CSC2420 Spring 2016: Lecture 2

Allan Borodin

January 21,2016

Announcements and todays agenda

- First part of assignment 1 was posted last weekend. I plan to assign more questions as we discuss additional topics. Please try to work on the questions week by week and not postpone until the due date. I will set due date for assignment 1 after I assign more questions.
- I try to post the slides within a day or so of the lecture and usually post what was discussed. Some times I will post all the intended slides for context.
- Todays agenda
  1. Review and continue discussion of the set packing problem.
  2. Sketch s-set packing greedy algorithm analyis
  3. Abstraction of s-set packing to $(s+1)$-claw free graphs.
  4. State $O(\sqrt{m})$-approximationi for set packing.
  5. Priority algorithms with revocable acceptances (for packing problems). The “greedy” algorithm for weighted interval scheduling (WIS) and weighted job interval scheduling problem (WJIS).
  6. Abstraction of WIS (resp. WJIS) to max independent set in chordal graphs and (respectively) inductive 2-independent graphs.
  7. Priority stack algorithms.
  8. The random order model. (ROM)

Greedy algorithms for the set packing problem

The set packing problem

We are given n subsets $S_1, \ldots, S_n$ from a universe U of size m. In the weighted case, each subset S_i has a weight w_i. The goal is to choose a disjoint subcollection $\mathcal{S}$ of the subsets so as to maximize $\sum_{S_i \in \mathcal{S}} w_i$. In the s-set packing problem we have $|S_i| \leq s$ for all i.

- This is a well studied problem and by reduction from the max clique problem, there is an $m^{\frac{1}{2}-\epsilon}$ hardness of approximation assuming $NP \neq ZPP$. For s-set packing, there is an $\Omega(s/\log s)$ hardness of approximation assuming $P \neq NP$.
- Set packing is the underlying allocation problem in what are called combinatorial auctions as studied in mechanism design.
- We will consider two "natural" greedy algorithms for the s-set packing problem and a somewhat less obvious greedy algorithm for the set packing problem. These greedy algorithms are all fixed order priority algorithms.

The first natural greedy algorithm for set packing

Greedy-by-weight (*$Greedy_{wt}$*)

```
Sort the sets so that $w_1 \geq w_2 \ldots \geq w_n$.
$\mathcal{S} := \varnothing$
For $i : 1 \ldots n$
    If $S_I$ does not intersect any set in $\mathcal{S}$ then
        $\mathcal{S} := \mathcal{S} \cup S_i$.
End For
```

- In the unweighted case (i.e. $\forall i, w_i = 1$), this is an online algorithm.
- In the weighted (and hence also unweighted) case, greedy-by-weight provides an s-approximation for the s-set packing problem.
- The approximation bound can be shown by a charging argument.

Two types of approximation arguments

- Recall the argument for makespan on identical machines.
  1. We identify some intrinsic limiting bounds for any solution including an OPT solution; in this case average load/machine and processing time for any job.
  2. Then we relate the algorithmic solution (in this case the natural greedy solution) to those bounding factors.
  3. We will see something similar when consider "LP rounding".
- We now consider a different type of argument. Namely a charging argument.
- We will consider this in the context of a maximization problem, namely the charging argument for $Greedy_{wt}$ for s-set packing.
  1. We will charge the weight of every set in an OPT solution to the first set in the greedy solution with which it intersects.
  2. How many sets in OPT can be charged to the same set in $Greedy_{wt}$?
  3. If say set $S_i \in OPT$ is being charged to $S_j \in Greedy_{wt}$, then we know $w_i \leq w_j$.

