Princeton Univ. F'20 COS 521: Advanced Algorithm Design

Lecture 22: Communication Complexity

Lecturer: *Sahil Singla* Last updated: November 17, 2020

Lectures Notes sourced from Chapters 4 and 7 of "Communication Complexity for Algorithm Designers" cited at end. I strongly recommend referring to the original source for details - these are just posted to recall what was covered in class.

1 Communication Complexity

In this class we'll discuss communication problems. That is, we'll imagine problems where no single party holds the entire input and we'll want to know how much communication the parties need to possibly solve the problem. To be clear, we'll forget about computational complexity and only focus on the communication requirements.

Definition 1. *A two-party communication problem consists of a boolean function* $f : \{0,1\}^a \times \{0,1\}^b \to \{0,1\}$ *(for this entire lecture, let* $a = b = n$*). Alice is given some input* $A \in \{0,1\}^a$ *(but doesn't know anything about* B*), and Bob is given some input* $B \in \{0,1\}^b$ *(but doesn't know anything about* A*). Their goal is to compute* $f(A, B)$*.*

Definition 2. *A* deterministic communication protocol *for* f *specifies for Alice, as a function of her input* A *and all messages sent so far* $a_1, b_1, \ldots, a_k, b_k$*, what message* a_{k+1} *to send next. It also specifies for Bob, as a function of his input* B *and all messages sent so far what message* b_{k+1} *to send next. We'll be interested in the* communication complexity *of a protocol: the maximum number of bits ever sent in total between Alice and Bob.*

Example 1 (Equality)**.** $f(A, B) = 1$ *if and only if* $A = B$*. One protocol to solve this is the following: Alice and Bob both output their entire string in the first round, and declare* 1 *if and only if their outputs match. A slightly more clever protocol might have Alice and Bob each output the first bit of their string and see if they match. If so, continue. If not, output* 0*. If they make it all the way to the end without stopping, output* 1*.*

Example 2 (Disjointness)**.** *For this problem, it will make more sense to think of* A *as a subset of* $[n]$ *(the coordinates which are* 1*), and* B *as a subset of* $[n]$*. Then consider* $f(A, B) = 1$ *if and only if* $A \cap B = \emptyset$*. One protocol to solve this is the following: Alice and Bob both output their entire set, and declare* 1 *if and only if their outputs match.*

Both examples are core problems for communication complexity. Interestingly, the natural procedures we posed seem to basically communicate the entire input in the worst case. It turns out that this is because the protocols are (almost) optimal (among deterministic protocols). We'll prove this using a generic type of argument, called a *fooling set argument*. The main idea is the following: Imagine for instance that the complete transcript (that is, all messages sent by Alice and all messages sent by Bob, in order) are identical for inputs (A_1, B_1) and (A_2, B_2). Then we claim that the transcripts must also be identical for (A_1, B_2) and (A_2, B_1).

Observation 1 (Rectangles)**.** *Let $T(A, B)$ denote the complete transcript (all messages sent, in order, and their final answers) produced by Alice and Bob using a communication protocol on input (A, B). Then if $T(A_1, B_1) = T(A_2, B_2)$, we also have $T(A_1, B_1) = T(A_2, B_1) = T(A_1, B_2) = T(A_2, B_2)$. Therefore, the protocol gives the same answer on all four input pairs.*

Proof. Assume for contradiction that this were not the case, and w.l.o.g. say that the different transcript is $T(A_1, B_2) \neq T(A_2, B_2)$. Because the protocol is deterministic, and Bob has the same input in both instances, Bob will continue outputting the same message every round until Alice sends a different message. Therefore, in order for $T(A_1, B_2)$ to possibly not equal $T(A_2, B_2)$, it must be the case that Alice sends the first different message. However, because $T(A_1, B_1) = T(A_2, B_2)$, we also have $T(A_1, B_2) \neq T(A_1, B_1)$. By exactly the same reasoning, Alice will only send a different message if Bob sends a different message first. Since it is not possible for both Alice and Bob to wait for the other to send a different message, one of the non-equalities must be violated. So all four transcripts must be the same. □

The above is called a "rectangle argument" because if we were to draw a matrix where the rows are Alice's input and columns are Bob's, it means that entries that share the same transcript must form a rectangle (okay, not exactly, but for every transcript T, there exists a relabeling of the rows/columns so that the pairs that induce this transcript are a rectangle; the formal name is "combinatorial rectangle"). It seems quite simple but is surprisingly powerful because if protocols use limited communication, it must be the case that multiple inputs have the same transcript, and the rectangle argument lets us conclude that other inputs necessarily use the same transcript as well.

