CSC 373: Algorithm Design and Analysis Lecture 3

Allan Borodin

January 11, 2013

Lecture 3: Outline

• Write bigger and get better markers

• A little more on charging arguments

- Continue examples of greedy algorithms
 - Maximal Matching algorithm for minimum vertex cover
 - 2 m machine interval scheduling (m-ISP)
 - Interval colouring

Charging arguments

- A common method for proving optimality or approximation results for an optimization algorithm ALG is by a charging argument.
- For a profit maximization problem:
 - Charge the profit of an arbitrary solution (and hence that of an optimal solution OPT) to the profit of your ALG.
 - 2 Argue that not too much profit from *OPT* gets charged to ALG.

• For a cost minimization problem:

- Charge the cost of ALG to an OPT solution
- Argue that not too much cost from ALG is charged to OPT.

Charging argument for EFT (as discussed last lecture)

- For the (unweighted) ISP problem, the profit of an algorithm is simply the number of intervals selected.
- For an input I, we will write
 - ► |ALG(I)| to denote the profit of algorithm ALG
 - |OPT(I)| to denote an optimal solution
- Then to show optimality of EFT for ISP, it suffices to show that

There is a 1-1 function h mapping OPT(I) into EFT(I).

• Since *OPT* is an optimal solution, the mapping must be onto.

Charging argument to obtain approximation bound

- As stated in the first class, I like to integrate some results about approximation algorithms as we proceed rather than treat approximation algorithms as a separate topic.
- We can easily adapt the EFT algorithm so as to apply to the JISP problem.
- JISP problem: Two intervals are compatible if
 - they do not intersect and
 - they do not belong to the same job class.

Claim (Question 2 of problem set)

For the JISP problem we can show that the same h is a 2-1 function mapping OPT(I) into EFT(I).

A greedy algorithm for the vertex cover problem

Vertex Cover problem

- Given: A graph G = (V, E). A vertex cover for G is a subset V' ⊂ V such that for every edge e = (u, v) ∈ E, at least one of u or v is in V'.
- The goal is to find a minimum size vertex cover.

Maximal Matching (MM)

[greedy algorithm for vertex cover]

```
\begin{array}{l} V':=\varnothing\\ \textbf{While } E\neq\varnothing\\ \textbf{For any } e=(u,v)\in E, \ V':=V'\cup\{u,v\}\\ \text{\%add both endpoints of the edge to the cover}\\ \textbf{Delete all edges from } E \text{ that are adjacent to this edge } e.\\ \textbf{End While} \end{array}
```

The MM greedy algorithm is a 2-approximation for vertex cover

- Let OPT(G) be any (e.g. an optimal) vertex cover and MM(G) be the solution of algorithm MM for graph G.
- Then MM(G) is a vertex cover for G and |MM(G)| ≤ 2 ⋅ |OPT(G)| for all graphs G.

The *m*-ISP problem

• The *m* "machine" interval scheduling problem (*m*-ISP) schedules a set of intervals on *m* machines so that

intervals assigned to the same machine do not intersect.

• We next consider two extensions of the one machine EFT algorithm.

First Fit EFT

1: Sort intervals so that $f_1 \le f_2 \le \ldots \le f_n$ 2: for j = 1 to n do 3: $k := \begin{cases} \min\{\ell : I(j) \text{ doesn't intersect intervals on machine } \ell \} & \text{if such } \ell \text{ exists} \\ 0 & \text{if no such } \ell \end{cases}$ 4: $\sigma(j) := k$ $\% \sigma(i)$ specifies if and on which machine interval I(j) is scheduled 5: end for

Fact

First Fit EFT is not an optimal algorithm

An optimal greedy algorithm for *m*-ISP

Another extension of the one machine EFT algorithm:

Best Fit EFT 1: Sort intervals so that $f_1 < f_2 < \ldots < f_n$ 2: for k = 1 to m do 3: $e_k = 0$ % e_k specifies the lastest completion for intervals on machine k 4: end for 5: **for** i = 1 to *n* **do** $\text{Let } k := \begin{cases} argmin_{\ell} \{ s_i - e_{\ell} \ge 0 \} & \text{ if such } \ell \text{ exists} \\ 0 & \text{ if no such } \ell \end{cases}$ 6: 7: $\sigma(i) := k$ % $\sigma(i)$ specifies if and on which machine interval J(i) is scheduled $e_{k} := f_{i}$ 8. 9: end for

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.



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.
- Note: This is a simple example of a polynomial time reduction which is an essential concept when we study NP-completeness.
- Note: There are examples of graph classes where the colouring problem can be efficiently computed whereas on this class of graohs, the *m*-ISP problem is NP-hard when *m* is a parameter of the problem.

Greedy interval colouring

• 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

Greedy Interval Colouring

- 1: Sort intervals so that $s_1 \leq s_2 \leq \ldots \leq s_n$
- 2: for i = 1 to n do
- 3: $k := \min\{\ell : \ell \neq \chi(j) \text{ for all } j < i \text{ such that the } j^{th} \text{ interval intersects }$ the i^{th} interval}
- 4: σ(i) := k /* the ith interval is greedily coloured by the smallest non conflicting colour */
- 5: end for

An example of interval colouring



• We use the colors in the following order: red, yellow, blue, green

Idea of optimality proof

Look at the interval that (first) caused the largest colour, say k, to be used by EST. Then there must be k intervals containing a given "time" x and hence k colours are required.