zero. Due to the constraints of the primal LP, the S_i are pair-wise disjoint. Consider a path P from t_i to t_j. This path will have to cross the boundary of S_i to the outside and then cross the boundary of S_j to step into S_j. For each i, let B_i be the set of vertices reachable from S_i in one edge, other than those in S_i. By the definition of S_i, x_v^* must be positive for all v in B_i. We thus have found some set of vertices v with positive x_v^*.

Another simple observation is that, if $v \in B_i \cap B_j$ for $i \neq j$, then $x_v^* = 1$. Let B be the union of all B_i. Then, B is naturally partitioned into two subsets $X_1 \cup X_2$, where X_1 consists of all vertices v which are in the intersection of two distinct B_i and B_j.

In order for $\mathbf{z}^*$ to have the same cost ax $\mathbf{x}^*$, we will try to maintain both of the complementary slackness conditions. The primal condition is easy to maintain if $\{v \mid z_v^* > 0\} \subseteq \{v \mid x_v^* > 0\}$. To maintain the dual condition, we have to make sure that each path P with $y_P^* > 0$ has length 1, still. Thus, we want to know which vertices such a path contain.

Consider a path P from t_i to t_j, where $y_P^* > 0$. The two end segments of P will contain vertices from S_i and S_j. In the middle, there must be at least one vertex in $X_1 \cup X_2$. If a vertex v of X_1 belongs to P, then all other vertices u on P must have $x_u^* = 0$. In particular, in this case P will not contain any vertex in X_2. On the other hand, suppose P does not contain any vertex from X_1, then it must contain two vertices $u \in B_i$ and $v \in B_j$ from X_2. Can there be another vertex w other and u and v, where w belongs to some B_k? Here, k could be equal to i or j. Let P_i be the part of P from t_i to w, and P_j be the part of P from w to t_j. Let P_k be a shorest path from w to t_k (whose length is x_w^*). Now, either $P_i P_k$ is a valid path, or $P_j P_k$ is a valid path, both of which have length strictly less than 1, contradicting the feasibility of $\mathbf{x}^*$.

We have just shown that every path P with $y_P^* > 0$ must either contain exactly one vertex from X_1 or, exclusively, exactly two vertices from X_2. Now, set

$$z_v^* = \begin{cases} 1 & v \in X_1 \\ 1/2 & v \in X_2 \\ 0 & o.w. \end{cases}.$$

Then, it is easy to see that $\mathbf{z}^*$ is feasible. Moreover, $\mathbf{z}^*$ satisfies both of the complementary slackness conditions. Thus, $\mathbf{z}^*$ is a half-integral optimal solution to the LP. Moreover, $\mathbf{z}^*$ can be constructed from $\mathbf{x}^*$ in polynomial time. It is interesting to note that we did not have to solve the dual program at all. We merely used it for analytical purposes. This analysis leads directly to the following theorem.

Theorem 6.1. *A* 2*-approximation algorithm for the* MULTIWAY NODE CUT *problem can be obtained by rounding all positive* z_v^* *up to* 1. *In effect, the set* B *is our approximated node cut.*

Exercise 23. When $k = 2$, the MULTIWAY NODE CUT problem is equivalent to finding a minimum cut, which can be done in polynomial time. Thus, it would be nice to have an approximation ratio which is a function of k, and which is equal to 1 when $k = 2$. Devise a $(2 - 2/k)$-approximation algorithm for the MULTIWAY NODE CUT problem.

Historical Notes

Recent books on approximation algorithms include [1, 6, 13, 15].

It has been shown that approximating WEIGHTED VERTEX COVER to within $10\sqrt{5} - 21$ [2] is **NP**-hard. A recent major result using PCP by Dinur and Safra showed that approximating WVC to within 1.3606 is **NP**-hard [3]. See also the paper by Khot and Regev [8]. The approximation ratio 2 for VERTEX COVER is still the best known to date.

Algorithm APPROX-SCHEDULING-2 is from [10]. A PTAS for the uniform parallel machine case was devised in [7]. See the surveys [4, 5, 9] for more results on scheduling problems.

The filtering and rounding method was developed by Lin and Vitter [11]. Shmoys, Tardos, and Aardal [14] used this method to give a 3.16-approximation algorithm for the metric UFL problem. The 4-approximation we presented was a variance of their idea. For a brief historical discussion on the facility location problem, see [12]. Their method of dual-fitting is also very interesting, which we will discuss in a later lecture.

References

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

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

[3] I. DINUR AND S. SAFRA, *On the hardness of approximating minimum vertex cover*, Ann. of Math. (2), 162 (2005), pp. 439–485.

[4] R. L. GRAHAM, E. L. LAWLER, J. K. LENSTRA, AND A. H. G. RINNOOY KAN, *Optimization and approximation in deterministic sequencing and scheduling: a survey*, Ann. Discrete Math., 5 (1979), pp. 287–326. Discrete optimization (Proc. Adv. Res. Inst. Discrete Optimization and Systems Appl., Banff, Alta., 1977), II.

[5] L. HALL, *Approximation algorithms for scheduling*, in Approximation Algorithms for NP-Hard Problems, D. Hochbaum, ed., PWS Publishing Company, 1997, pp. 1–45.

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

[7] D. S. HOCHBAUM AND D. B. SHMOYS, *A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach*, SIAM J. Comput., 17 (1988), pp. 539–551.

[8] S. KHOT AND O. REGEV, *Vertex cover might be hard to approximate to within* $2 - \epsilon$, in Proceedings of the 18th IEEE Annual Conference on Computational Complexity (CCC), 2003, pp. 379–386.

[9] J. K. LENSTRA AND D. SHMOYS, eds., *Networks and matroids; Sequencing and scheduling*, North-Holland Publishing Co., Amsterdam, 1998. Dedicated to the memory of Eugene L. Lawler, Math. Programming **82** (1998), no. 1-2, Ser. B.

[10] J. K. LENSTRA, D. B. SHMOYS, AND É. TARDOS, *Approximation algorithms for scheduling unrelated parallel machines*, Math. Programming, 46 (1990), pp. 259–271.

[11] J.-H. LIN AND J. S. VITTER, *Approximation algorithms for geometric median problems*, Inform. Process. Lett., 44 (1992), pp. 245–249.

[12] M. MAHDIAN, E. MARKAKIS, A. SABERI, AND V. VAZIRANI, *A greedy facility location algorithm analyzed using dual fitting*, in Approximation, randomization, and combinatorial optimization (Berkeley, CA, 2001), vol. 2129 of Lecture Notes in Comput. Sci., Springer, Berlin, 2001, pp. 127–137.

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

[14] D. B. SHMOYS, É. TARDOS, AND K. AARDAL, *Approximation algorithms for facility location problems (extended abstract)*, in STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, ACM Press, 1997, pp. 265–274.

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