Advanced Algorithms Homework 5

Vijayraj Shanmugaraj, 20171026

April 13, 2021

Exercise 1

For a case that is very close to 2, consider m machines, and $m \cdot (m-1)$ jobs of time 1. They will be balanced among all machines, each machine having makespan m-1. Then finally a long job of time m arrives in the end. This makes the makespan $2m-1$. In the ideal case, one machine would have the large job of size m, and the other $m \cdot (m-1)$ jobs would be distributed across the $m-1$ machines, making the ideal makespan m. So the approximation ratio for this case is $\frac{2m-1}{m}$, which tends to 2 for large values of m.

Exercise 2

First, we will try putting a tighter bound on the approximation ratio of the sorted greedy assignment algorithm (SGA). We will show that the SGA is a $\frac{4}{3}$ approximation algorithm.

Say there is an instance $P_1, P_2, \dots P_n$ for which SGA gives a makespan $C_n > \frac{4}{3} \cdot T^*$. Say without loss of generality, $P1 \geq P2 \geq \dots \geq Pn$. We can claim that the job that defines the makespan i.e. the one that finishes last is actually the job P_n (the one with the smallest processing time. Suppose that was not the case.

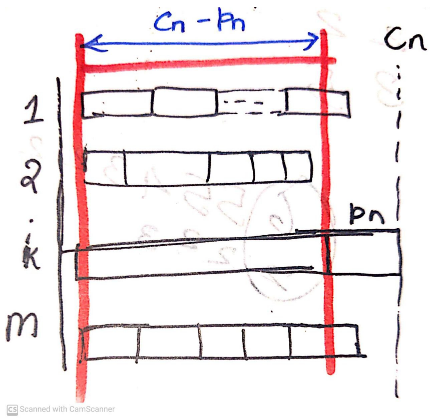
Say some other job L with time P_L defined the makespan and $P_L \geq P_n$. Now if we run SGA on jobs 1 to L, we would get the makespan of this subset to be $C_l = C_n$ again, since in the case of n jobs, the C_n is defined by the L'th job. And moreover the optimal solution of the subset of L jobs can only be lesser than or equal to the optimal solution of n jobs (because we have fewer jobs in the subset).

Now, if the above instant with n jobs is not within a factor of 4/3, we can also say that the approximation ratio of the subset of the first L jobs will not be within a factor of 4/3 as well. In that case we can take the subset $P_1, P_2, \dots P_L$ too. In this case C_n would be defined by P_L (the last job) in that list, and hence our analysis would be similar to the case when the n'th job is the last one.

Hence, let us proceed taking set $P_1, P_2, \dots P_n$ assuming that that P_n defines C_n, we can consider two cases:

Case 1: Length of nth job $P_n \leq T^*/3$.

Now if we exclude the n'th job, we can clearly say $\sum_j P_j \geq m \cdot (C_n - P_n)$ (See picture) and we also know that $T^* \geq \frac{1}{m} \sum_j P_j$.



Now,

$$
\begin{aligned}
T^* &\geq C_n - P_n \\
\frac{T^*}{3} &\geq P_n \\
\implies C_n &\leq \frac{4 \cdot T^*}{3}
\end{aligned}
$$

Case 2: $P_n > T^*/3$.

Then, $T^* < 3P_n$. Since P_n is the smallest processing time this will imply that the optimal schedule has at most 2 jobs per machine. (Say we have 3 or more jobs on a machine. We know that $P_i \geq P_n$. And hence, the makespan of that machine $\geq 3P_n > 3 \cdot (T^*/3) > T^*$ which is not possible. So the number of jobs $J \leq 2m$.

When the number of jobs $\leq 2m$ and each processor has at most 2 jobs, we can show that the greedy sorted algorithm gives the best allocation.

Proof: Say there are $k < 2m$ jobs. Processors $\phi_1, \phi_2, ..., \phi_{2m-k}$ will have one job (ϕ_i will have job i), and the other processors will have two jobs (ϕ_i will have job i and job $2m - i + 1$). There are two cases here:

Case 1: Processor i defines C_n, $i \leq 2m - k$.

In this case, nothing can be done, since only one job defines C_n. We know that $T^* \geq max_j P_j$, which in our case is P_1. And this case signifies the lower bound of T^* i.e. $max_j P_j$. And hence $C_n \equiv T^*$

Case 2: Processor i defines T^*, $i > 2m - k$.

In this case, $P_{2m-i+1} + P_i$ is the C_n. Now, in order to decrease the time, we can do one of the following:

Case 2a) Shifting $P_{2m-i+1} = P_b$ (say) to any other processor: Now shifting P_b to any other processor (with one job - since we have at most two jobs on a processor) would be detrimental, since $P_i \leq P_a$, $a \leq 2m - k$ (since the first m jobs are allocated in descending order of time). So $P_b + Pi \leq P(b)|P(a) \forall$ possible a. (Note: If k = 2m only case 2b) holds).

