CSC2420 Fall 2012: Algorithm Design, Analysis and Theory
Lecture 6

Allan Borodin

October 18, 2012

Finishing up last lecture: weighted interval scheduling

We have seen that :

1. For the unweighted interval scheduling problem on m machines, the "best fit" EFT greedy algorithm is optimal.
2. For $m = 1$, there is an optimal DP (that can be imnplemented as a priority backtracking (pBT) algorithm.
3. For arbitrary m, the one machine DP can be extended to an optimal algorithm running in time and space (n^m).
4. As previously noted, within the priority BT model, for any fixed m, any pBT algorithm requires width (i.e. space and hence time) $\Omega(n^m)$.
5. There is also a priority stack algorithm (using the EFT ordering) that optimally solves the $m = 1$ weighted problem.
6. For arbitrary m, the local ratio algorithm has approximation ratio $2 - \frac{1}{m}$ and it can be shown that no fixed order priority stack algorithm can be an optimum. (However, the inapproximation approaches 1 as m increases.)

Weighted interval scheduling continued

- Arkin and Silverberg [1987] reduce the m machine weighted interval problem to a min cost flow problem yielding an $O(n^2 \log n)$ time algorithm.
- It can also be seen that the weighted version of m machine interval scheduling problem is polynomial time since the problem can be expressed as an integer program (IP) which is totally unimodular.
- Yannakakis and Gavril [1987] show that the Maximum m colourable subgraph problem is NP hard for split graphs (which are chordal).
- While at first it may seem that the Arkin and Silverberg reduction only needs the perfect elimination property for interval graphs (which characterizes chordal graphs), the reduction is using the charcterization of interval graphs as those graphs whose nodes are contained in consecutive maximal cliques (i.e. where the maximal cliques are induced by the perfect ordering defined by the finishing times of the intervals).

Flow networks with costs

We now augment the definition of a flow network $\mathcal{F} = (G, s, t, c, \kappa)$ where $\kappa(e)$ is the non negative cost of edge e. Given a flow f, the cost of a path or cycle π is $\sum_{e \in \pi} \kappa(e) f(e)$.

MIn cost flow problem

Given a network $\mathcal{F}$ with costs, and given flow f in $\mathcal{F}$, the goal is to find a flow f of minimum cost. Sometimes we are only interested in a min cost max flow.

- Given a flow f, we can extend the definition of an augmenting path in $\mathcal{F}$ to an augmenting cycle which is just a simple cycle (not necessarily including the source) in the residual graph G_f.
- If there is a negative cost augmenting cycle, then the flow can be increased on each edge of this cycle which will not change the flow (by flow conservation) but will reduce the cost of the flow.
- A negative cost cycle in a directed graph can be detected by the Bellman Ford DP for the single source shortest path problem.

The metric labelling problem

We consider a problem well motivated by applications in, for example, information retrieval. (See Kleinberg and Tardos text)

The metric labelling problem

Given: graph $G = (V, E)$, a set of labels $L = \{a_1, \ldots, a_r\}$ in a metric space M with distance δ, and a cost function $\kappa : V \times L \to \Re^{\geq 0}$. The goal is to construct an assignment α of labels to the nodes V so as to minimize $\sum_{i \in V} \kappa(i, \alpha(i)) + \sum_{(i,j) \in E} p_{i,j} \cdot \delta(\alpha(i), \alpha(j))$

The idea is that κ represents a cost for labelling the node (e.g. a penalty for a bad classification of a web page), p represents the importance of that edge (e.g. where in a web page a particular link occurs) and δ represents the (basic or unweighted) cost of giving different labels to nodes that are related (e.g. the penalty for different labellings of web pages that are linking to each other or otherwise seem to be discussing similar topics.

The binary label case

- A case of special interest and the easiest to deal with is when the metric is the binary $\{0, 1\}$ metric; that is, $\delta(a, b) = 1$ if $a \neq b$ and 0 otherwise. (When there are only two labels, the binary $\{0, 1\}$ metric is the only metric.)
- The case of two labels suggests that the problem might be formulated as a min cut problem. Indeed this can be done to achieve an optimal algorithm when there are only two labels. For more than two labels, the binary metric case remains NP hard but a there is a 2-approximation via a local search algorithm that uses min cuts to search a local neighbourhood.

The case of two labels

- The problem for two labels can be restated as follows: find a partition $V = A \cup B$ of the nodes so as to minimize $\sum_{i \in A} b_i + \sum_{j \in B} a_j + \sum_{(i,j) \in A \times B} p_{i,j}$
- We transform this problem to a min cut problem as follows: construct the flow network $\mathcal{F} = (G', s, t, c)$ such that
  - $G' = (V', E')$
  - $V' = V \cup \{s, t\}$
  - $E' = \{(u, v) | u \neq v \in V\} \cup \{(s, u) | u \in V\} \cup \{(u, t) | u \in V\}$
  - $c(i, j) = c(j, i) = p_{i,j}$; $c(s, i) = a_i$; $c(i, t) = b_i$

Claim:

For any partition $V = A \cup B$, the capacity of the cut $c(A, B) = \sum_{i \in A} b_i + \sum_{j \in B} a_j + \sum_{(i,j) \in A \times B} p_{i,j}$.

-

Integer Programming (IP); 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 efficiently solved.
- Our initial emphasis will be on linear program (LP) relaxations of IPs. LPs can be solved optimally in polynomial time. This was shown by Khachiyan's ellipsoid method [1979] and then Karmarkar's' [1984] more practical interior point method. In many cases, implementations of Danzig's [1947] simplex method will outperform (in terms of time) the worst case polynomial time methods. Smoothed analyis gives an explanation for the success of simplex. It is an open problem if there is 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 contained 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 $A \cdot \mathbf{x} + \mathbf{b} = \mathbf{s}$
- $\mathbf{x} \geq 0; \mathbf{s} \geq 0$

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 may 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.
- 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.

LP rounding fo the natural vertex cover IP

- It is clear that by allowing the variables to be arbitrary reals in [0,1], we are admitting more solutions than an LP 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)
  2. $\sum_j w_j \bar{x}_j \leq 2 \cdot \sum_j w_j x_j^*$; 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 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 there are a some conditions that need to be established to make this into a provable statement. For the VC example, we observe that the rounding is 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 does represent a tight bound on the formulation.

Makespan for the unrelated and restricted machine models: non-independent 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, LST 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 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 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 non-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 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 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 problem.
- Shmoys shows that for the special case that $p_j \in \{1, 2\}$ that the problem can be solved in polynomial time. (Bonus problem.)
- 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 hopefully return to LP/LP rounding later. There are some nice notes by Allan Jepson provide some of the geometric concepts underlying LP solutions. (Note: these slides are password protected but I will provide password in class.)
- 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, 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.
- It turns out that this does not essentially change the integrality gap.
- Adding such constraints corresponds to one round of what is called the LS left and project method.
- There are a number of lift and project methods. If you are interested, then consult our local expert Toni Pitassi. You may also want to look at Siavosh Bennabas' very recent thesis.