CSC2420 Fall 2012: Algorithm Design, Analysis and Theory
Lecture 5

Allan Borodin

February 11, 2016

Announcements and todays agenda

- Announcements
  1. Assignment 1 due today. I will go over any questions on the assignment.
  2. I hope to start Assignment 2 over reading week. In any case enjoy reading week.
- Todays agenda
  1. Begin linear programming and rounding
  2. LP Duality

Discussion of question 5 in problem set 1

- Proof sketch for question 5.
  We want to argue that that Graham's algorithm for makespan does not achieve $2 - c$ for any constant $c > 0$ in the ROM model. For a fixed k, take the instance with $m(m-k)$ '1' values and k 'm' values. The expected position for the last occurence of m is at the $k/(k+1)$ th fraction of the input. So the value obtained by the algorithm is about $\frac{k}{(k+1)} * (m-k) + m$ whereas the OPT is m. This gives a ratio which is more than $2 - c$ if k is sufficiently large and $m >> k$.

Discussion of question 6 on problem set

- Proof sketch for question 6.
  We want to proceed just like we did for the case of the identical machine model. First as before we will use binary search to hone in on the optimal value of the makespan. So now lets see how we determine whether or not the given set of jobs can be scheduled within makespan (say) T while using m_1 machines with speed 1 and m_2 with speed s_1. We let V_1 be the set of configurations that can complete on a machine with speed 1, and V_2 be the set of configurations that can complete on a machine with speed with speed s_1. Now we define our DP matrix $M[x_1, \ldots, x_d] = (\min t; t = \ell_1 + \ell_2$ and we can schedule x_i jobs of size z_i within makespan T using ℓ_1 machines with speed 1 and ℓ_2 machines with speed s_1. Now the recursive definition of M consists of choosing a machine type and all possible configuations that can be packed into the machine within makespan T.
  There are some details to be worked out but I claim that this idea can be made to provide the desired algorithm.

Integer Programming (IP) and Linear Programming (LP)

- We now introduce what is both theoretically and in practice one of the most general frameworks for solving search and optimization problems. Namely, we consider how many problems can be formulated as integer programs (IP). (Later, we will also consider other mathematical programming formulations.)
- Solving an IP is in general an NP hard problem although there are various IP problems that can be solved optimally. Moreover, in practice, many large instances of IP do get solved.
- Our initial emphasis will be on linear program (LP) relaxations of IPs. LPs can be solved optimally in polynomial time as first shown by Khachiyan's ellipsoid method [1979] and then Karmarkar's‘ [1984] more practical interior point method. In some (many?) cases, Danzig's [1947] simplex method will outperform (in terms of time) the worst case polynomial time methods.
- Smoothed analysis gives an explanation for the success of simplex.
- Open: a strongly polynomial time algorithm for solving LPs?

Some IP and LP concepts

Integer Programs

An IP has the following form:

- Maximize (minimize) $\sum_j c_j x_j$
- subject to $(\sum_j a_{ij} x_j) R_i b_i$ for $i = 1, \ldots, m$ and where R_i can be $=, \geq, \leq$
- x_j is an integer (or in some prescribed set of integers) for all j

Here we often assume that all parameters $\{a_{ij}, c_j, b_i\}$ are integers or rationals but in general they can be real valued.

An LP has the same form except now the last condition is realized by letting the x_j be real valued. It can be shown that if an LP has only rational parameters then we can assume that the $\{x_j\}$ will be rational.

Canonical LP forms

Without loss of generality, LPs can be formulated as follows:

Standard Form for an LP

- Maximize $\mathbf{c} \cdot \mathbf{x}$ — Minimize $\mathbf{c} \cdot \mathbf{x}$
- subject to $A \cdot \mathbf{x} \leq \mathbf{b}$ — $A \cdot \mathbf{x} \geq \mathbf{b}$
- $\mathbf{x} \geq 0$ — $\mathbf{x} \geq 0$

Slack form

- maximize/minimize $\mathbf{c} \cdot \mathbf{x}$
- subject to $A \cdot \mathbf{x} + \mathbf{s} = \mathbf{b}$
- $\mathbf{x} \geq 0; \mathbf{s} \geq 0$

The $\{s_j\}$ variables are called slack variables.

