CSC2421h - Approximation Algorithms
Lecture #9

Lecturer: Allan Borodin
Scribe: Greg Brunet

March 17, 2005

1 Introduction

In this lecture, we discuss some scheduling problems and an integrality gap result for Vertex Cover. In section 2, we introduce some standard notation and features of a typical scheduling problem, and briefly discuss two such problems that have optimal greedy algorithms. In sections 3 and 4, we obtain approximation algorithms for some NP-Hard scheduling problems by using a general method of "relaxation". From now on, a relaxation is no longer just the removal of integrality constraints. Instead, problem B is a relaxation of problem A if any solution of problem A is also a solution of B. The general method of using relaxation is then as follows:

- Formulate a relaxed version of the problem and solve it optimally to obtain a solution x^*.
- Use x^* as a "starting point" for constructing a solution $\overline{x}$ to the original problem.

In particular, we hope that $cost(\overline{x}) \leq C{\cdot}cost(x^*)$, for some constant C. It then follows that $cost(\overline{x}) \leq C{\cdot}cost(OPT)$, since x^* is a lower bound on the true optimal solution.

Currently, there is no known polynomial-time solvable IP / LP formulation for the Vertex Cover problem that has integrality gap better than $2-o(1)$, and it may be the case that no such formulation exists. However, proving such statements may be related to the P vs. NP problem, since linear programming is complete for P. Instead, it is possible to prove nonexistence results for certain families of LP's. In [1], Arora, Bollobás, and Lovász show that for three such families of LP's, the integrality gap for Vertex Cover is $2 - o(1)$. In Section 5, we present their results for one family of LP's.

2 Scheduling problems

The scheduling problems we consider in this lecture share some common elements. Each problem contains a set $\{1, \ldots, n\}$ of jobs to be scheduled, where every job j has a processing time p_j. The processing time of a job is simply the amount of time the job takes to finish after it has been scheduled. There is a also a set of processors, where each processor may schedule only one job at any given time. Finally, there may be some constraints on how the jobs are to be scheduled, and there must be a specific optimization criterion for each problem (more on this below). We seek a schedule that assigns each job to a given processor[1]. Given a schedule, we denote the completion time of a job j by C_j. Sometimes we write C_j^* or $\overline{C}_j$ for the completion time of job j with respect to a given schedule S^* or $\overline{S}$, respectively. The somewhat cryptic notation we use to describe the problems is standard in scheduling theory (introduced in [2]). Because we only study a small subset of the possible problems, we will only describe the relevant notation. A problem is defined by a triple $x|y|z$ where:

[1]The problems we study in this lecture must schedule every job, but this is not true in general

- x defines the machine environment. We study the case where $x = P1$, which indicates only one processor. In general, Pm indicates m identical parallel machines.
- y defines side constraints and characteristics of the jobs. We use r_j to indicate that jobs have release times. That is, a job j must be scheduled at time t, where $t \geq r_j$. Jobs may also have a precedence ordering that constrains the order in which jobs may be scheduled, indicated by *prec*. We write $i \leq_{prec} j$ to mean that job i must be scheduled before job j. Additionally, job preemption may be allowed, indicated by *pmtn*. This means that the processing of any job may be interrupted and resumed later. Every job must still complete, so that the total processing time of all parts of job i is still equal to p_i.
- z defines the optimality criterion (i.e., the objective function to minimize / maximize). The two optimality criterion we focus on are $\sum C_j$ and $\sum w_j C_j$. If $\sum C_j$ is present, we are minimizing the *average completion time* by minimizing $\sum C_j$. If $\sum w_j C_j$ is present, then each job j has a weight $w_j > 0$, and we are minimizing the *average weighted completion time* by minimizing $\sum w_j C_j$.

