CS 369P: Polyhedral techniques in combinatorial optimization
Instructor: Jan Vondrák
Lecture date: November 2, 2010 **Scribe: Roy Frostig**

1 Iterated rounding

We are leaving the realm of P and switching to NP-hard problems. Still, we want to use polyhedral descriptions to our advantage. One way to use an LP for an NP-hard problem is to show that a vertex solution has some nice properties that allow us to find a solution which approximates the optimal one in some sense. In particular, we use the *rank argument* to prove that a vertex solution must assign some variable a special value (e.g. integer or half integer), and then exploit this fact for rounding purposes. The iterated rounding approach usually works as follows. Given a linear program for our problem,

1. Solve and find a vertex solution of the LP.
2. Prove that a vertex has desirable properties (e.g. it assigns some variable the value 0 or 1), or that, otherwise, there must exist some constraint that we can discard and guarantee that we will not violate too greatly in the future.
3. Return to 1 and re-solve the LP until we have a solution made of desirable variable values.

Let's take the bipartite matching polytope,

$$P = \{\mathbf{x} \geq 0 : \forall v; x(\delta(v)) \leq 1\},$$

as an example of a polytope whose vertices have nice properties provable by a rank argument. (We already know that P is an integer polytope but let's forget that for now.)

Lemma 1 *For every vertex of P, there exists a variable $x_e \in \{0, 1\}$.*

Proof: Assume that $0 < x_e < 1$ for all $e \in E$. If $\mathbf{x}$ is a vertex of P, there are $|E|$ linearly independent tight constraints. That is, $x(\delta(v)) = 1$ for all $v \in W$ where W is some set such that $|W| = |E|$. Since $0 < x_e < 1$ and $x(\delta(v)) = 1$ in W, each vertex in W has degree at least 2. This means

$$2|W| \leq \sum_{v \in W} \deg(v) \leq 2|E| = 2|W|$$

which implies that every vertex in W has degree exactly 2 and every vertex not in W has degree 0. It follows that $E[W]$ is a disjoint union of cycles. Also, $E[W]$ is a bipartite graph; let W_1 and W_2 be its vertex classes. We have

$$\begin{cases} \displaystyle\sum_{v \in W_1} x(\delta(v)) &= x(E) \\ \\ \displaystyle\sum_{v \in W_2} x(\delta(v)) &= x(E) \end{cases}$$

which implies that the constraints $x(\delta(v)) = 1$ for $v \in W_1 \cup W_2$ are not linearly independent. □

2 The Generalized Assignment Problem

Consider a set J of jobs and a set M of machines. For every job j and machine i, we are given c_{ij} as the cost of processing job j on machine i and p_{ij} as the processing time of job j on machine i. Our goal is to assign all the jobs to machines so that

- every machine processes all of its jobs within time T, and
- the total cost (over all machines) is minimized.

Definition 2 (Makespan) *In an assignment of jobs to machines $(J_i : i \in M)$, we define the makespan as $\max_{i \in M} \sum_{j \in J_i} p_{ij}$, i.e. the maximum, over all machines, of the total processing time of jobs assigned to the machine.*

Notice that even if we restrict M to two machines, $p_{ij} = p_j$ depends only on j, and $\sum_j p_j = 2T$ (that is, taking processing time to be machine-independent), then determining whether these jobs can be processed within time T is equivalent to the *partition problem*. (In the partition problem, we are given integers $a_1, \ldots, a_n$ and asked whether there is a partition $[n] = A \cup B$ such that $\sum_{i \in A} a_i = \sum_{i \in B} a_i$.) The partition problem is NP-complete, so generalized assignment is NP-hard.

It is also known that the generalized assignment problem is "more difficult" than the partition problem in the sense that the partition problem has a fully polynomial-time approximation scheme (approximation arbitrarily close to 1), but generalized assignment is APX-hard (i.e. it is NP-hard to find an α-approximate solution for some fixed $\alpha > 1$).

Theorem 3 (Shmoys, Tardos) *There is an efficient algorithm such that if there exists an assignment of cost C and makespan T then the algorithm finds an assignment of cost at most C and makespan at most $2T$.*

Let E be the set of pairs (edges) such that job j can be assigned to machine i. We take on the convention that if $p_{ij} > T$, we remove (i, j) from E. (Since the optimal solution cannot use such edges, we still keep the optimum as a feasible solution.) We write an LP inspired by the bipartite matching LP:

$$\begin{aligned} \min \sum_{(i,j) \in E} c_{ij} x_{ij} : \quad & \forall j \in J; \quad \sum_{i:(i,j) \in E} x_{ij} = 1, \\ & \forall i \in \tilde{M}; \quad \sum_{j:(i,j) \in E} p_{ij} x_{ij} \leq T_i, \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned}$$

where initially $\tilde{M} = M$ and $T_i = T$ for all $i \in M$ (these will change as we modify them between invocations of the LP). What are the useful properties of a vertex solutions to this LP?