LP relaxation and rounding

- One standard way to use IP/LP formulations is to start with an IP representation of the problem and then relax the integer constraints on the x_j variables to be real (but again rational suffice) variables.
- We start with the well known simple example for the weighted vertex cover problem. Let the input be a graph $G = (V, E)$ with a weight function $w : V \rightarrow \Re^{\geq 0}$. To simplify notation let the vertices be $\{1, 2, \ldots . n\}$. Then we want to solve the following "natural IP representation" of the problem:
  - Minimize $\mathbf{w} \cdot \mathbf{x}$
  - subject to $x_i + x_j \geq 1$ for every edge $(i, j) \in E$
  - $x_j \in \{0, 1\}$ for all j.
- The intended meaning is that $x_j = 1$ iff vertex j is in the chosen cover. The constraint forces every edge to be covered by at least one vertex.
- Note that we could have equivalently said that the x_j just have to be non negative integers since it is clear that any optimal solution would not set any variable to have a value greater than 1.

LP rounding for the natural vertex cover IP

- The "natural LP relaxation" then is to replace $x_j \in \{0,1\}$ by $x_j \in [0,1]$ or more simply $x_j \geq 0$ for all j.
- It is clear that by allowing the variables to be arbitrary reals in [0,1], we are admitting more solutions than an IP optimal with variables in $\{0,1\}$. Hence the LP optimal has to be at least as good as any IP solution and usually it is better.
- The goal then is to convert an optimal LP solution into an IP solution in such a way that the IP solution is not much worse than the LP optimal (and hence not much worse than an IP optimum)
- Consider an LP optimum $\mathbf{x}^*$ and create an integral solution $\bar{\mathbf{x}}$ as follows: $\bar{x}_j = 1$ iff $x_j^* \geq 1/2$ and 0 otherwise. We need to show two things:
  1. $\bar{\mathbf{x}}$ is a valid solution to the IP (i.e. a valid vertex cover). Why?
  2. $\sum_j w_j \bar{x}_j \leq 2 \cdot \sum_j w_j x_j^* \leq 2 \cdot \mathit{IP\text{-}OPT}$; that is, the LP relaxation results in a 2-approximation.

The integrality gap

- Analogous to the locality gap (that we encountered in local search), for LP relaxations of an IP we can define the integrality gap (for a minimization problem) as $\max_{\mathcal{I}} \frac{IP-OPT}{LP-OPT}$; that is, we take the worst case ratio over all input instances $\mathcal{I}$ of the IP optimum to the LP optimum. (For maximization problems we take the inverse ratio.)
- Note that the integrality gap refers to a particular IP/LP relaxation of the problem just as the locality gap refers to a particular neighbourhood.
- The same concept of the integrality gap can be applied to other relaxations such as in semi definite programming (SDP).
- It should be clear that the simple IP/LP rounding we just used for the vertex cover problem shows that the integrality gap for the previously given IP/LP formulation is at most 2.
- By considering the complete graph K_n on n nodes, it is also easy to see that this integrality gap is at least $\frac{n-1}{n/2} = 2 - \frac{1}{n}$.

Integrality gaps and approximation ratios

- When one proves a positive (i.e upper) bound (say c) on the integrality gap for a particular IP/LP then usually this is a constructive result in that some proposed rounding establishes that the resulting integral solution is within a factor c of the LP optimum and hence this is a c-approximation algorithm.
- When one proves a negative (i.e. lower) bound (say c') on the integrality gap then this is only a result about the given IP/LP. In practice we tend to see an integrality gap as strong evidence that this particular formulation will not be able to result in a better than c' approximation. Indeed I know of no natural example where we have a lower bound on an integrality gap and yet nevertheless the IP/LP formulation leads "directly" into a better approximation ratio.
- In theory some conditions need to be established to make this into a provable statement. For the VC example, the rounding was independent (for each variable) and "oblivious" (to the input graph). In contrast to the K_n input, the LP-OPT and IP-OPT coincide for an even length cycle. Hence this intergrality gap represents a tight bound on the formulation using a graph oblivious rounding.

Makespan for the unrelated and restricted machine models: a more sophisticated rounding

In the VC example I use the terms "(input) independent rounding" and "oblivious" rounding.)

- We now return to the makespan problem with respect to the unrelated machines model and the special case of the restricted machine model.
- Recall the unrelated machines model where a job j is represented by a tuple $(p_{j,1}, \ldots, p_{j,m})$ where $p_{j,i}$ is the time that job j uses if scheduled on machine i.
- An important scheduling result is the Lenstra, Shmoys, Tardos (LST) [1990] IP/LP 2-approximation algorithm for the makespan problem in the unrelated machine model (when m is part of the input). They also obtain a PTAS for fixed m.

