Assignment 1 - Homework Exercises on Approximation Algorithms

Pattarawat Chormai, Duy Pham

September 8, 2015

1 A.I-1

We will show that the approximation ratio of the $GreedySchedulingAlgorithm$ is at least $2 - \frac{1}{m}$ by showing an example as follow.

Let's consider this setting:

- 3 machines: M1, M2, M3
- 7 jobs: 1, 1, 1, 2, 1, 1, 2

$GreedySchedulingAlgorithm$ will come up with this scheduling:

- M1: 1 2 2
- M2: 1 1
- M3: 1 1

Thus $ALG = makespan = 1 + 2 + 2 = 5$
We know that:

$$OPT \geq Average_{load} \tag{1}$$

Where $Average_{load} = \frac{1}{m} \sum_{i=1}^{n} j_i = \frac{9}{3} = 3$
Here we can find a solution with $makespan = 3$. That is:

- M1: 1 2
- M2: 1 2

- M3: 1 1 1

Therefore, $OPT = 3$
Thus, the approximation ratio is:

$$\rho = \frac{ALG}{OPT} = \frac{5}{3} \tag{2}$$

According to the theorem, the estimated ratio is:

$$\rho_{estimated} = 2 - \frac{1}{m} = 2 - \frac{1}{3} = \frac{5}{3} \tag{3}$$

From 2 and 3, we have $\rho_{estimated} = \rho$. Therefore, this bound is tight.

2 A.I-2

From the question, we know that

$$\begin{aligned} m &= 10 \\ \sum_{j=1}^{n} t_j &\geq 1000 \\ t_j &\in [1, 20]\,; for\, all\, i \leq j \leq n \end{aligned}$$

Let T'_{i^*} denote the load of M_i before t^*_j, last job, is assigned to the machine. Thus T^*_i, which represents makespan of the assignment, equals to

$$T^*_i = T'_{i^*} + t^*_j$$

Because T'_{i^*} is the minimum load among all machines, so that we can derive

$$T'_{i^*} \leq \frac{1}{m} \sum_{i=1}^{m} T'_i = \sum_{j=1}^{j^*} t_j \leq \frac{1}{m} \left[\sum_{j=1}^{n} t_j - t^*_j \right] \leq LB$$

Then we can derive

$$\begin{aligned} T_i^* &= T'_{i^*} + t_j^* \\ &\leq \frac{1}{m}\left[\sum_{j=1}^{n} t_j - t_j^*\right] + t_j^* \\ &\leq \frac{1}{m}\sum_{j=1}^{n} t_j + (1 - \frac{1}{m})t_j^* \\ &\leq 100 + (1 - \frac{1}{10})20 \\ &\leq 118 \end{aligned}$$

According *Algorithm Greedy Scheduling* and the question, we know

$$max\left(\frac{1}{m}\sum_{j=1}^{n} t_j \,, \max_{1\leq j\leq n}(t_j)\right) \leq LB \leq OPT$$

$$\max_{1\leq j\leq n}(t_j) = 20$$

Then we can derive

$$max\left(\frac{1}{m}\sum_{j=1}^{n} t_j \,, 20\right) \leq LB$$

Since $\frac{1}{m}\sum_{j=1}^{n} t_j \geq 1000$, thus

$$\begin{aligned} 100 &\leq LB \\ &\leq OPT \end{aligned}$$

Therefore, approximation-ratio(ρ) equals to

$$\begin{aligned} T_i^* &\leq \rho\, OPT \\ \frac{118}{100} &\leq \rho \\ 1.18 &\leq \rho \end{aligned}$$

For this particular setting, *Algorithm Greedy Scheduling* is 1.18 approximation algorithm.

3 AI-3-i)

Assume we have the optimal solution, which has n squares

$$n \leq LB \leq OPT$$

Lemma 1. *Each unit square in the grid can overlap at most 4 cells. Let n_s be the number of square in the integer grid solution. Thus*

$$n_s \leq 4n \leq 4OPT$$

4 A.I-3-ii

We propose the algorithm as follow.

Algorithm 1 Finding minimum row square cover

Require: Set of Points P
Ensure: Min Square Cover min
Operation:

```
set currentCoveringPosition = 0
QuickSortAscending(S)
for all Point p in P do
    if p.x ≤ currentCoveringPosition then
        create square s = (p.x, 1, p.x + 1, 0);
        add s to S
        set currentCoveringPosition = p.x + 1
    end if
    set min = sizeofS
     return min
end for
```

This algorithm is correct because:

- Every point in p will be covered by a square
- There are no intersections between the squares because we traverse in one direction

This algorithm consists of 2 parts: QuickSort and Traversing the Point to create squares. Let t be the run time of this algorithm, $t_{quicksort}$ be the time for quick-sort, and t_{assign} be the time for creating the squares. We have:

$$t = t_{quicksort} + t_{assign} \leq n \log n + n = O(n \log n) \tag{4}$$

Thus the runtime of this algorithm is $O(n \log n)$.

5 A.I.3.iii

The idea of our algorithm is that, we put all the points in to a coordinate system, then we divide the coordinate system into a set of unit rows (i.e. rows with height 1). For each row, we use algorithm 1 to find the minimum size square-cover. The global min-square-cover is the sum of all row-square-cover.

Algorithm 2 Finding global minimum square cover

Input: Set of points P
Output: Min Square Cover min
Operation:
currentMin = 0;
for all Row r in the space **do**
currentMin += FindRowMinSquare()
end for
set $min = currentMin$
return min

Theorem. FindingGlobalMinimumSC is $2 - approximation$

Proof. Suppose we have the optimal solution with n squares. Then $OPT \geq n$, thus $LB = n$. We know that each unit square in the coordinate system can overlap with at most 2 unit rows.

Let min be our algorithm solution, then:

$$min \leq 2n = 2.LB \leq 2.OPT \tag{5}$$

Therefore, this algorithm is $2 - approximation$.

□