Lemma 4 *If $\mathbf{x}$ is a vertex solution of the LP, then either*

1. *there is a variable $x_{ij} \in \{0, 1\}$, or*
2. *there is a machine i such that $\deg_E(i) \leq 1$, or $\deg_E(i) = 2$ and $\sum_{j:(i,j) \in E} x_{ij} \geq 1$.*

Proof: If there exists $x_{ij} \in \{0,1\}$ then the proof is complete, so assume $0 < x_{ij} < 1$ for all $(i,j) \in E$. By the properties of a vertex solution, there exist $|E|$ linearly independent tight constraints, i.e. for some $J' \subseteq J, M' \subseteq \tilde{M}$, $|J'| + |M'| = |E|$ and

$$\begin{aligned} \forall j \in J'; \quad & \sum_{i:(i,j)\in E} x_{ij} \quad = 1, \\ \forall i \in M'; \quad & \sum_{j:(i,j)\in E} p_{ij} x_{ij} \quad = T_i. \end{aligned}$$

If $\deg(i) \leq 1$ for some $i \in \tilde{M}$ then the proof is complete, so assume that $\deg(i) \geq 2$ for all machines i. Moreover, we know $\deg(j) \geq 2$ for all jobs j since we need at least two values in the interval $(0,1)$ to sum to 1. Now,

$$2|E| = 2|J' \cup M'| \leq \sum_{v \in J' \cup M'} \deg(v) \leq 2|E|$$

so we must have equality throughout and we know that the degrees of all machines in M' and all jobs in J' are exactly 2, and all other degrees are 0. Hence $E[J' \cup M']$ is a disjoint union of cycles, i.e. a biparite graph with classes J', M'. Consider a single cycle C in $E[J' \cup M']$. Along C, vertices alternate in membership between M' and J'. Suppose for contradiction that for all machines $i \in M' \cap Z$, $\sum_{j:(i,j)\in C} x_{ij} < 1$. We have

$$\begin{aligned} \frac{|C|}{2} &= \sum_{i \in M' \cap V(C)} 1 \\ &> \sum_{i \in M' \cap V(C)} \sum_{j:(i,j)\in C} x_{ij} \\ &= \sum_{(i,j)\in C} x_{ij} \\ &= \sum_{j \in J' \cap V(C)} \sum_{i:(i,j)\in C} x_{ij} \\ &= \sum_{j \in J' \cap V(C)} 1 = \frac{|C|}{2}, \end{aligned}$$

a contradiction. Hence there must be some machine $i \in M' \cap Z$ such that $\deg(i) = 2$ and $\sum_{j:(i,j)\in E} x_{ij} \geq 1$. □

The following algorithm by Shmoys and Tardos uses this property.

G.A.P. Algorithm.

1. Set $F := \emptyset$, $\tilde{M} := M$, $T_i := T$ for all machines i.
2. Solve the LP to obtain a vertex solution $\mathbf{x}$.
3. (a) If $x_{ij} = 0$ for some (i,j), remove (i,j) from E.

(b) If $x_{ij} = 1$ for some (i, j), include (i, j) in F and remove (i, j) from E. Set $T_i := T_i - p_{ij}$ and remove j from J.

(c) If there is a machine i such that $\deg(i) \leq 1$ or $\deg(i) \leq 2$ and $\sum_j x_{ij} \geq 1$ then remove i from $\tilde{M}$ (effectively discarding a constraint).

4. If $E \neq \emptyset$, return to 2.

5. Output F.

We know that this algorithm terminates because in each execution of step 2, one of the cases applies (due to Lemma 4) and either $|E|$ or $|\tilde{M}|$ decreases. We now prove the theorem stated earlier about this algorithm — namely, that if there is a solution with cost C and makespan T then the algorithm finds a solution of cost at most C and makespan at most $2T$.

Proof: We prove that the algorithm finds a solution of cost at most C by showing that it maintains the invariant $\text{cost}(F)+\text{cost}(LP) \leq C$. When F is $\emptyset$ initially, this claim is clearly true, since solutions to general assignment are feasible LP solutions. When we assign job j to machine i (at cost c_{ij}), we also remove the corresponding edge with $x_{ij} = 1$, so the cost of the LP decreases equally by $c_{ij}x_{ij} = c_{ij}$. Otherwise, when we remove a machine constraint, we can only decrease the cost of the LP. Hence, at the end of the algorithm's run, $\text{cost}(F) \leq C$.

Now consider the processing time on machine i. While i is still in $\tilde{M}$ — i.e. as long as we have a constraint $\sum_j p_{ij}x_{ij} \leq T_i$ — the jobs already allocated take time $T - T_i$, so we maintain the invariant that

$$\sum_{(i,j)\in F\cap\delta(i)} p_{ij} + \sum_j p_{ij}x_{ij} \leq (T - T_i) + T_i = T$$

When we remove a constraint, either of the following occurs:

1. We are discarding $i \in \tilde{M}$ such that $\deg(i) \leq 1$. In this case, at most one more job j can be allocated, and we know that $p_{ij} \leq T$.

2. We are discarding $i \in \tilde{M}$ such that $\deg(i) = 2$ and $\sum_j x_{ij} \geq 1$. For clarity of notation, suppose the neighbors of i are jobs 1 and 2. Previously, these two jobs contributed $\sum_{j=1}^{2} p_{ij}x_{ij}$ to the processing time on machine i. Once we discard the time constraint associated with machine i, they can contribute at most $\sum_{j=1}^{2} p_{ij}$. Hence, after discarding the time constraint of i, these two jobs may be allocated for an additional time contribution to i of maximum possible value

$$\sum_{j=1}^{2} p_{ij} - \sum_{j=1}^{2} p_{ij}x_{ij} = \sum_{j=1}^{2} p_{ij}(1 - x_{ij}) \leq \sum_{j=1}^{2} T(1 - x_{ij}) = T\left(2 - \sum_{j=1}^{2} x_{ij}\right) \leq T$$

In either case, after dropping the time constraint associated with machine i, we may allocate jobs to i that together require no more than T time on top of the constraint that we removed. Hence, all in all, no machine is allocated a job schedule requiring more than $2T$ time. □