The second natural greedy algorithm for set packing

Greedy-by-weight-per-size

```
Sort the sets so that $w_1/|S_1| \geq w_2/|S_2| \ldots \geq w_n/|S_n|$.
$\mathcal{S} := \varnothing$
For $i : 1 \ldots n$
    If $S_I$ does not intersect any set in $\mathcal{S}$ then
        $\mathcal{S} := \mathcal{S} \cup S_i$.
End For
```

- In the weighted case, greedy-by-weight provides an s-approximation for the s-set packing problem.
- For both greedy algorithms, the approximation ratio is tight; that is, there are examples where this is essentially the approximation. In particular, greedy-by-weight-per-size is only an m-approximation where $m = |U|$.
- We usually assume $n >> m$ and note that by just selecting the set of largest weight, we obtain an $\min\{n, m\}$-approximation.

Improving the approximation for greedy set packing

- In the unweighted case, greedy-by-weight-per-size can be restated as sorting so that $|S_1| \leq |S_2| \ldots \leq |S_n|$ and it can be shown to provide an $\sqrt{m}$-approximation for set packing.
- On the other hand, greedy-by-weight-per-size does not improve the approximation for weighted set packing.

Greedy-by-weight-per-squareroot-size

```
Sort the sets so that $w_1/\sqrt{|S_1|} \geq w_2/\sqrt{|S_2|} \ldots \geq w_n/\sqrt{|S_n|}$.
$\mathcal{S} := \varnothing$
For $i : 1 \ldots n$
    If $S_I$ does not intersect any set in $\mathcal{S}$ then
        $\mathcal{S} := \mathcal{S} \cup S_i$.
End For
```

Theorem: Greedy-by-weight-per-squareroot-size provides a $2\sqrt{m}$-approximation for the set packing problem. And as noted earlier, this is essentially the best possible approximation assuming $NP \neq ZPP$.

Another way to obtain an $O(\sqrt{m})$ approximation

There is another way to obtain the same aysmptototic improvement for the weighted set packing problem. Namely, we can use the idea of partial enumeration greedy; that is somehow combining some kind of brute force (or naive) approach with a greedy algorithm.

Partial Enumeration with Greedy-by-weight ($PGreedy_k$)

Let Max_k be the best solution possible when restricting solutions to those containing at most k sets. Let G be the solution obtained by $Greedy_{wt}$ applied to sets of cardianlity at most $\sqrt{m/k}$. Set $PGreedy_k$ to be the best of Max_k and G.

- Theorem: $PGreedy_k$ achieves a $2\sqrt{m/k}$-approximation for the weighted set packing problem (on a universe of size m)
- In particular, for $k = 1$, we obtain a $2\sqrt{m}$ approximation and this can be improved by an arbitrary constant factor $\sqrt{k}$ at the cost of the brute force search for the best solution of cardinality k; that is, at the cost of say n^k.

$(k+1)$-claw free graphs

A graph $G = (V, E)$ is $(k+1)$-claw free if for all $v \in V$, the induced subgraph of $Nbhd$(v) has at most k independent vertices (i.e. does not have a $k+1$ claw as an induced subgraph).

$(k+1)$-claw free graphs abstract a number of interesting applications.

- In particular, we are interested in the (weighted) maximum independent set problem (W)MIS for $(k+1)$-claw free graphs. Note that it is hard to approximate the MIS for an arbiitrary n node graph to within a factor $n^{1-\epsilon}$ for any $\epsilon > 0$.
- We can (greedily) k-approximate WMIS for $(k+1)$-claw free graphs.
- The (weighted) k-set packing problem is an instance of (W)MIS on $k+1$-claw free graphs. What algorithms generalize?
- There are many types of graphs that are $k+1$ claw free for small k; in particular, the intersection graph of translates of a convex object in the two dimensional plane is a 6-claw free graph. For rectangles, the intersection graph is 5-claw free.

