CSC2420 - Fall 2010 - Lecture 6

Lecturer: Prof. Allan Borodin
Scribe: Frank Vanderzwet

20-October-2010

1 Last Time and k+1 Claw Free Graphs

Last time we discussed greedy algorithms relating to weighted MISproblem. We also noted that k-set packing $\Rightarrow$ can be viewed as the (weighted) MIS problem on a $(k+1)$ clawfree graph. Recall, for $k+1$ clawfree graphs we have that for each node v, N(v) has at most k indep vertices (N(v) = neighborhood of v). We presented a natural ("the standard") greedy algorithm, sort so that $w_1 \geq w_2 \geq \dots\ w_n$ and accept greedily. This is a k-approximation of weighted MIS with respect to graphs which are k+1 clawfree. In the future "clawfree graphs" without specification means 3 clawfree.

A large class of k-clawfree graphs for small k can be obtained by considering the intersection graph obtained from translating geometric objects such as circles and squares.

The intersection graph of unit intervals defines a 3-claw free graph. This is because intersections can only happen at two places, the beginning of the interval and at the end. Since these are 3-clawfree we can achieve a 2-approx by a greedy algorithm for WMIS of such graphs. (In fact, the WMIS problem can be solved optimally for clawfree graphs.)

Definition : Proper interval graph are ones in which no interval is completely included in another.

The unit interval graph is a proper interval graph. Although not obvious the class of proper interval graphs is equal to the class of unit interval intersection graphs. However, arbitrary interval graphs are not r-clawfree for any fixed r. This follows since for arbitrary interval graphs we can have any number of subintervals included within a larger one.

Question : How well can greedy algorithms solve (weighted) interval selection problem, that is, (weighted) MIS on interval graphs? We will see that the unweighted MIS problem can be solved optimally for interval graphs but we will also present a model for greedy algorithms and show that within this model, the weighted MIS problem cannot be solved within any approximation ratio by a greedy algorithm.

2 Priority Algorithms ("Myoptic Algorithms")

2.1 Model for Greedy Algorithms

We must formalize what a greedy algorithm is if we wish to seriously study them. The following are models for greedy (greedy-like) algorithms which we call priority algorithms. So far all greedy algorithms that we have seen have been priority algorithms. We present the following two types of priority algorithms. We use the term "local ordering" which we will define later. We use $\mathcal{I}$ to represent the set of actual input items.

Fixed Priority

```
1. order $\mathcal{I}$ using a "local ordering"
2. for j = 1 ... n
3.     make an irrevocable decision $d_j$ for $I_j$
4. end for
```

Adaptive priority

```
1. while $\mathcal{I} \neq 0$
2.     sort $\mathcal{I}$ using a local ordering,
       possibly based on items that have already been seen and decided upon.
3.     say $I_{first}$ is first ordered in this ordering
4.     make irrevocable decision for $I_{first}$
5.     remove $I_{first}$ from $\mathcal{I}$
6. end while
```

2.2 Examples

So far all our optimization problems have been as follows :
$\mathcal{I} = \{I_1, I_2 ... I_n\}$ as input set
and
(I_1, d_1), ..., (I_n, d_n) as output. (d_j represents the decision for input item I_j)

Examples :
Makespan on restricted machine
The input is $I_j = (p_j,\ S_j)$ where p_j is processing time of job j and $S_j \subseteq \{1, ..., m\}$ is the set of allowable machines for job I_j.
d_j is the machine on which we choose to run I_j.

Set Cover
The input is $I_j = S_j = (\{v_j^1, ... v_j^{t_j}\},\ w_j)$ such that we specify the weight of the set and the items in the set.
In this case $d_j = 0$ if we have chosen not to include S_j and 1 if we chose to include it.

Graph problems can be given in two different models :

Edge input model $I_j = e_j = ((u_j, v_j), w_j)$
Vertex adjacency model $I_j = v_j = (u_i : u_i \in \mathrm{N}(v_j))$

There are thus many representations for a given problem,
Interval selection (s_j, f_j, w_j)
Interval graph MIS $(v_j, \mathrm{N}(v_j), w_j)$
Combined representation $(s_j, f_j, w_j, \mathrm{N}(v_j))$

2.3 Defining Local Ordering

We provide two definitions of “local ordering” used in the models for greedy algorithms earlier :

Definition 1
Specify an f : $\mathcal{I} \Rightarrow \Re$
And obtain f(I_j) = $r_j \in \Re$
By notation f(I_j) does not depend on I_k k$\neq$ j
The local ordering of I_j is such that $r_1 \leq ... \leq r_n$

Definion 2
$\mathcal{I} = \{I_1, I_2 ... I_n\}$ set of actual input items.
$\mathcal{J}$ = set of all possible input items
Provide a total ordering on $\mathcal{J}$.
Then $\mathcal{I}$ inherits this ordering and we call this the local ordering.

```
1. while $\mathcal{I} \neq 0$
2.       provide a total ordering on $\mathcal{J}$ let $I_{first} \in \mathcal{I}$ be first in this order
         make irrevocable decision about $I_{first}$
3.       remove $I_{first}$ from $\mathcal{I}$ and $\mathcal{G}$
4.       also remove I $\in \mathcal{J}$ for which I $< I_{first}$
5. end while
```

Both definitions have a “IIA” aspect. Independence of irrelevant attributes. By this we mean our preference between A and B does not depend on the existence of C.

