CSC 2420 Spring 2019, Assignment 1

Due date: February 25 (the last day to drop a graduate course without penalty)

It is certainly preferable for you to solve the questions without consulting a published source. However, if you are using a published source then you must specify the source and you should try to improve upon the presentation of the result.

If you would like to discuss any questions with someone else that is fine BUT at the end of any collaboration you must spend at least one hour playing video games or watching two periods of Maple Leaf hockey or maybe even start reading a good novel before writing anything down.

Unless stated otherwise, all subquestions (1(i), 1(ii), 1(iii), 1(iv), etc) are worth 10 points. If you do not know how to answer a question, state "I do not know how to answer this (sub) question" and you will receive 20% (i.e. 2 of 10 points) for doing so. You can receive partial credit for any reasonable attempt to answer a question BUT no credit for arguments that make no sense.

In class I can clarify any questions you may have about this assignment.

1. Consider the makespan problem for the identical machines model with m machines. We sketched the proof that the worst case *competitive ratio* $\frac{C_{Greedy}}{C_{OPT}}$ of the natural online greedy algorithm is $\leq 2 - \frac{1}{m}$. We also gave an example showing that this ratio is "tight" (i.e. cannot be improved) for this algorithm.

   (i) Finish the proof that was sketched in Lecture 1; that is, use the fact that $C_{OPT} \geq \sum_{1 \leq i \leq n} p_i/m$ for any sequence of n input jobs and $C_{OPT} \geq p_i$ where job J_i has "load" p_i.
   Note: You can easily find the proof of this result but you should try to solve it without searching for a solution.

   (ii) Argue for $m = 2$ (resp. $m = 3$) machines that *any* (not necessarily greedy) online algorithm would have competitive ratio no better than $\frac{3}{2}$ (resp. $\frac{5}{3}$) so that the natural greedy online algorithm approximation is tight for $m = 2$ and $m = 3$ for any online algorithm.

   (iii) Consider greedy online algorithm for the makespan problem but now in the random order ROM model. Show that for any $\epsilon > 0$, there exists a sufficiently large m such that the (expected) approximation ratio $\frac{E[C_{Greedy}]}{C_{OPT}} \geq 2 - \epsilon$. Here the expectation is with respect to the uniform distribution on input arrival order.
   If you cannot prove the stated claim then prove any ratio greater than 1. Hint: generalize the nemesis sequence for the adversarial competitive ratio.

   (iv) Consider the LPT algorithm for the makespan problem. The LPT aproximation ratio is $\frac{4}{3} - \frac{1}{3m}$. The outline of the proof is as follows:

      i. Without loss of generality, the job causing the makespan is p_r, the job having minimum processing cost.

      ii. Recall the two facts about bounds for OPT that was used to prove the approximation for the online Greedy algorithm. That is, the makespan is always at least $\max_i p_i$ and at least $\frac{\sum_{i=1}^{m} T_i}{m}$ where T_i is the makespan (i.e., completion time) of machine i.

      iii. Use these two facts to show that if the stated approximation bound does not hold, then $p_r > OPT/3$.

      iv. It follows that OPT can only schedule at most 2 jobs per machine.

v. Show how to transform an OPT schedule into the LPT schedule without increasing the makespan and thereby deriving a contradiction.

Fill in the details for the above proof outline. .

2. Consider the following algorithm for the set packing problem. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ be the input instance with $S_i \subseteq \{1, 2, \ldots, m\}$.

   - Partition $\mathcal{S} = \mathcal{S}^1 \cup \mathcal{S}^2$ where $\mathcal{S}^1 = \{S_i : |S_i| \leq \sqrt{m}\}$.
   - Let R^1 be the result of running the greedy algorithm $Greedy_{wt}$ on $\mathcal{S}^1$ and let R^2 be the set having the maximum weight of sets in $\mathcal{S}^2$.
   - Return the better of R^1 and R^2.

   Show that $2\sqrt{m}$ is a bound on the approximation ratio of this set packing algorithm.