Extensions of the priority model: priority with revocable acceptances

- For packing problems, we can have priority algorithms with revocable acceptances. That is, in each iteration the algorithm can now reject previously accepted items in order to accept the current item. However, at all times, the set of currently accepted items must be a feasible set and all rejections are permanent.
- Within this model, there is a 4-approximation algorithm for the weighted interval selection problem WISP (Bar-Noy et al [2001], and Erlebach and Spieksma [2003]), and a ≈ 1.17 inapproximation bound (Horn [2004]). More generally, the algorithm applies to the weighted job interval selection problem WJISP resulting in an 8-approximation.
- The model has also been studied with respect to the proportional profit knapsack problem/subset sum problem (Ye and B [2008]) improving the constant approximation. And for the general knapsack problem, the model allows a 2-approximation.

The *Greedy*$_\alpha$ algorithm for WJISP

The algorithm as stated by Erlebach and Spieksma (and called ADMISSION by Bar Noy et al) is as follows:

```
$S$ := $\emptyset$            % $S$ is the set of currently accepted intervals
Sort input intervals so that $f_1 \leq f_2 \ldots \leq f_n$
for $i = 1..n$
    $C_i$ := min weight subset of $S$ s.t. $(S/C_i) \cup \{I_i\}$ feasible
    if $v(C_i) \leq \alpha \cdot v(I_i)$ then
        $S$ := $(S/C_i) \cup \{I_i\}$
    end if
END FOR
```

Figure : Priority algorithm with revocable acceptances for WJISP

The *Greedy*$_\alpha$ algorithm (which is not greedy by my definition) has a tight approximation ratio of $\frac{1}{\alpha(1-\alpha)}$ for WISP and $\frac{2}{\alpha(1-\alpha)}$ for WJISP.

Priority Stack Algorithms

- For packing problems, instead of immediate permanent acceptances, in the first phase of a priority stack algorithm, items (that have not been immediately rejected) can be placed on a stack. After all items have been considered (in the first phase), a second phase consists of popping the stack so as to insure feasibility. That is, while popping the stack, the item becomes permanently accepted if it can be feasibly added to the current set of permanently accepted items; otherwise it is rejected. Within this priority stack model (which models a class of primal dual with reverse delete algorithms and a class of local ratio algorithms), the weighted interval selection problem can be computed optimally.
- For covering problems (such as min weight set cover and min weight Steiner tree), the popping stage is insure the minimality of the solution; that is, while popping item *I* from the stack, if the current set of permanently accepted items plus the items still on the stack already consitute a solution then *I* is deleted and otherwise it becomes a permanently accepted item.

Chordal graphs and perfect elimination orderings

An interval graph is an example of a chordal graph. There are a number of equivalent definitions for chordal graphs, the standard one being that there are no induced cycles of length greater than 3.

We shall use the characterization that a graph $G = (V, E)$ is chordal iff there is an ordering of the vertices $v_1, \ldots, v_n$ such that for all i, $Nbdh(v_i) \cap \{v_{i+1}, \ldots, v_n\}$ is a clique. Such an ordering is called a perfect elimination ordering (PEO).

It is easy to see that the interval graph induced by interval intersection has a PEO (and hence is chordal) by ordering the intervals such that $f_1 \leq f_2 \ldots \leq f_n$. Using this ordering we know that there is a greedy (i.e. priority) algorithm that optimally selects a maximum size set of non intersecting intervals. The same algorithm (and proof by charging argument) using a PEO for any chordal graph optimally solves the unweighted MIS problem. The following priority stack algorithm provides an optimal solution for the WMIS problem on chordal graphs.

The optimal priority stack algorithm for the weighted max independent set problem (WMIS) in chordal graphs; Akcoglu et al [2002]

```
$Stack$ := $\emptyset$          % $Stack$ is the set of items on stack
Sort input intervals so that $f_1 \leq f_2 \ldots \leq f_n$
**For** $i = 1..n$
    $C_i$ := nodes on stack that are adjacent to $v_i$
    **If** $w(v_i) > w(C_i)$ **then** push $v_i$ onto stack, **else** reject
**End For**
$S$ := $\emptyset$              % $S$ will be the set of accepted nodes
**While** $Stack \neq \emptyset$
    Pop next node $v$ from $Stack$
    **If** $v$ is not adjacent to any node in $S$, **then** $S$ :=$S \cup \{v\}$
**End While**
```