Case 2b) Swapping $P_{2m-i+1} = P_b$ (say) with P_g, $g > b$: We try swapping P_b with some smaller P_g such that we get a smaller C_n. But this means that we would be taking from processor 2m-g-1. Now, the second round of allocation in the greedy sorted process with at most two processes on a machine would happen from processor m to processor 2m-k+1 (in reverse). So, when we're looking for a smaller task P_g, we must remember that $P_{2m-g-1} \geq P_i$. So when we swap P_b with P_g, the makespan on processor 2m-g-1 would be $P_{2m-g-1} + P2m - i + 1$, which is greater than $P_{2m-i+1} + P_i$. Hence we cannot do better than $P_{2m-i+1} + P_i$ for the best makespan. Hence, here, $C_n = T^*$

So we have shown that the approximation ratio of this algorithm is actually $\frac{4}{3}$, a tighter bound.

Now for the example as close to the ratio of 1.5, we can get as close as possible only to $\frac{4}{3}$. Take m machines, and 2m + 1 jobs. We have 2 jobs each of time m, m+1, ... 2m-1 and one more job of length m. The best achievable makespan would be 3m (all 3 m's on one processor, and pairs of jobs that add up to 3m i.e. (2m-1, m+1), (2m-2, m+2) etc.), but the SGA would give a makespan of 4m-1. (Basically any processor would have paired up jobs in such a way that they all have a makespan of 3m-1, and the final job of time m would add it up to 4m-1 for the processor it gets assigned to). So the ratio would be $\frac{4m-1}{3m}$ Which tends to $\frac{4}{3}$ as m gets larger.

Exercise 3

Take a perfect binary tree of a depth h (say h ≥ 2 so that we have enough nodes for analysis). Note that the definition of a (c-d) FIS $S \subseteq V$ in G is as follows:

- S is independent
- $\forall v \in S$, deg(v) ≤ d
- |S| ≥ |V|/c

Take d = 1. Now the set of all leaf nodes, say L, is an independent set, moreover they constitute a (c, 1) independent set. Now to calculate c,

$$\begin{aligned}
\frac{1}{c} &= \frac{|L|}{|S|} \\
&= \frac{2^h}{2^{h+1} - 1} \\
&= \frac{1}{2 - \frac{1}{2^h}} \\
c &= 2 - \frac{1}{2^h} \leq 2
\end{aligned}$$

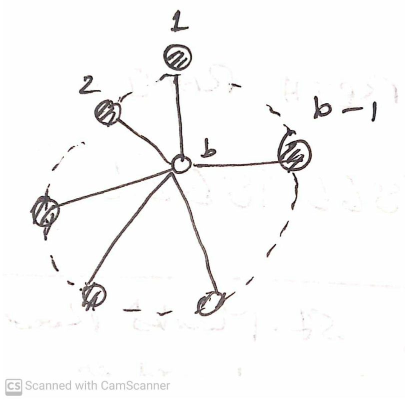
$c \leq 2$, which means the fraction of nodes that have degree 1 is greater than half the nodes, and hence $|L| \geq |V|/2$. Hence L can be considered as a (2, 1) FIS of |S| **(Example for c = 2)**. Now since $|L| \geq |V|/2$, we can take a set $T \subset L$ such that $|V|/3 \leq |T| < |V|/2$. So |T| will be a (3, 1) FIS of |S| **(Example for c = 3)**.

Exercise 4

To start off with, we need $c \geq 1$. Since if $c < 1$, |S| ≥ |V|/c, which means $|S| > |V|$, since $c < 1$. (It goes unsaid that $c > 0$). Now if c = 1, $|S| \geq |V|$, but $S \subseteq V$. So the only possibility for c = 1 is a graph with nodes having no neighbours i.e. a graph with no edges. We take d = 1 for this, and since $\forall v \in V$, $deg(v) = 0$, It satisfies the second condition of FIS as well. So this graph is a (1,1) FIS.

Now, for $c > 1$, we need a subset of independent vertices such that $|S|/|V| \geq 1/c$. We know that S and V are positive integers, and hence, $|S|/|V|$ is a rational number. Now we know that $1/c \leq |S|/|V| \leq 1$. We have two cases in front of us:

Case 1: c is rational: Since c is rational, $1/c$ will be of the form a/b, a < b, $a, b \in \mathbb{N}$ (Since c > 1 $\implies$ 1/c < 1). So for this case, take a graph with b nodes. Now, add b-1 edges, one edge each from v_i, $i = 1, 2 ... b - 1$ to v_b, and set d = 1. Clearly, this is a ($b/(b-1)$, d) FIS.



Now from this FIS (say S), chose any a vertices and put it in a set T. This set T will be a (b/a, 1) = (c, 1) FIS of G.

Case 2: c is irrational: Now, since c is irrational and $c > 1$, $1/c \in (0, 1)$ and is irrational too. Now, according to the Archimedian property of natural numbers, if we have two real numbers x_1, x_2, such that $x_1 - x_2 < 1$, there exists an $b \in \mathbb{N}$ such that $b \cdot (x_1 - x_2) > 1$. This implies that the distance between $b \cdot x_1$ and $b \cdot x_2 > 1$, and hence $\exists a \in \mathbb{N}$ such that $b \cdot x_1 < a < b \cdot x_2$. And hence this implies that there is a rational number a/b such that $x_1 < a/b < x_2$. Now we can take $x_1 = 1/c$ and $x_2 = 1$. Hence we will get a rational number a/b. And just like the above example, we can construct a graph with a (b/a, 1) FIS, where b/a is the closest rational number less than c (and hence $\frac{a}{b} \cdot |V| > \frac{|V|}{c}$, but is as close as possible, satisfying the FIS conditions.