3. This question concerns local search for the exact Max-2-Sat problem where the input is a CNF formula with exactly 2 literals per clause. The goal is to maximize the number (or the total weight) of clauses that can be satisfied by some truth assignment. Khanna et al consider the locality gap achieved by oblivious and non-oblivious local search. (See week 5 lecture notes and, in particular, the proof of the $\frac{2}{3}$ locality gap for the 1-flip neighbourhood oblivious search.) As suggested by Khanna et al, one can also obtain a locality gap of $\frac{3}{4}$ by an oblivious local search algorithm that defines the neighbourhood of a solution (i.e. truth assignment) to include flipping any single variable and also flipping all variables. Modify the $\frac{2}{3}$ locality gap result to obtain the improved locality gap for this larger neighbourhood.

4. Consider the minimum spanning tree (MST) problem. Namely, lets say given a connected graph $G = (V, E)$ with weights $w : E \to \mathbb{R}$, find a spanning tree $T \subseteq E$ minimizing $\sum_{e \in T} w(e)$. The generalization to a weighted matroid $M = (U, w)$ with $w : U \to \mathbb{R}$ is to find a basis $B \subseteq U$ so as to minimize $\sum_{e \in B} w(e)$.

   (a) Why is the MST (or its generaliztion to a minimum weight basis) problem equivalent to the maximium weight spanning tree problem (resp. to maximize $\sum_{e \in B} w(e)$)?

   (b) Consider the following reverse greedy algorithm for finding a maximum weight basis in a matroid $M = (U, w)$.

   Let $m = r(M)$ where $r(M)$ is the rank of M.
   $S := U$
   While $|S| > m$
       remove from S a minimum weight element x such that
         $r(S \setminus \{x\}) = m$
   End While

   Claim: The resulting S will be a maximum weight basis.

   Restate the reverse greedy algorithm for the minimum (or maximum) spanning tree problem.

   (c) Prove that the reverse greedy algorithm will always produce a maximum weight basis in a matroid.

5. The following questions concern maximizing a monotone submodular function subject to a matroid constraint.

   (a) Prove the two facts about monotone submodular functions that appear on slide 40 of the week 5 lecture slides.

   (b) Show precisely where monotonicity is being used in the proof of the approximation ratio. Conclude that if the submodular function f was α monotone (for $\alpha \geq 1$) then the 1-exchange algorithm would provide a $\frac{1}{2\alpha}$ approximation. Here I define α monotone by the condition:

$$f(S) \leq \alpha f(T) \quad \forall S \subseteq T$$

   .

6. The following question is an alternative to the previous question and concerns matroids. As stated in class, you only have to do one of this question and the previous question. Some bonus grades for reasonable solutions to both questions.

   Let $G = (V, E, w_1, w_2)$ be a bipartite graph where the $V = U_1 \cup U_2$ and $w_i : U_i \to \mathbb{R}$ for $i = 1, 2$. We will be interested in computing a matching in G to maximize certain objectives.

   (a) We want to show the vertices in U_1 that are part of a maximum matching constitute a matroid. Prove this using the reduction of maximum bipartite matching to max flow and solving max flow by a Ford Fulkerson algorithm. If you are not familliar with Ford Fulkerson max flow algorithms, just state what you need to know in terms of the bipartite graph.

   (b) What is an algorithm to compute a matching M that will maximize the sum of the weights of vertices in U_1 that occur in the matching M.

   (c) Consider the modification of the above objective so that now you want a matching M of size at most $k \leq |U_1|$ that maximizes the weights of vertices in U_1. Hint: In general, the intersection of two matroids is not a matroid. Can you think of a general condition under which the interesection of two matroids is a matroid?

7. Consider the problem of online binary classification in the realizable case. Let $\mathcal{X} = \{0, 1\}^n$ and $\mathcal{Y} = \{0, 1\}$. Describe a hypothesis class $\mathcal{H}$ where the halving algorithm is not optimal, that is, where you would get a better worst-case mistake bound by not always going with the majority vote of the available hypotheses.

   Hint: The issue is that it's possible for the size of class $\mathcal{H}$ to not be a good indicator of how hard it is to learn.

   Note: it's OK if your hypothesis class is a bit contrived.