Exercise 4.4

donderdag 14 september 2023 15:44

(i) There are at most $\frac{1}{\varepsilon}$ large jobs; the time taken by those jobs would then be at least $\frac{1}{\varepsilon} \cdot \varepsilon \cdot T = T$, whereas the total size of all jobs cannot be larger than this. Also, since the number of large jobs must be an integer, we can say that the number of large jobs is at most $\lfloor \frac{1}{\varepsilon} \rfloor$.

The number of ways in which $\lfloor \frac{1}{\varepsilon} \rfloor$ large jobs can be distributed among the two machines is $2^{\lfloor \frac{1}{\varepsilon} \rfloor}$.

Furthermore, the number of 'distinguishable' such ways is $2^{(\lfloor \frac{1}{\varepsilon} \rfloor - 1)}$, since it does not matter whether the machines are swapped.

(ii) Consider the following algorithm:

```
Load-2($t_1, \dots, t_n, \varepsilon$):

1.  $T = \sum_{i=1}^{n} t_i$                                   # time $O(n)$
2.  large jobs ← new array of size $\lfloor \frac{1}{\varepsilon} \rfloor$, filled with zeroes      # time $O(\lfloor \frac{1}{\varepsilon} \rfloor)$
3.  small jobs ← new array of size n, filled with zeroes
4.  $j$ ← 1
5.  $k$ ← 1
6.  for $i = 1$ to $n$
7.      if $t_i \geq \varepsilon \cdot T$
8.          large jobs[$j$] ← $t_i$
9.          $j$ ← $j+1$
10.     else
11.         small jobs[$k$] ← $t_i$
12.         $k$ ← $k+1$

13. OPT-large ← result of a brute-force search for the optimal solution to
                load-balancing among all large jobs; i.e. result among the jobs with times in 'large jobs' array.

14. solution ← Apply Greedy-Scheduling to the jobs in the small jobs array, initializing its state such that the large
               jobs are already assigned to the machines as determined on the previous line.

15. return solution
```

To prove the approximation ratio, we distinguish between 3 cases:

1. All jobs are large. In that case, we obviously get an optimal solution (by line 13 of the algorithm) and hence the required approximation ratio of $1 + \epsilon$ is achieved.
2. There are only small jobs. In this case, we note that a lower bound on the size of the optimal solution is given by $\frac{T}{2}$ (since we have two machines, one of which must run at least half of the total workload). Furthermore, the maximum difference in the workloads between the machines in the solution computed by the Greedy-Scheduling algorithm is at most $\epsilon \cdot T$. To see why, consider that all small jobs have size at most (infinitesimally less than) $\epsilon \cdot T$. Hence, if the difference in workload is at least $\epsilon \cdot T$, then there must be at least one job smaller than $\epsilon \cdot T$ running on the machine determining the makespan that can be moved to the other machine to reduce the makespan. We then have that the maximum size of the solution computed by Greedy-Scheduling is at most $\frac{1}{2}T + \frac{1}{2}\epsilon T$. *(To see why, note that, since we have two machines, the load must be divided among them. This gives that one machine has load $\frac{1}{2}T + x$ and the other has load $\frac{1}{2}T - x$. If the difference between the two, which is given by $\frac{1}{2}T + x - \left(\frac{1}{2}T - x\right) = 2x$, may be at most ϵT, it then follows that, in the worst case, $x = \frac{1}{2}\epsilon T$. This gives the largest machine a load of $\frac{1}{2}T + \frac{1}{2}\epsilon T$.)* The approximation ratio achieved by the algorithm will then be $\frac{\frac{1}{2}T + \frac{1}{2}\epsilon T}{LB} = \frac{\frac{1}{2}T + \frac{1}{2}\epsilon T}{\frac{1}{2}T} = 1 + \epsilon$, which is indeed the maximum allowable approximation ratio for a PTAS.
3. There are both large and small jobs. In this case, we can distinguish between two different cases:
    a. The last job assigned to the machine which determines the makespan (i.e. the machine with the largest load) was a large job. In this case, no small jobs were assigned to the machine determining the makespan. This means that the solution must be optimal, since we effectively have the same solution as the one to the problem consisting of only the large jobs. Furthermore, we note that adding jobs to the problem can never decrease the size of the optimal solution, and hence, if the solution found for the problem including the small jobs is as large as the optimal solution to the problem excluding the small jobs, then the solution found must be optimal for the problem including the small jobs as well.
    b. The last job assigned to the machine which determines the makespan (i.e. the machine with the largest load) was a small job. This means that the machine with the largest load must have had a small job assigned to it as the last job. But in that case, the difference in load between the two machines may be at most (infinitesimally less than) $\epsilon \cdot T$, as the last job determining the makespan (which has size less than $\epsilon \cdot T$) would otherwise have been assigned to the other machine. From this point on, the same reasoning as given under case 2 applies and hence we conclude that the approximation ratio is within the allowable range for a PTAS.

Hence, the algorithm achieves the required approximation ratio for a PTAS.

To analyze the running time, we consider the following:

1. Line 1 runs in $O(n)$ time.
2. Line 2 runs in $O\left(\left\lfloor \frac{1}{\epsilon} \right\rceil\right)$ time.
3. Line 3 runs in $O(n)$ time.
4. Line 4 runs in $O(1)$ time.
5. Line 5 runs in $O(1)$ time.
6. The loop in lines 6-12 runs in $O(n)$ time (since all lines in the loop can be concluded in $O(1)$ time and the loop runs for n iterations).

13. A brute-force search over $2^{\left\lfloor \frac{1}{\epsilon} \right\rceil}$ possible options, as performed in line 13, runs in $O\left(2^{\left\lfloor \frac{1}{\epsilon} \right\rceil}\right)$ time.
14. Applying the Greedy-Scheduling algorithm, as done in line 14, runs in $O(n \log 2) = O(n)$ time (strictly speaking, n should be replaced by the size of the filled-in parts of the small jobs array, but we ignore this for simplicity).
15. Line 15 runs in $O(1)$ time.

All in all, the algorithm runs in $O\left(2^{\left\lfloor \frac{1}{\epsilon} \right\rceil} + n\right)$ time, which satisfies the demands for a PTAS (but not for an FPTAS).