Specifically, the problems we study in this lecture are:

1. $P1||\sum C_j$
2. $P1||\sum w_j C_j$
3. $P1|r_j, pmtn|\sum C_j$
4. $P1|r_j|\sum C_j$
5. $P1|prec|\sum w_j C_j$
6. $P1|r_j, prec|\sum w_j C_j$

Using exchange arguments, it can be shown that SPT (shortest processing time) is an optimal greedy algorithm for the first problem, and that Smith's rule, which orders jobs so that $p_1/w_1 \leq \ldots \leq p_n/w_n$, gives an optimal greedy algorithm for the second problem. We will see in section 3 that there is an optimal greedy algorithm for the third problem. The last three problems are NP-hard, and are studied in Sections 3 and 4.

3 $P1|r_j|\sum C_j$

In this section, we show that by using the problem $P1|r_j, pmtn|\sum C_j$ as a relaxed version of the problem $P1|r_j|\sum C_j$, we can obtain obtain a 2-approximation algorithm for $P1|r_j|\sum C_j$.

3.1 $P1|r_j, pmtn|\sum C_j$ as a relaxation

It is clear that any solution of $P1|r_j|\sum C_j$ is also a solution of $P1|r_j, pmtn|\sum C_j$, but not vice versa. Thus, our general notion of a relaxation holds in this case. It turns out that we can solve $P1|r_j, pmtn|\sum C_j$ optimally using the SRPT rule (shortest remaining processing time):

- each time a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among all other available jobs.

Theorem 3.1 *SRPT is optimal for $P1|r_j, pmtn|\sum C_j$.*

Proof (exchange argument) Suppose that we have an optimal schedule S such that at some time t, a job j is being processed, but there is an available job i (i.e., $t \geq r_i$) with a shorter remaining processing time than j. Let p_i^r and p_j^r be the remaining processing times for jobs i and j at time t, respectively. By definition, $p_i^r < p_j^r$. We now perform an exchange to obtain a feasible schedule S' as follows: use the first p_i^r units of time that were being used for either job i or j after time t to process job i until completion. Then, use the remaining p_j^r units of time that were being used to process jobs i and j after time t to process job j only. Then we have constructed a schedule S' such that $C_i' < C_j$ and $C_j' = C_i$. Because all other jobs in S' have the same completion time as in S, we have:

$$\sum_k C_k' = \sum_{k \neq i,j} C_k' + C_i' + C_j' < \sum_{k \neq i,j} C_k + C_i + C_j = \sum_k C_k,$$

which contradicts the fact that S was optimal. ∎

3.2 Obtaining a solution to $P1|r_j|C_j$ from the relaxation

Let S^* be the optimal solution to $P1|r_j, ptmn|\sum C_j$ found by $SRPT$. We now show how to obtain a 2-approximation algorithm for $P1|r_j|C_j$ by using the solution S^* to construct a solution $\overline{S}$ of $P1|r_j|C_j$. The procedure, call it `Remove-Pmtn`, is as follows:

- reorder S^* so that $C_1^* \leq C_2^* \leq \cdots \leq C_n^*$;
- use this ordering to construct $\overline{C}_1 \leq \overline{C}_2 \leq \cdots \leq \overline{C}_n$ (a nonpreemptive schedule constructed by successively scheduling the next job in the ordering, waiting until its release time if necessary).

Theorem 3.2 *The procedure* `Remove-Pmtn` *is a 2-approximation algorithm for $P1|r_j|C_j$.*

Proof First note that in the schedule $\overline{S}$, any idle time ends at the release time of some job k. So, for each job j, there is no idle time between $\max_{k \leq j}\{r_k\}$ and the completion time $\overline{C}_j$. Thus,

$$\overline{C}_j \leq \max_{k \leq j}\{r_k\} + \sum_{k=1}^{j} p_k.$$

Now, certainly $r_k \leq C_k^*$ for any job k. Thus, for $k \leq j$, we have $r_k \leq C_k^* \leq C_j^*$ by the ordering of S^*. Also,

$$\sum_{k=1}^{j} p_k \leq C_j^*,$$

since all of the jobs $k \leq j$ finished before job j in S^*. Therefore, $\overline{C}_j \leq 2 \cdot C_j^*$ for any job j. It then follows that

