COMS E6232 - Problem Set #3

Alex Wong (asw2181) - asw2181@columbia.edu

March 23, 2017

Problem 1

a. Consider an instance where we have two items x and y where $s_x = \epsilon$, $v_x = 2\epsilon$, $s_y = B$, $v_y = B$, and $\epsilon << 1$. The v_i/s_i ratio of item x is 2 and item y is 1. Thus, the Greedy algorithm will always pick item x regardless of how large B is and regardless how small ϵ is. So as B gets larger and/or ϵ gets smaller, the approximation ratio of the algorithm will increase, thus showing that the approximation ratio of Greedy is not bounded by any constant. $\qquad\square$

b. **Theorem 1b** The Modified Greedy algorithm achieves approximation ratio 2.

Proof: Let OPT be the optimal solution that has maximum value where $v(OPT) = \sum_{i \epsilon OPT} v_i$ subject to $\sum_{i \epsilon OPT} s_i \leq B$. We first assume that the items are ordered in a non-increasing fashion according to the ratio v_i/s_i. Let's call the first item that does not fit in the knapsack using the Greedy algorithm as item m. We know that item m's v_i/s_i ratio $\geq$ items $m+1, ..., n$'s v_i/s_i ratio. Thus, if we are able to fit some fraction of item m so that it fills up to the capacity of the knapsack, that solution will be $\geq$ OPT. We can define the fraction of item m that fits into the knapsack as α where $\alpha = (B - \sum_{i=1}^{m-1} v_i)/s_m$. Thus, $OPT \leq (\sum_{i=1}^{m-1} v_i) + \alpha v_m$. Since α is some fraction ≤ 1, we can also say that:

$$OPT \leq (\sum_{i=1}^{m-1} v_i) + \alpha v_m \leq (\sum_{i=1}^{m-1} v_i) + v_m$$

From the inequality above, $\sum_{i=1}^{m-1} v_i$ or v_m must be at least $OPT/2$, showing that the Modified Greedy algorithm will always get a solution at least $OPT/2$, thus achieving an approximation ratio 2. $\qquad\square$

Problem 2

a. The lower bound for OPT is $max(max_i(p_i), \frac{\sum_{i=1}^{n} p_i}{m})$.

For $m = 2$ machines and 5 jobs with processing times 3,3,2,2,2, the LPT algorithm will schedule 3,2,2 on m_1 and 3,2 on m_2, giving a makespan of 7. For OPT, we know that a lower bound is max $= max(3, 12/2) = 6$. We can achieve OPT by scheduling 2,2,2 on one machine and 3,3 on the other machine.

For $m = 3$ machines and 7 jobs with processing times 5,5,4,4,3,3,3, the LPT algorithm will schedule 5,3,3 on m_1, 5,3 on m_2, and 4,4 on m_3, giving a makespan of 11. For OPT, the lower bound is $max(5, 27/3) = 9$. We can achieve OPT by scheduling 3,3,3 on one machine and 5,4 on each of the other two machines.

b. If $p_n > OPT/3$, then $3p_n > OPT$, showing that no machine can process more than 2 jobs or else it would be > OPT, which would contradict OPT being the optimal solution. From this, we know that $n \leq 2m$, thus the largest m jobs will first get scheduled on each of the m machines and the rest of the $n - m$ jobs will be assigned to the machine that has the least load at the point of assignment, thus showing that the LPT schedule is optimal. □

c. **Theorem 2c** LPT achieves an approximation ratio of 4/3.

   Proof: Let's define job j as the job that finishes last in the LPT schedule and t_j as the span of time from time 0 until the time that job j starts. We know that in the timespan of t_j, $m \cdot t_j$ amount of processing has been done. This amount of processing can't be more than the total amount of processing of all jobs, thus $m \cdot t_j \leq \sum_{i=1}^{n} p_i \implies t_j \leq \frac{\sum_{i=1}^{n} p_i}{m}$.

As we saw earlier in part a, $\frac{\sum_{i=1}^{n} p_i}{m}$ is essentially a lower bound for OPT, thus we can say that $t_j \leq OPT$. Then by adding p_j to t_j, we get the makespan of the machine that has scheduled job j, and we can say that $t_j + p_j \leq OPT + p_j$. What the inequality tells us is that the processing time of job j indicates how well the LPT algorithm performs. Referring to part b, if $p_j > OPT/3$, the LPT schedule is optimal. But in the worst case, if $p_j = OPT/3$, then LPT will give an $OPT + p_j = OPT + OPT/3 = 4/3OPT$ makespan, thus showing that LPT achieves an approximation ratio of 4/3. □

d. **Theorem 2d** The ratio of 4/3 of LPT is asymptotically tight as $m \to \infty$.

