CSC2420: Algorithm Design, Analysis and Theory Spring 2019

Allan Borodin

January 24, 2019

Week 3

Todays agenda

- Set packing and another way to obtain a $O(\sqrt{m})$ approximation.
- ($k+1$-claw free graphs as a generalization of s-packing.
- Greedy algorithms for weighted set cover and vertex cover.
- The (Weighted) Job Intreval Scheduling Problem.
- Priority algorithms with revocable acceptances.
- Chordal graphs, perfect elimination ordering (PEO), and priority stack algorithms.
- Interval colouring
- Weighted interval scheduling on m machines
- k-PEOs and priority and priority stack algorithms

Next week, more on online algorithms and, in particular, online learning algorithms.

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 cardinality 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: generalizing s-set packing

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) s-set packing problem is an instance of (W)MIS on $s+1$-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.

Vertex cover: where (again) the "natural greedy" is not best

- We consider another example (weighted vertex cover) where the "natural greedy algorithm" does not yield a good approximation.
- The vertex cover problem: Given node weighted graph $G = (V, E)$, with node weights $w(v), v \in V$.
  Goal: Find a subset $V' \subset V$ that covers the edges (i.e. $\forall e = (u, v) \in E$, either u or v is in V') so as to mininize $\sum_{v \in V'} w(v)$.
- Even for unweighted graphs, the problem is known to be NP-hard to obtain a 1.3606 approximation and under another (not so universally believed) conjecture (UGC) one cannot obtain a $2 - \epsilon$ approximation.
- For the unweighted problem, there are simple 2-approximation greedy algorithms such as just taking V' to be any maximal matching.
- The set cover problem is as follows: Given a weighted collection of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ over an n element universe U with set weights $w(S_i)$.
  Goal: Find a subcollection $\mathcal{S}'$ that covers the universe so as to minimize $\sum_{S_i \in \mathcal{S}'} w(S_i)$.

The natural greedy algorithm for weighted set cover

"The natural" greedy algorithm for set cover

$\mathcal{S}' = \varnothing$

While there are uncovered elements in the universe U

Let $j = argmin_i\{w(S_i)/|S_i \cap U|$

$\mathcal{S}' = \mathcal{S}' \cup \{S_i\}$

$U = U \setminus \{S_i\}$

End While

- The set cover problem is one of the first NP-complete problems.
- Johnson[1974] and Lovasz[1975] independently showed that this natural greedy algorithm provides a $H(m) \approx \ln m$ approximation for the unweighted case where $m = \max_i |S_i| \leq |U|$. This was extended by Chvatal[1979] to the weighted case.
- Under a reasonable complexity assumption, Feige[1979] showed that it was not possile to acheive a $(1 - \epsilon) \ln n$ approximation even for the unweighted case.

The natural greedy algorithm for weighted vertex cover (WVC)

Vertex cover can be viewed as a special case of set cover. (How?)

The natural greedy algorithm for weighted vertex cover (WVC)

Vertex cover can be viewed as a special case of set cover. (How?) Then the natural greedy set cover algorithm (which is essentially optimal for set coveri up to standard comlexity assumptions) becomes the following:

$d'(v) := d(v)$ for all $v \in V$

% $d'(v)$ will be the residual degree of a node

While there are uncovered edges

Let v be the node minimizing $w(v)/d'(v)$

Add v to the vertex cover;

remove all edges in $Nbhd(v)$;

recalculate the residual degree of all nodes in $Nbhd)v)$

End While

Figure: Natural greedy algorithm for weighted vertex cover. Approximation ratio $H_n \approx \ln n$ where $n = |V|$.

Clarkson's [1983] modified greedy for WVC

$d'(v) := d(v)$ for all $v \in V$

% $d'(v)$ will be the residual degree of a node

$w'(v) := w(v)$ for all $v \in V$

% $w'(v)$ will be the residual weight of a node

While there are uncovered edges

Let v be the node minimizing $w'(v)/d'(v)$

$w := w'(v)/d'(v)$

$w'(u) := w'(u) - w$ for all $u \in Nbhd(v)$

% For analysis only, set $we(u, v) = w$

Add v to the vertex cover;

remove all edges in $Nbhd(v)$;

recalculate the residual degree of all nodes in $Nbhd(v)$

End While

Figure: Clarkson's greedy algorithm for weighted vertex cover. Approximation ratio 2. Invariant: $w(v) = w'(v) + sum_{e \in E} we(e)$

Extending problems and extending the priority paradigm

Now that we know that for arbitrary profits priority algorithms cannot achieve a constant approximation for the weighted interval selection problem (WISP), lets return to ways to extend the model and the problem.

The interval selection problem is a packng problem so that any subset of a feasible solution is a feasible solution.