$$\sum_{j=1}^{n} \overline{C}_j \leq 2 \sum_{j=1}^{n} C_j^* \leq 2 \cdot \text{OPT},$$

because $\sum C_j^*$ is a lower bound on the optimal (nonpreemptive) schedule. ∎

4 $P1|prec|\sum w_j C_j$ and $P1|r_j, prec|\sum w_j C_j$

In this section, we show how to obtain a 2-approximation algorithm for $P1|prec|\sum w_j C_j$ using relaxations. The formulation and analysis for $P1|prec|\sum w_j C_j$ given in sections 4.1 and 4.2 can be extended to obtain a 3-approximation algorithm for $P1|r_j, prec|\sum w_j C_j$ without too much additional effort. Additionally, two other formulations of $P1|prec|\sum w_j C_j$ are given in sections 4.3 and 4.4. We begin by describing the relaxed version of the problem $P1|prec|\sum w_j C_j$.

4.1 A relaxation

Consider the following natural formulation for $P1|prec|\sum w_j C_j$:

$$\text{minimize } \sum w_j C_j \tag{1}$$

subject to

$$\begin{aligned} C_j &\geq C_k + p_j, & &\text{if } k \leq_{prec} j & (2)\\ C_j \geq C_k + p_j \text{ OR } C_k &\geq C_j + p_k, & &\text{for each } j,k & (3)\end{aligned}$$

If the jobs had release times, then we would have to add the additional constraint $C_j \geq r_j + p_j$ for each j. Now, the solutions to the above formulation correspond to optimal solutions of $P1|prec|\sum w_j C_j$. However, the set of constraints in (3) involves an "OR", which is not a linear constraint. Thus, we replace this set of constraints by a set of necessary constraints. For each $S \subseteq \{1, \ldots, n\}$, define $p(S) = \sum_{i \in S} p_i$ and $p^2(S) = \sum_{i \in S} p_i^2$. Then we instead require that

$$\sum_{i \in S} p_i C_i \geq \frac{1}{2}(p(S)^2 + p^2(S)), \text{ for each } S \subseteq \{1, \ldots, n\}. \tag{4}$$

In fact, the above conditions are necessary for any feasible schedule on one machine.

Lemma 4.1 *Suppose that $C_1, \ldots, C_n$ are the completion times for the jobs in any feasible schedule on one machine. Then the C_j's satisfy the constraints given in (4).*

Proof Without loss of generality, we can assume that $S = \{1, \ldots, n\}$, because if $S \neq \{1, \ldots, n\}$, then we can just ignore the jobs not in S, view it as the entire set of jobs, and apply the same argument as follows. Reorder the jobs so that $C_1 \leq \ldots \leq C_n$. Then for any job i, $C_i \geq \sum_{j=1}^{i} p_j$ (this is equality if jobs do not have release times). Thus,

$$\sum_{i=1}^{n} p_i C_i \geq \sum_{i=n}^{n} p_j \sum_{j=1}^{i} p_j = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} p_i p_j = \frac{1}{2}(p^2(S) + p(S)^2), \tag{5}$$

which is what we wanted. ∎

Note that the formulation given in this section has an exponential number of constraints. However, there is a polynomial time algorithm that can determine whether a fractional solution is feasible, and if not, returns a violated constraint. Such a procedure is called a *separation oracle*. Therefore, the ellipsoid algorithm can be used to solve this LP in polynomial time [4].

4.2 Obtaining a solution to $P1|prec|\sum w_j C_j$ from the relaxation

We begin by solving the LP relaxation given by (1), (2), and (4) in the previous section to obtain a solution S^*. The procedure to obtain a solution $\overline{S}$ of $P1|prec|\sum w_j C_j$ from S^* is as follows:

- reorder the jobs in S^* so that $C_1^* \leq \ldots \leq C_n^*$;
- schedule the jobs in the same order $\overline{C}_1 \leq \ldots \leq \overline{C}_n$ to obtain $\overline{S}$.