   Proof: We generalize the examples of part a: given m machines, we have 3 jobs that have a processing time of m, and 2 jobs for each processing time $2m-1, ..., m+1$ (if $2m-1 = m+1$, then there are only 2 jobs for both $2m-1$ and $m+1$) giving us a total

of 2m+1 jobs. With LPT scheduling, we find that every machine will get scheduled two jobs with a total processing time of $3m - 1$ with exception of the first machine that will get scheduled with 3 jobs with a total processing time of $3m - 1 + m = 4m - 1$. Thus, the makespan with LPT scheduling is $4m - 1$. For an optimal makespan OPT, we first schedule all 3 jobs with processing time m on the first machine, and then use LPT scheduling for the rest of the jobs. This will give each machine a total processing time of $3m$, thus making the makespan of $OPT = 3m$. Without loss of generality, the figure below illustrates the makespan of LPT and OPT of the example from part a when $m = 3$:
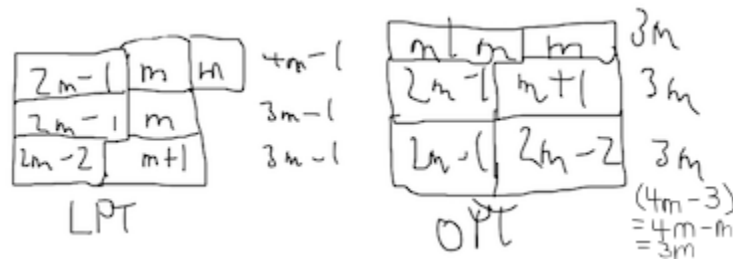

Figure 1: Makespan of LPT = 4m - 1, Makespan of OPT = 3m

Thus, we see the ratio of $LPT/OPT = (4m - 1)/3m = \frac{4m}{3m} - \frac{1}{3m} = \frac{4}{3} - \frac{1}{3m}$, showing that the approximation ratio 4/3 is asymptotically tight as $m \to \infty$. □

Problem 3

a. **Lemma 3a** If I is an independent set of G then I^k is an independent set of $G^{(k)}$.

Proof: For I to be an independent set of G, there must be no adjacent edges between any of the nodes of I. Given the definition of $E^{(k)}$ of $G^{(k)}$,

$$E^{(k)} = \{(u, v) \in N^{(k)} \times N^{(k)} \mid \exists i, j \in [k], (u_i, u_j) \in E \text{ or } (v_i, v_j) \in E \text{ or } (u_i, v_j) \in E\}$$

we can see that there are no edges between any pairs (u, v) of k-tuples of nodes that only contain nodes of I. When taking the Cartesian product of I, I^k, we create k-tuples using only those nodes from I so $I^k \subseteq N^{(k)}$ since $I \subseteq N$. Also, as explained earlier, the k-tuples of I^k do not have adjacent edges between them since they only contain nodes of I, thus showing that I^k is an independent set of $G^{(k)}$. $\qquad\square$

b. **Lemma 3b** If J is an independent set of $G^{(k)}$ then we can construct in polynomial time an independent set I of G of size at least $|J|^{1/k}$ and conclude that $\alpha(G^{(k)}) = (\alpha(G))^k$

Proof: From Lemma 3a, if I is an independent set of G, then I^k is an independent set of $G^{(k)}$. Since we have defined J to be an independent set of $G^{(k)}$, we can use Lemma 3a to say that $J = I^k$. The size of J is then equal to the size of I^k, which is $|I|^k$, thus $|J| = |I|^k \implies |I| = |J|^{1/k}$, showing that we can construct in polynomial time an independent set I of G of size at least $|J|^{1/k}$. Also, if given the maximum independent set of G, $\alpha(G)$, we can use our earlier conclusion and see that $(\alpha(G^{(k)}))^{1/k} = \alpha(G) \implies \alpha(G^{(k)}) = (\alpha(G))^k$. $\qquad\square$

c. **Lemma 3c** If the Maximum Independent Set problem can be approximated in polynomial time within some constant factor $c > 1$, then it has a PTAS.

Proof: We first define $\alpha(G)$ as the maximum independent set of G. From the PCP theorem, we can trivially say that the Maximum Independent Set problem has a 2-approximation algorithm. Let us define I as an independent set of G that agrees with the 2-approximation algorithm. This means that the size of I will be 1/2 the size of the maximum independent set of G, thus $\frac{|\alpha(G)|}{|I|} = 2$. Now, using Lemma 3b, if we take the k-th power of the graph G, we know that $\alpha(G^{(k)}) = (\alpha(G))^k$ and by Lemma 3a, I^k is an independent set of $G^{(k)}$. Thus, $\frac{|\alpha(G)|^k}{|I|^k} = (\frac{|\alpha(G)|}{|I|})^k = 2^k$ which shows that the approximation ratio grows as apply the approximation algorithm to larger and larger k-th powers of graph G. Thus, the Maximum Independent can not be approximated in polynomial time within some constant factor unless $\mathbf{P = NP}$. $\qquad\square$

Problem 4

a. **Lemma 4a** MDAS can be trivially approximated within a factor of 2.

    Proof: Suppose we have an arbitrary ordering of nodes $v_1, ..., v_n$ and we have two subsets of edges A_1 and A_2 where

$$A_1 = \{(v_i, v_j) \mid (v_i, v_j) \in A, i < j\}$$

$$A_2 = \{(v_i, v_j) \mid (v_i, v_j) \in A, i > j\}$$

By separating the edges into these two subsets, the only way for A_1 to contain a cycle is if there is an edge where $i > j$, which would be contained in the A_2 subset, and the only way for A_2 to contain a cycle is if there is an edge where $i < j$, which would be contained in the A_1 subset. Thus, neither subset will contain a cycle and at least one of the two subsets will contain at least $|A|/2$ edges since $A = A_1 + A_2$. Thus, MDAS can be trivially approximated within a factor of 2 by taking the larger of the two subsets.
$\square$

b.