The previous definitions attempt to capture what we mean by “greedy-lie”. We use the term greedy priority to indicate that the irrevocable decision is a “live for the moment” decision in the sense of optimizing the current decision (ignoring the future).

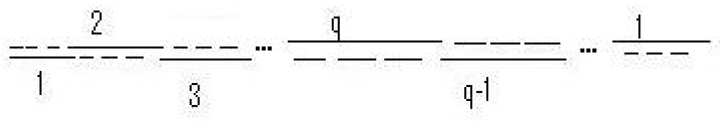
3 Negative Results and APX-Hard Problems

No priority algorithm can achieve a c-approx for:
1) Proportional profit interval schedule for c<3
Given $I_j = (s_j, f_j, w_j)$ proportional profit means $w_j = |f_j - s_j|$ (or equally some fixed multiple). Using the “LPT” greedy (fixed order priority) algorithm we achieve a 3-approximation. “LPT” means we sort so that $w_1 \geq ... \geq w_n$ then we choose intervals greedily. The negative result is that

we can do no better for priority algorithms.
For arbitrary weighted interval schedule priority algorithms cannot acheive a c-approximation for any constant c.

The method for proving such ideas is to let the algorithm pick and argue that it has made a mistake such that it can't make back the value of the items it has failed to pick.

We show (1) for proportionally weighted intervals.



If we choose a small interval we can remove the other small intervals that are near such that we have now missed out on a big interval that is 3 times larger than the one we picked that we can no longer make up with small jobs.

If we pick a large job we miss out on the two adjacent large jobs since they slightly overlap and we can remove the small jobs that are subintervals of the adjacent large jobs such that we can't make up what we've lost.

Alternatively, we could have picked the 3 small jobs and 2 adjacent large jobs to have nearly 3 times the weight we have included. Thus for any choice we miss out on about 3 times what we could have had. So no c-approx is possible for c<3.

Unweighted interval scheduling has a greedy optimal algorithm (even on m machines). Namely we sort such that $f_1 \leq ... \leq f_n$. Also "SPT" is a 2-approx (Intervals taken by shortest first). We can extend this to m machines by still using $f_1 \leq ... \leq f_n$ and choosing the machine which is still feasible and leaves the least gap between the previous finishing time and this jobs starting time.

Interval graphs are **chordal graphs**. One characterization of chordal graphs is that there exists an ordering of the vertices v_1, ..., v_n such that N(v_i) $\cap$ $\{v_{i+1}$, ..., $v_n\}$ is a clique. Such an ordering is called a perfect elimination order, PEO. Unweighted MIS for chordal graph has an optimal fixed priority greedy algorithm using the PEO (for 1 machine).

Graph coloring problem
χ : V $\Rightarrow$ {1, ..., k} so as to minimize k given (u,v) $\in$ E $\Rightarrow$ (χ(u) $\neq$ χ(v))
Using $s_1 \leq ... \leq s_n$ as the sorting is bad for interval scheduling but for interval coloring it is optimal.

Optimal coloring can also be derived from the reverse of a PEO. Suppose the maximum number of previously colored neighbors in this order is k. To color with any coloring we need at least k+1 colors since the previously colored form a clique in this order. We do not need more than k+1 since at no point do we have more than k previously colored neighbors by assumption. Thus this ordering provides an optimal fixed priority algorithm.

A PEO for interval graphs is given by $f_1 \leq ... \leq f_n$. But the interval graph from right to left is still an interval graph and provides a second PEO. This PEO places $s_n \geq ... \geq s_1$ (earliest finishing time in reverse is latest starting time), so the reverse is just $s_1 \leq ... \leq s_n$.

Min sum coloring

Goal is to get a feasible coloring χ : V $\Rightarrow$ {1, ..., ∞} so as to minimize Σ $\chi(v_i)$ while having the colors of any two adjacent nodes not be equal.

Min sum multi-coloring
The problem is similar except now v has demand d(v) and we must assign it d(v) colors.
Non-preemptive : d(v) colors must be consecutive.

Min sum coloring is NP hard for interval graph and APX-hard.

Definition APX-hard $\Rightarrow$ no 1+ϵ approximation for sufficiently small ϵ. ie. no PTAS exists (assuming $P \neq NP$).

Note, proper interval graphs have greedy optimal algorithm for min sum coloring.
Open. min sum preemptive multi-coloring for proper interval graph.

Definition G=(V, E) inductive k-indep : if there exists and ordering v_1, ..., v_n such that N(v_i) $\cap$ {v_{i+1}, ..., v_n} has at most k-indep vertices.

Unweighted MIS problem on inductive k-independent ordering gives k-approximation. This is because if we choose a node in this order we miss out on at most k others.

(W) JISP
The input is $I_j = (s_j, f_j, w_j, c_j)$, where c_j is the class of jobs that I_j belongs to. A feasible schedule for such jobs means, I_j and I_k accepted for j $\neq$ k $\Rightarrow$ $I_j \cap I_k = 0$ and $c_j \neq c_k$ (no more than one of each class)

The graphs induced by (W) JISP are inductively 2-independent. Therefore the greedy algorithm using fixed priority $f_1 \leq ... \leq f_n$ for unweighted JISP is a 2-approx. However, the problem is APX-hard.