Note that because there are no release times, there is no idle time in the schedule $\overline{S}$. Additionally, the ordering of S^* preserves the precedence relations because if $k \leq_{prec} j$, we have $C_k^* \leq C_j^*$ by inequality (2).

Lemma 4.2 *The above procedure gives a 2-approximation algorithm for $P1|prec|\sum w_j C_j$.*

Proof It is enough to show that $C_j^* \geq \frac{1}{2}\sum_{k\leq j} p_k$, since $\overline{C}_j = \sum_{k=1}^{j} p_k$ by the ordering of $\overline{S}$. If we let $S = \{1, \ldots, j\}$, then

$$\begin{aligned}
\sum_{k=1}^{j} p_k C_j^* &\geq \sum_{k=1}^{j} p_k C_k^* && \text{(since } C_j^* \geq C_k^* \text{ for } j \geq k)\\
&\geq \frac{1}{2}(p^2(S) + p(S)^2) && \text{(by inequality (4))}\\
&\geq \frac{1}{2}p(S)^2 && \text{(since } p^2(S) \geq 0).
\end{aligned}$$

Notice that the first summation above can be written as $(\sum_{k=1}^{j} p_k)C_j^*$, which is just $p(S)C_j^*$. Thus,

$$p(S)C_j^* \geq \frac{1}{2}p(S)^2, \tag{6}$$

giving $C_j^* \geq \frac{1}{2}p(S) = \frac{1}{2}\overline{C}_j$. ∎

4.3 Avoiding an exponential system

To avoid an exponential number of conditions for $P1|prec|\sum w_j C_j$, we can introduce auxiliary variables δ_{ij} with intended meaning as follows:

$$\begin{aligned}
\text{For } i \neq j,\ \delta_{ij} &= \begin{cases} 1, & \text{if job } i \text{ gets scheduled before job } j\\ 0, & \text{otherwise.}\end{cases}\\
\delta_{ii} &= 0.
\end{aligned}$$

Then we formulate the problem as:

$$\text{minimize } \sum w_j C_j \tag{7}$$

subject to

$$\begin{aligned}
p_j + \sum_{i=1}^{n} p_i \delta_{ij} &= C_j, && \text{for all } j && (8)\\
\delta_{ij} + \delta_{ji} &= 1, && \text{for all pairs } i,j && (9)\\
\delta_{ij} + \delta_{jk} + \delta_{ki} &\leq 2, && \text{for all } i,j,k && (10)\\
\delta_{ij} &= 1, && \text{if } i \leq_{prec} j. && (11)
\end{aligned}$$

The IP is $\delta_{ij} \in \{0, 1\}$, whereas the LP is just $\delta_{ij} \geq 0$ (since $\delta_{ij} \leq 1$ is guaranteed). Condition (8) says that the completion time of job j is equal to the processing time of job j plus the processing time of all jobs that are scheduled before it. Condition (9) says that for all jobs i and j, either job i is scheduled before job j or job j is scheduled before job i, but not both. Condition (10) ensures transitivity. For example, if job i is scheduled before job j, and job j is scheduled before job k, then job k cannot be scheduled before job i. Finally, condition (11) just preserves the precedence relations. It can be shown that this LP satisfies the inequalities (2) and (4) from section 4.1, and so a similar analysis yields an upper bound of 2.

4.4 A very different formulation

We can also formulate the problem $P1|prec|\sum w_j C_j$ in a way that involves time. Introduce auxiliary variables x_{jt} for $1 \leq j \leq n$ and $0 \leq t \leq T$, where $T = \sum_{j=1}^{n} p_j$. The intended meaning of x_{jt} is that $x_{jt} = 1$ if job j completes at time t, and $x_{jt} = 0$ otherwise. The formulation is as follows:

