CPSC 420 — March 27 Lecture

Scribe: Stephanie Knill

1 Approximation Algorithms

Definition 1.1. Approximation Algorithm[1]

ρ-approximation if for every input I with optimal solution value OPI(I), then output of algorithm A(I) is close, i.e.

$$\max\left\{\frac{A(I)}{OPT(I)}, \frac{OPT(I)}{A(I)}\right\} \leq \rho$$

1.1 Application: minimium vertex cover

Given an undirected graph $G = (V, E)$, find the smallest set of vertices $S \subseteq V$ such that all edges in G have at least one endpoint in the set S.

Approaches

1. **Greedy Algorithm**

```
Repeat:
  Include in S the vertex with highest degree
  Remove vertex from G and all incident edges
```

Pros / Cons

- (-) Guaranteed Approximation factor gets worse/grows as $\uparrow n$:

$$|\text{GreedyVC}(G)| \leq \log n \cdot |\text{OPTVC}(G)|$$

[1]See Jeff Erikson's notes on Approximation Algorithms for further reading.

2. **Matching Vertex Cover (MVC) Algorithm**

```
Repeat:
  S = {}
  Pick any edge (u, v) in G
  Remove (u, v) from G and all edges adjacent to u or v
  Add u and v to S
```

Pros / Cons

- (+) Better Approximation: guarantees for all graphs

$$|\text{MVC}(G)| \leq 2 \cdot |\text{OPTVC}(G)|$$

Proposition 1.1. *MVC is a 2-approximation for VERTEX COVER*

- *Rmk: We don't know how big OPTVC(G) is, but we can lower bound its size*

Proof. Since $|\text{OPTVC}(G)| \geq$ #edges picked by MVC (because edges selected by MVC form a matching, thus no vertex covers more than one edge in a matching), then the #vertices picked by MVC is $2\cdot$ #edges picked. Thus we have

$$|\text{MVC}(G)| \leq 2 \cdot |\text{OPTVC}(G)|$$

■

1.2 Application: list scheduling (Graham, 1966)

Given

- n jobs, where job i must execute uninterruptedly for p_i time units
- m (identical) machines, where each machine can work on one job at a time

Find a schedule of jobs that minimizes the completion time (time when last machine finishes).

Approaches

1. **Greedy Algorithm**

```
Repeat:
  Whenever machine beomes idle, assign next job to it
```

Pros / Cons

- (+) Decent Approximation: guarantees for all graphs

$$|\text{Greedy}(p_1, \ldots, p_n)| \leq \left(2 - \frac{1}{m}\right) \cdot |\text{OPT}(p_1, \ldots, p_n)|$$

Proposition 1.2. *MVC is a* $\left(2 - \frac{1}{m}\right)$*-approximation for LIST SCHEDULING*

Proof.

- $\text{OPT} \geq p_i, \quad \forall i$
- $\text{OPT} \geq \frac{\sum_i p_i}{m}$
- Let job k be last job to finish. Then $p_k \leq \text{OPT}$ and the time when job k starts executing s_k is bounded by
$$s_k \leq \frac{\sum_{i \neq k} p_i}{m}$$
Thus
$$\begin{aligned} s_k + p_k \leq \frac{1}{m} \sum_{i \neq k} p_i + p_k &= \frac{1}{m} \sum_i p_i + (1 - 1/m) p_k \\ &\leq OPT + (1 - 1/m) OPT \\ &= (2 - 1/m) OPT \end{aligned}$$

which gives us our desired result

$$|\mathrm{MVC}(G)| \leq \left(2 - \frac{1}{m}\right) \cdot |\mathrm{OPTVC}(G)|$$

■

Example
For the $n = 6$ jobs with time units $p = \{5, 7, 17, 10, 9, 30\}$ the greedy algorithm (Figure 1) gives us

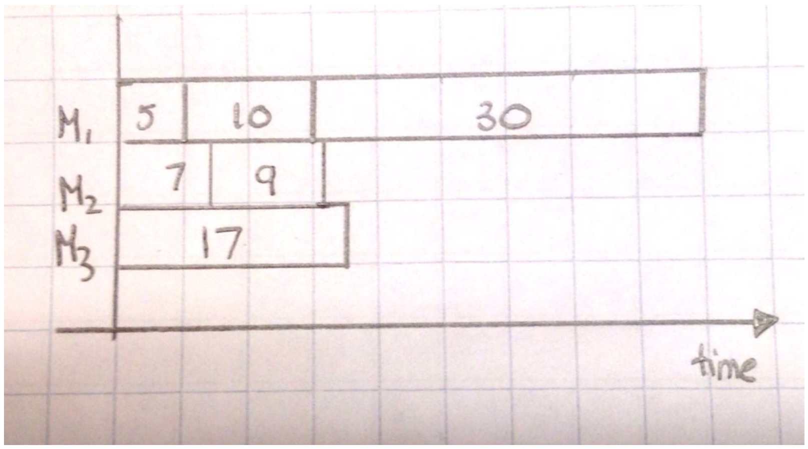

Figure 1: Greedy algorithm for list scheduling problem.

*a total time of **45** units.*

<u>Sorted Greedy Algorithm</u>

```
Repeat:
  Sort job array in descending order
  Greedy Algorithm: whenever machine beomes idle, assign next job to it
```

Example
*For the $n = 6$ jobs with time units $p = \{5, 7, 17, 10, 9, 30\}$ the greedy algorithm (Figure 2) gives us a total time of **30** units.*

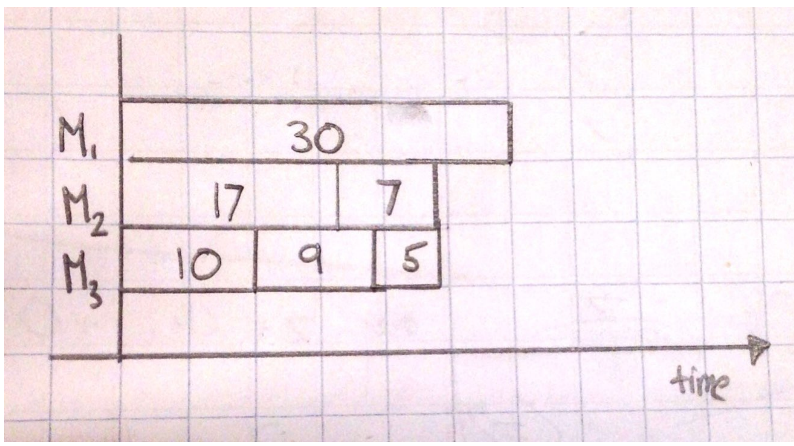

Figure 2: Sorted greedy algorithm for list scheduling problem.