Figure : Priority stack algorithm for chordal WMIS

A k-PEO and inductive k-independent graphs

- An alternative way to describe a PEO is to say that $Nbhd(v_i) \cap v_{i+1}, \ldots, v_n\}$ has independence number 1.
- We can generalize this to a k-PEO by saying that $Nbhd(v_i) \cap v_{i+1}, \ldots, v_n\}$ has independence number at most k.
- We will say that a graph is an inductive k-independent graph if it has a k-PEO.
- Inductive k-independent graphs generalize both chordal graphs and $k+1$-claw free graphs. They also obviously generalize *inductive degree k* which contains all "treewidth k" graphs.
- The intersection graph induced by the JISP problem is an inductive 2-independent graph.
- Using a k-PEO, a fixed-order priority algorithm (resp. a priority stack algorithm) is a k-approximation algorithm for MIS (resp. for WMIS) wrt inductive k-independent graphs.

More extensions of the priority model

- So far we have been implicitly assuming deterministic priority algorithms. We can allow the ordering and/or the decisions to be randomized.
- A special case of fixed priority with randomized orderings is when the input set is ordered randomly without any dependence on the set of inputs. In the online setting this is called the random order model.
- The revocable acceptances model is an example of priority algorithms that allow reassignments (of previous decisions) to some extent or at some cost.
- The partial enumeration greedy is an example of taking the best of some small set of adaptive priority algorithms.
- Priority stack algorithms are an example of 2-pass (or multi-pass) priority algorithms where in each pass we apply a priority algorithm. Of course, it has to be well specified as to what information can be made available to the next pass.

The random order model (ROM)

Motivating the random order model

The random order model provides a nice compromise between the often unrealistic negative results for worst case (even randomized) online algorithms and the often unrealistic positive setting of inputs being generated by simple distributions.

- In many online scenarios, we do not have realistic assumptions as to the distributional nature of inputs (so we default to worst case analysis). But in many applications we can believe that inputs do arrive randomly or more precisely uniformly at random.
- The ROM can be (at least) traced back to what is called the (classical) secretary (aka marriage or dowry) problem, popularized in a Martin Gardner Scientific American article.
- As Fiat et al (SODA 2015) note, perhaps Johannes Kepler (1571-1630) used some secretary algorithm when interviewing 11 potential brides over two years.

The secretary problem

The classical problem (which has now been extended and studied in many different variations is as follows:

- The classic problem (as in the Gardiner article) assumes an adversarially chosen set of distinct values for (say N) items that arrive in random order (e.g. candidates for a position, offers for a car, etc.). N is assumed to be known.
- Once an item (e.g. secretary) is chosen, that decision is irrevocable. Hence, this boils down to finding an *optimal stopping rule*, a subject that can be considered part of stochastic optimization.
- The goal is to select one item so as to maximize the probability that the item chosen is the one of maximum value.
- For any set of N values, maximizing the probability of choosing the best item immediately yields a bound for the expected (over the random orderings) value of the chosen item. For an "ordinal algorithm", these two measures are essentially the same. Why?

The secretary problem continued

- It is not difficult to show that any deterministic or randomized (adversarial order) online algorithm has competitive ratio [1] at most $O(\frac{1}{N})$. Hence the need to consider the ROM model to obtain more interesting (and hopefully more meaningful) results.
- We note (and this holds more generally) that "positive results" for the ROM model subsume the stochastic optimization scenario where inputs are generated by an unknown (and hence known) i.i.d. process. Why?
- There are many variations and extensions of the secretary problem some of which we will consider later (or at least mention).
- In general, any online problem can be studied with respect to the ROM model.