$$\text{minimize } \sum_{j=1}^{n} w_j \sum_{t=1}^{T} t \cdot x_j t, \tag{12}$$

subject to

$$\sum_{t=1}^{T} x_{jt} = 1, \qquad \text{for all } 1 \leq j \leq n \tag{13}$$

$$\sum_{s=1}^{t} x_{js} \geq \sum_{s=1}^{t+p_k} x_{ks}, \qquad \text{if } j \leq_{prec} k,\ 1 \leq t \leq T - p_k \tag{14}$$

$$\sum_{j=1}^{n} \sum_{s=t}^{t+p_j-1} x_{js} \leq 1, \qquad \text{for } 1 \leq t \leq T \tag{15}$$

$$x_{jt} = 0, \qquad \text{if } t < p_j. \tag{16}$$

The IP is then $x_{jt} \in \{0,1\}$. The formulation given here is also exponential, but it is possible to give a compact LP relaxation, called an *interval-indexed formulation* [3], that turns out to be polynomial sized. The idea is that time is divided into the intervals $[1,1], (1, 1+\epsilon], (1+\epsilon, (1+\epsilon)^2], ((1+\epsilon)^2, (1+\epsilon)^3], \ldots$, where $\epsilon > 0$ is arbitrarily small. The LP only says what interval a job is completed in (rather than at what time), and since the completion times within a given interval are at most a factor of $(1+\epsilon)$ apart, the scheduling within each interval is "easy".

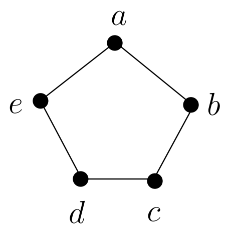
5 The Integrality Gap for Vertex Cover in a family of LP's

Here we focus on a single family of relaxations, call it $LP_{\beta n}$, which allows only the variables $x_1, x_2, \ldots, x_n \in [0,1]$ for the vertices (and no auxiliary variables) with constraints of the form $A \cdot x \geq b$, where each row of the constraint matrix A has at most βn nonzero variables. Naturally for Vertex Cover, we are minimizing $\sum_{i=1}^{n} x_i$ subject to the constraints. It will be shown that any relaxation of Vertex Cover in the family $LP_{\beta n}$ has integrality gap $2 - o(1)$. Before we can do this, we must give a preliminary definition and an important lemma.

Definition A *fractional γ-coloring* of a graph $G = (V, E)$ is a collection $\mathcal{C} = \{U_1, \ldots, U_N\}$ of independent sets of vertices of G such that every vertex v is in at least N/γ members of $\mathcal{C}$. The *fractional chromatic number* of G is

$$\chi_f(G) = \inf\{\gamma : G \text{ has a fractional } \gamma\text{-coloring}\}.$$

Example Consider the odd cycle $G = C_5$ shown below.



Note that $\{\{a,c\}, \{a,d\}, \{b,d\}, \{b,e\}, \{c,e\}\}$ is a fractional 2.5-coloring with $N = 5$ and $\gamma = 2.5$, so that $\chi_f(G) \leq 2.5$. However, the chromatic number $\chi(G)$ of any odd cycle is 3.

Lemma 5.1 *Let $0 < \alpha, \delta < 1/2$ be constants. Then there are constants $\beta = \beta(\alpha, \delta) > 0$ and $n_0 = n_0(\alpha, \beta, \delta)$ such that for every $n \geq n_0$ there is a graph $H = (V, E)$ with $|V| = n$ that satisfies the following conditions:*

1. *The independence number of H is at most αn (therefore the minimum vertex cover has size at least $(1-\alpha)n$), and*

2. *every subgraph H' of H induced by a subset of at most βn vertices has fractional chromatic number at most $2 + \delta$.*

Lemma 5.1 is proven using the probabilistic method. Roughly speaking, the probabilistic method is used to prove the existence of combinatorial objects (such as graphs) without explicitly constructing them. A probability model is defined on a large class of objects (for example, graphs with n vertices), and the occurrence of a desired structure is an event. If the event is shown to have positive probability, then some object with the desired structure exists. Now, given Lemma 5.1, we can prove the main result in this section.

