Problem Set 3, Answers

Kevin Lacker

June 1, 2020

Problem 1. This is a four-part problem.

1. Let L be the set of leaves of the minimum-height protocol tree for f that give output 1. For each $l \in L$, let M_l be the matrix which is 1 on the input pairs whose path through the protocol tree ends at the lead l. By the definition of a protocol tree, M_l can also be defined by a set X and Y with:
$$M_l[i,j] = 1 \iff i \in X, j \in Y \tag{1}$$
Thus, there are two possible rows in M_l, either all zeros or 1 precisely on indices in Y. Thus, the rank of M_l is 1, regardless of what field the rank is taken over.

   Note that:
$$M_f = \sum_{l \in L} M_l \tag{2}$$
Since rank is subadditive, we can conclude:
$$rank(M_f) \leq \sum_{l \in L} rank(M_l) = |L| \leq 2^{D(f)} \tag{3}$$
The last inequality holds since L is a subset of leaves of a tree of height $D(f)$. We can then conclude
$$log_2(rank(M_f)) \leq D(f) \tag{4}$$

2. There are 2^n rows of M_f. Consider them as vectors over the field $\mathbb{F}_2$, and say the rows span a subspace of dimension d, so $rank(M_f) = d$. Since the field has size 2, this subspace has exactly 2^d elements in it. Since the rows of M_f are 2^n distinct vectors in this subspace of size 2^d, we have $2^n \leq 2^d$. So $log_2(n) \leq log_s(d)$ and by the conclusion of the previous problem, $\log_2(n) \leq D(f)$.

3. TODO

4. TODO

Problem 2. Our randomized protocol for GT will use the randomized protocol for EQ, to do a binary search to find the first bit in which the two bit strings disagree. The algorithm works like:

- Find the middle of our bit string.
- Use the EQ protocol to determine whether the bits to the left of the midpoint are the same in the two strings.
- If they are the same, recurse on the right half of the string. If they are not the same, recurse on the left half of the string.
- Once we have found the first bit that disagrees, in constant communication we can figure out which of the strings is greater.

We have $O(logn)$ steps, and each step introduces a constant error. We need to reduce this error to $O(1/log(n))$ so that it is constant over the whole algorithm, so we need to repeat the protocol at each steg $O(log(log(n)))$ times. This gives a total communication complexity of $R^{pub}(GT) \leq O(log(n)log(log(n)))$.

Newman's theorem states that this can be converted into a private coin protocol with an additive log penalty, but this complexity is larger than the log penalty already, so the same asymptotic bound holds for $R(GT)$ as for $R^{pub}(GT)$.

Problem 3. Assuming f has a protocol tree T with l leaves, our task is to construct another protocol tree for f that has depth $O(l)$.

First, let's find a subtree of T that contains approximately half of its nodes. Start at the root and at each step recurse toward the larger subtree. The size of the subtree can drop by a factor of at most $1/2$ in each iteration. It starts at l and ends at 1. Therefore, it must pass through the range of $l/3$ to $2l/3$ at some point. We pick the subtree in that range and call it U.

So T can be seen as the combination of two trees, U and $T-U$. Each of these trees is at most $2/3$ the size of T. Each node in T corresponds to a rectangle, so there are two sets X and Y such that inputs (x, y) are determined by T iff $x \in X$ and $y \in Y$.

We can also use recursion to find two balanced trees, B_U and B_{T-U}, which are protocol trees for U and $T-U$ respectively, with depth is less than $c \cdot log(2l/3)$ for some constant c.

Now, we can construct a new protocol tree for f.

- Determine whether the inputs are in the (X, Y) rectangle. This just takes two bits of communication, one from each player whether their input is in the rectangle.
- If they are in the rectangle, use B_U.
- If they are not in the rectangle, use B_{T-U}.

The total depth of this tree is $2 + c \cdot log(2l/3)$, which for large enough c is less than $c \cdot log(l)$. So there is a protocol tree for f with $O(log(l))$ depth.

Problem 4.