[1]Recall that for maximization problems, competitive and approximation ratios can sometimes presented as fractions $\alpha = \frac{ALG}{OPT} \leq 1$ and sometimes as ratios $c = \frac{OPT}{ALG} \geq 1$. I will try to follow the convention mainly used in each application.

The optimal stopping rule for the classical secretary problem

The amusing history of the secretary problem and the following result is taken up by Ferguson in a 1989 article.

Theorem: For N and r, there is an exact formula for the probability of selecting the maximum value item after observing the first r items, and then selecting the first item (if any) that exceeds the value of the items seen thus far. In the limit as $N \to \infty$, the optimal stopping rule is to observe (i.e. not take) the first $r = N/e$ items. The probability of obtaining the best item is then $1/e$ and hence the expected value of the item chosen is at least $\frac{1}{e} v_{max}$.

Variations and extensions of the secretary problem

- Instead of maximizing the probability of choosing the best item, we can maximize the expected rank of the chosen item.
- Perhaps the most immediate extension is to be choosing k elements.
- This has been generalized to the matroid secretary problem by Babaioff. For arbitrary matroids, the approximation ratio remains an open problem.
- Another natural extension is to generalize the selection of one item to the online (and ROM) edge weighted bipartite matching problem, where say $N = |L|$ items arrive online to be matched with items in R. In online matching the goal is usually to maximize the size (for the unweighted case) or weight of a maximum matching.
- I will next to discuss online matching and then later (hopefully) the extension to the adwords problem where the online nodes L represent advertisers/bidders with budgets and preferences/values for the R nodes representing keywords/queries.

The unweighted bipartite matching problem

Before leaving (for now) online, ROM and priority algorithms, I want to briefly discuss one more (surprising) ROM algorithm (equivalently for this algorithm, a randomized online algorithm) that has generated a good deal of recent research.

- Let $G = (U, V, E)$ be an unweighted bipartite graph with edges $E \subset U \times V$. Lets say that the vertices in U are the online vertices that arrive one at a time $u_1, \ldots u_n$, revealing the offline nodes in V to which they are adjacent.
- The online algorithm must irrecvocably decide whether and how to match u_i to an unmatched $v \in V$ (if there is such a node).
- It is easy to see that any greedy algorithm (i.e. one that matches each u_i if possible) produces a maximal matching and hence is a $\frac{1}{2}$-approximation (following the convention here for using fractional approximation ratios). This is also a tight bound for any deterministic online algorithm as can be seen by a simple 2×2 bipartite graph.

The Karp, Vazirani, Varizani (KVV) algorithm

- The KVV Ranking algorithm chooses a random permutation of the nodes in V and then when a node $u \in U$ appears, it matches u to the highest ranked unmatched $v \in V$ such that (u, v) is an edge (if such a v exists).
- Aside: making a random choice for each u is still only a $\frac{1}{2}$ approx.
- The analysis of this algorithm can be used to show that there is a deterministic greedy algorithm in the ROM model.
- That is, let $\{v_1, \ldots, v_n\}$ be any fixed ordering of the vertices and let the nodes in U enter randomly, then match each u to the first unmatched $v \in V$ according to the fixed order.
- To argue this, consider fixed orderings of U and V; the claim is that the matching will be the same whether U or V is entering online.

The KVV result and recent progress

KVV Theorem

Ranking provides a $(1 - 1/e) \approx .63$ approximation.

- Original analysis is not rigorous.
- There is an alternative proof (and extension) by Goel and Mehta [2008], and other proofs (e.g. in Birnbaum and Mathieu [2008]).
- KVV show that the $(1 - 1/e)$ bound is essentially tight for any randomized online (i.e. adversarial input) algorithm. In the ROM model, Goel and Mehta state inapproximation bounds of $\frac{3}{4}$ (for deterministic) and $\frac{5}{6}$ (for randomized) algorithms.
- In the ROM model, Karande, Mehta, Tripathi [2011] show that Ranking achieves approximation at least .653 (beating $1 - 1/e$) and no better than .727.