The natural IP and the LP relaxation

The IP/LP for unrelated machines makespan

- Minimize T
- Subject to
  1. $\sum_i x_{j,i} = 1$ for every job j % schedule every job
  2. $\sum_j x_{j,i} p_{j,i} \leq T$ for every machine i % do not exceed makespan
  3. $x_{j,i} \in \{0, 1\}$ % $x_{j,i} = 1$ iff job j scheduled on machine i

- The immdiate LP relaxation is to just have $x_{j,i} \geq 0$
- Even for identical machines (where $p_{j,i} = p_j$ for all i), the integrality gap IG is unbounded since the input could be just one large job with say size T leading to an LP-OPT of T/m and IP-OPT $=$ OPT $= T$ so that the IG $= m$.

Adapting the natural IP

- As in the PTAS for the identical machine makespan PTAS, we use binary search to find an appropriate approximation T for the optimal makespan.
- Given a candidate T, we remove all x_{ji} such that $p_{j,i} > T$ and obtain a "search problem" (i.e. constant or no objective function) for finding $x_{j,i}$ satisfying the IP constraints.
- Once we have found the optimal T for the search problem, the LST algorithm then shows how to use a non-independent rounding to obtain an integral solution yielding a 2-approximation.
- Note: We use the term "rounding" in a very general sense to mean any efficient way to convert the LP solution into an intergral solution.

Sketch of LST rounding for makespan problem

- Using slack form, LP theory can be used to show that if $\mathcal{L}$ is a feasible bounded LP with $m + n$ constraints (not counting the non-negativity constraints for the variables) then $\mathcal{L}$ has an optimal basic solution such that at most $n + m$ of the variables are non-zero.
- It follows how? that there are at most m of the n jobs that have fractional solutions (i.e. are not assigned to a single machine).
- Jobs assigned to a single machine do not need to be rounded; i.e. if $x_{j,i} = 1$ then schedule job j on machine i.
- Construct a bipartite graph between the $y \leq m$ fractionally assigned jobs and the m machines.

The rounding continued

- The goal is then to construct a matching of size y; that, is, the matching dictates how to schedule these fractionally assigned jobs. So it "only" remains to show that this bipartite graph has a matching of size y. Note, of course, this is what makes the "rounding" non-independent .
- The existence of this matching requires more LP whereby it can be shown (LST credit Dantzig [1963]) that the connected components of the bipartite graph are either trees or trees with one added edge (and therefore causing a unique cycle).
- The resulting schedule then has makespan at most $2T$ since each fractional job has $p_{j,i} \leq T$ and the LP has guaranteed a makespan at most T before assigning the fractional jobs.

The restricted machine makespan problem

- The restricted machines model is a special case of the unrelated machines problem where for every job j, $p_{j,i} \in \{p_j, \infty\}$. Hence the LST 2-approximation applies.
- LST show that it is NP hard to do better than a 1.5 approximation for the restricted machines (and hence unrelated machines) problem.
- Shmoys shows that for the special case that $p_j \in \{1, 2\}$ that the problem can be solved in polynomial time.
- There is a relatively new (somewhat strange) result due to Svensson [2011]. He shows how to approximate the value of the optimum makespan to within a factor of $33/17 \approx 1.9413 < 2$. This is proven constructively by a local search algorithm satisfying the approximation. However, the local search is not shown to terminate in polynomial time.
- Note that if we could determine the optimal makespan value in polynomial time, then we can also find an optimal solution in polynomial time. However, the same cannot be said when we are only approximating the makespan value.

The special case of graph orientation

- Consider the special case when there are (at most) two allowable machines for each job. This is called the graph orientation problem.
- It turns out easier to reason about the LP rounding applied to the graph orientation problem for the given IP/LP but still the integrality gap is 2.
- A more refined IP/LP by Eveblendr, Krcal and Sgall [2008] achieves a 1.75 approximation for the graph orientation problem.
- Even for the case when each job can only be scheduled on at most 3 machines, beating the 2-approximation remains an open problem.