We already mentioned *priority algorithms with revocable acceptances* which can be used for any packing problem and in particular yields a constant approximation of WISP and also to the following NP-hard generalization of the WISP problem.

The (weighted) job interval selection problem WJISP

A *job* is a set of intervals. In addition to the start and finishing times of each interval, we will say that intervals belong to exactly one job. A feasible set of intervals are non-intersecting (as in WISP) and there is at most one interval per job.

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:

$S := \varnothing$ % 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

$Stack := \varnothing$ % $Stack$ is the set of items on stack
Sort nodes as in a PEO.
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
$S := \varnothing$ % S will be the set of accepted nodes
While $Stack \neq \varnothing$
    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

Sketch of the WMIS chordal graph result

Let ALG (resp. OPT) denote the nodes in the solution of the algorithm (resp. of an optimal solution). Let S be the contents of the stack at the end of the push phase and let S_i be the contents of the stack (in the push phase) as we are about to consider v_i.

Define $\tilde{w}(v_i) = w(v_i) - \sum_{v_j \in S_i \cap Nbhd(v_i)} \tilde{w}(v_j)$. Then we push v_i on the stack iff $\tilde{w}(v_i) > 0$.

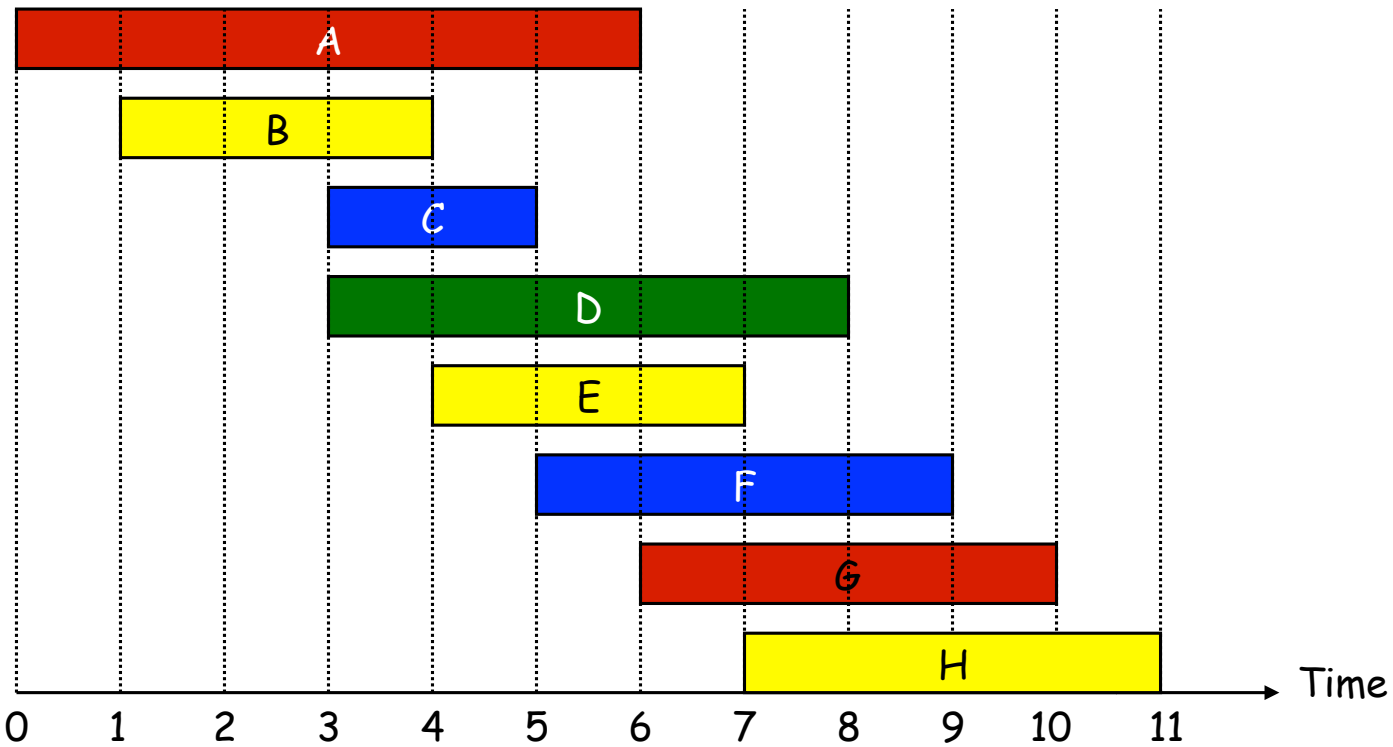
Fact: $\sum_{v_t \in ALG} w(v_t) = \sum_{v_t \in S} \tilde{w}(v_t)$

Then using the fact that the nodes were considered in the PEO ordering, we can show $\sum_{v_t \in OPT} w(v_t) = \sum_{v_t \in S} \tilde{w}_t$

Interval colouring

Interval Colouring Problem

- Given a set of intervals, colour all intervals so that intervals having the same colour do not intersect
- **Goal:** minimize the number of colours used.