Theorem 5.2 *For all constants $0 < \alpha, \delta < 1/2$, there is a constant $\beta = \beta(\alpha, \delta) > 0$ such that any linear relaxation for Vertex Cover in the family $LP_{\beta n}$ has integrality gap at least $(1 - \alpha)(\frac{2+\delta}{1+\delta})$.*

Proof (From [1]) Let H be the graph and β the constant constructed in Lemma 5.1 with α and δ. We show that the vector with all coordinates equal to $\frac{1+\delta}{2+\delta}$ is feasible for any linear relaxation of H in which each constraint involves at most βn variables. Therefore, the integrality gap is $\geq (1 - \alpha)(\frac{2+\delta}{1+\delta})$.

Suppose for each $I \subset \{1, \ldots, n\}$ with $|I| \leq \beta n$, the linear relaxation contains the constraints

$$A_I{\cdot}x \geq b_I$$

where A_I is an arbitrary constraint matrix with nonzero entries only in the columns corresponding to I (note that A_I may depend on H). Then the solution associated with these constraints is a polytope $P_I = Q_I \times [0, 1]^{n-|I|}$, where Q_I is a polytope on the coordinates in I. The solution polytope for the full relaxation is then $P = \cap_I P_I$. Thus, if we show that the vector consisting of all $\frac{1+\delta}{2+\delta}$'s is in each polytope P_I, then it is in P.

Let I be any subset of at most βn vertices of H and let $\{U_1, \ldots, U_N\}$ be its fractional $(2 + \delta)$-coloring (which exists by Lemma 5.1). Each vertex in I is contained in at least $1/(2 + \delta)$ fraction of U_i's, but we can assume without loss of generality that it is contained in exactly $1/(2 + \delta)$ fraction of the U_i's (simply remove it from some of the U_i's if need be). Note that each $I \backslash U_i$ is a vertex cover in the subgraph induced by I, and so can be extended to a vertex cover of the entire graph. Therefore, for $1 \leq k \leq N$, any vector $(x_1, \ldots, x_n)$ defined by $x_i = 1$ if $i \in I \backslash U_k$ is feasible for $A_I{\cdot}x \geq b_I$.

Now let $v_1, v_2, \ldots, v_N \in \mathcal{R}^n$ where $v_k = (x^k_1, \ldots, x^k_n)$ and

$$x^k_i = \begin{cases} 1, & \text{if } i \in I \backslash U_k \\ \frac{1+\delta}{2+\delta}, & \text{otherwise.} \end{cases}$$

Let $\overline{v} = \frac{1}{N} \sum_{k=1}^N v_k$. Each vertex in I lies in exactly $1 - 1/(2 + \delta) = (1 + \delta)/(2 + \delta)$ fraction of the vertex covers. Therefore, for each $1 \leq i \leq n$, we have $\frac{1}{N} \sum_{k=1}^N x^k_i = \frac{1+\delta}{2+\delta}$. So the vector $\overline{v}$ is precisely the vector of all $\frac{1+\delta}{2+\delta}$'s. Since we defined the v_i's so that each v_i satisfies $A_I{\cdot}x \geq b_I$, by convexity of the solution polytope P_I, we know that $\overline{v}$ also satisfies $A_I{\cdot}x \geq b_I$.

Thus, we have shown that for any subset I of at most βn vertices, the vector $\overline{v}$ consisting of all $\frac{1+\delta}{2+\delta}$'s is in P_I. Therefore, $\overline{v}$ is in the entire solution polytope P, and we are done. ∎

References

[1] S. Arora, B. Bollobás, and L. Lovász. Proving integrality gaps without knowing the linear program. *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS-02)*, pages 313–322, 2002.

[2] 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. *Annals of the Discrete Mathematics*, pages 287–326, 1979.

[3] L.A Hall, B. Shmoys, and J. Wein. Scheduling to minimize the average completion time: on-line and off-line algorithms. *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 142–151, 1996.

[4] M. Queyranne. Structure of a simple scheduling polyhedron. *Mathematical Programming*, 58:263–285, 1993.