Some concluding remarks (for now) about LP rounding

- We will return later to more LP applications. There are some nice notes by Allan Jepson providing some of the geometric concepts underlying LP solutions. (Note: these slides are password protected but I will provide password in class.) http://www.cs.toronto.edu/ jepson/csc373/index2012.html
- There can be, of course, many different IP/LP formulations for a given problem. In particular, one often adds additional constraints so that the polytope of the LP solutions is smaller.
- For example, in the vertex cover LP, one could simply add constraints $x_i + x_j + x_k \geq 2$ for every triangle in the graph and more generally, constraints for every odd length cycle. (These inequalities do not essentially change the integrality gap.)
- Adding such constraints corresponds to one round of what is called the LS lift and project method.
- There are a number of lift and project methods. If you are interested, then consult our local expert Toni Pitassi.

Duality: See Vazirani and Shmoys/Williamson texts, and Williamson article

- For a primal maximization (resp. minimization) LP in standard form, the dual LP is a minimization (resp. maximization) LP in standard form.
- Specifically, if the primal $\mathcal{P}$ is:
  - Minimize $\mathbf{c} \cdot \mathbf{x}$
  - subject to $A_{m \times n} \cdot \mathbf{x} \geq \mathbf{b}$
  - $\mathbf{x} \geq 0$
- then the dual LP $\mathcal{D}$ with dual variables $\mathbf{y}$ is:
  - Maximize $\mathbf{b} \cdot \mathbf{y}$
  - subject to $A^{tr}_{n \times m} \cdot \mathbf{y} \leq \mathbf{c}$
  - $\mathbf{y} \geq 0$
- Note that the dual (resp. primal) variables are in correspondence to primal (resp. dual) constraints.
- If we consider the dual $\mathcal{D}$ as the primal then its dual is the original primal $\mathcal{P}$. That is, the dual of the dual is the primal.

An example: set cover

As already noted, the vertex cover problem is a special case of the set cover problem in which the elements are the edges and the vertices are the sets, each set (ie vertex v) consisting of the edges adjacent to v.

The set cover problem as an IP/LP

minimize $\sum_j w_j x_j$
subject to $\sum_{j:e_i \in S_j} x_j \geq 1$ for all i; that is, $e_i \in U$
$x_j \in \{0, 1\}$ (resp. $x_j \geq 0$)

The dual LP

maximize $\sum_i y_i$
subject to $\sum_{i:e_i \in S_j} y_i \leq w_j$ for all j
$y_i \geq 0$

If all the parameters in a standard form minimization (resp. maximization) problem are non negative, then the problem is called a covering (resp. packing) problem. Note that the set cover problem is a covering problem and its dual is a packing problem.

Duality Theory Overview

- An essential aspect of duality is that a finite optimal value to either the primal or the dual determines an optimal value to both.
- The relation between these two can sometimes be easy to interpret. However, the interpretation of the dual may not always be intuitively meaningful.
- Still, duality is very useful because the duality principle states that optimization problems may be viewed from either of two perspectives and this might be useful as the solution of the dual might be much easier to calculate than the solution of the primal.
- In some cases, the dual might provide additional insight as to how to round the LP solution to an integral solution.
- Moreover, the relation between the primal $\mathcal{P}$ and the dual $\mathcal{D}$ will lead to primal-Dual algorithms and to the so-called dual fiiting analysis.
- In what follows we will assume the primal is a minimization problem to simplify the exposition.

Strong and Weak Duality

Strong Duality

If x^* and y^* are (finite) optimal primal and resp. dual solutions, then $\mathcal{D}(\mathbf{y}^*) = \mathcal{P}(\mathbf{x}^*)$.

Note: Before it was known that solving LPs was in polynomial time, it was observed that strong duality proves that LP (as a decision problem) is in $\mathbf{NP} \cap \mathbf{co} - \mathbf{NP}$ which strongly suggested that LP was not NP-complete.

Weak Duality for a Minimization Problem

If $\mathbf{x}$ and $\mathbf{y}$ are primal and resp. dual solutions, then $\mathcal{D}(\mathbf{y}) \leq \mathcal{P}(\mathbf{x})$.

- Duality can be motivated by asking how one can verify that the minimum in the primal is at least some value z. To get witnesses, one can explore non-negative scaling factors (i.e. the dual variables) that can be used as multipliers in the constraints. The multipliers, however, must not violate the objective (i.e cause any multiplies of a primal variable to exceed the coefficient in the objective) we are trying to bound.