We use 4 colors in this example. **Question:** Is this optimal?

Interval colouring

Interval Colouring Problem

- Given a set of intervals, colour all intervals so that intervals having the same colour do not intersect
- **Goal:** minimize the number of colours used.

- We could simply apply the m-machine ISP for increasing m until we found the smallest m that is sufficient. But this would not be as efficient as the greedy algorithm to follow.

Greedy interval colouring algorithm

- Consider the EST (earliest starting time) for interval colouring.
  - Sort the intervals by non decreasing starting times
  - Assign each interval the smallest numbered colour that is feasible given the intervals already coloured.
- Recall that EST is a terrible algorithm for ISP.
- **Note:** this algorithm is equivalent to LFT (latest finishing time first).

Theorem

EST is optimal for interval colouring

Proof idea: When does the algorithm use a new colour? In any graph, the colouring number is at least as large as the maximum clique size (and equal for interval and more generally perfect graphs).

Interval colouring continued

Greedy Interval Colouring

Sort intervals so that $s_1 \leq s_2 \leq \ldots \leq s_n$
FOR $i = 1$ to n
  Let $k := \min\{\ell : \ell \neq \chi(j)$ for all $j < i$ such that the
    j^{th} interval intersects the i^{th} interval$\}$
$\sigma(i) := k$
    % The i^{th} interval is greedily coloured by the smallest non conflicting colour.
ENDFOR

How does this generalize to a greedy algorithm for vertex coloring chordal graphs?

The m machine weighted interval scheduling problem and its graph theoretic interpretation

As a graph problem, m machine weighted interval scheduling becomes the maximum vertex m-colourable problem for interval graphs. Do results for m machine (weighted or unweighted) interval scheduling carry over to the maximum vertex m-colourable problem for chordal graphs?

Borodin, Cashman and Magen [2011] show that fixed order priority stack algorithms cannot optimally solve the m machine weighted interval scheduling problem. The one-machine stack algorithm extends to yield a $2 - \frac{1}{m}$ approximation which Bar-Noy et al first obtained by applying the one machine algorithm, "one machine at a time".

However, for any fixed m, dynamic programming can optimally solve the m machine weighted interval scheduling problem in polynomial time $O(n^m)$.

There is a min cost, max flow algorithm that can solve m (wlg. $m \leq n$) machine weighted interval scheduling in time $O(n^2 \log n)$.

m machine interval scheduling continued

We previously mentioned Regev's $\frac{\log m}{\log \log m}$ fixed order priority inapproximation for the restricted machines makespan problem. It is not currently known how to obtain an analogous inapproximation for adaptive priority algorithms.

Similarly, we can derive a (weak) fixed order priority stack inapproximation for the m machine weighted interval scheduling problem but do not know how to obtain an inapproximation for the adaptive model.

Curiously, thus far the best inapproximation we have is for 2 machines (namley, $\frac{6}{\sqrt{30}}$) ≈ 1.095 which can be extended to $\frac{m}{m-1}$ for m machines (i.e. the bound gets weaker) in contrast to the $2 - \frac{1}{m}$ algorithm where the approximation gets weaker.

And now to answer a question we previously raised, Yannakakis and Gavril [1987] show how to solve the chordal graph maximum m vertex colourable problem in polynomal time for any fixed m but show that it is NP-hard when m is a parameter of the problem. This shows that we cannot expect to reduce min vertex colouring to the max m colourable problem.

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 clearly generalize both chordal graphs and $k+1$-claw free graphs.

What other graphs are inductive k-independent for small k?

- 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.
- Using the reverse of a k-PEO, there is a k-approximation priority algorithm for coloring an inductive k indepdnent graph. .

Other examples of inductive k independent graphs

- The intersection graph induced by the JISP problem is an inductive 2-independent graph.
- The intersection graph induced by axis parallel rectangles in the plane are inductive 2-independent. The intersection graph of unit (respectively arbitrary) disks are inductive 3 (resp. 5) independent graphs.
- The intersection graphs of translates of a convex object in the plane (resp. d-dimensional) are inductive 3-(resp. $2d-1$)) independent.
- Planar graphs are inductive 3-independent.
- For a hereditary property of a graph, there analogies of inductive k-indpendent such as inductive max degree k which includes *tree-width* k graphs.

Note: For many specific grpahs (eg unit disk graphs, etc), much better approximation algorithms are known then by the use of priority and priority stack algorithms. But it is good to know that there is a simple "off-the shelf" solution that can always be used.