Definition 3. *A* fooling set *for the function f is a set $F \subseteq \{0,1\}^n \times \{0,1\}^n$ such that:*

- *$f(A, B) = 1$ for all $(A, B) \in F$.*
- *For every $(A_1, B_1), (A_2, B_2) \in F$, either $f(A_1, B_2) \neq 1$ or $f(A_2, B_1) \neq 1$ (or both).*

Proposition 1. *Let F be a fooling set for f. Then the deterministic communication complexity of f is at least $\log_2(|F|)$.*

Proof. Assume for contradiction that there exists a deterministic protocol with communication $< \log_2(|F|)$. Then by the pigeonhole principle, there exists some $(A_1, B_1), (A_2, B_2) \in F$ such that $T(A_1, B_1) = T(A_2, B_2)$. By the rectangle argument, this necessarily implies $T(A_1, B_2) = T(A_2, B_1) = T(A_1, B_1)$ as well. This means that if the protocol is correct, we must have $f(A_1, B_2) = f(A_2, B_1) = 1$ as well.

However, this directly contradicts the second property of fooling sets. □

So if a function admits a large fooling set, it is provably impossible to have a short deterministic communication protocol. The remaining work is just to show that both Equality and Disjointness admit large fooling sets.

Proposition 2. *Equality admits a fooling set of size 2^n.*

Proof. Let $F = (X, X)$, for all $X \subseteq \{0,1\}^n$. Then $f(A, B) = 1$ for all $(A, B) \in F$. But $F(A_1, B_2) = 0$ for all $A_1 \neq B_2$. So the second property of the fooling set is satisfied. □

Notice that after unraveling all the definitions, the proof is essentially saying the following: if you had a deterministic protocol that used $< n$ bits, there are two inputs of the form (X, X), (Y, Y) that are treated exactly the same. This protocol must therefore also treat (X, Y) and (Y, X) the same as (X, X), which Equality treats differently. That is, the protocol is "fooled" into thinking the input is (X, X) or (Y, Y) when it really could be (X, Y) or (Y, X), and this matters.

Proposition 3. *Disjointness admits a fooling set of size* 2^n.

Proof. Let $F = (X, \bar{X})$, for all $X \subseteq [n]$. Then $f(A, B) = 1$ for all $(A, B) \in F$. But now consider any $(X, \bar{X}), (Y, \bar{Y}) \in F$. It cannot be the case that $X \cap \bar{Y} = \emptyset$ **and** $Y \cap \bar{X} = \emptyset$ unless $Y = X$. The former implies that $X \subseteq Y$ and the latter implies that $Y \subseteq X$. Therefore, either $f(X, \bar{Y}) = 0$ or $f(Y, \bar{X}) = 0$. □

So both protocols necessarily require communication n to solve deterministically. It turns out that both problems are also hard to solve *nondeterministically*. Disjointness is also hard to solve with a randomized protocol, but Equality can be solved with high probability with very low randomized communication (see readings for definitions).

2 Multi-party communication

We'll also talk about multi-party communication instead of just two parties - this will be relevant when figuring out how good combinatorial auctions we should possibly hope for. Here, there's again a boolean function f but it takes as input $X_1, \ldots, X_n$ (for now, all inputs are k bits, or subsets of $[k]$). Various input models are studied, but we'll only talk about the *number-in-hand* model, where player i knows only X_i and nothing else.

We'll also want a canonical hard problem to start with, multi-disjointness. It also introduces a new concept, called a *promise problem*.

Example 3 (Multi-Disjointness)**.** $f(X_1, \ldots, X_n) = 1$ *iff* $X_i \cap X_j = \emptyset$ *for all* $i \neq j$. *Moreover, we'll be interested in the* promise problem*, where the protocol is promised that one of the following is true about the input:*

- $X_i \cap X_j = \emptyset$ *for all* $i \neq j$.
- $\cap_i X_i \neq \emptyset$ *(there exists an* ℓ *in all* X_i*).*

More clearly, we say that a communication protocol solves Multi-Disjointness if it outputs 1 *whenever* $X_i \cap X_j = \emptyset$ *for all* $i \neq j$, 0 *whenveer* $\cap_i X_i \neq \emptyset$, *and is allowed to behave arbitrarily if neither condition holds.*

Theorem 4. *Every deterministic protocol for multi-disjointness has communication complexity* $\Omega(k/n)$.

Proof. First, observe that an analog of the rectangles argument holds in multi-party communication as well: if $T(X_1, \ldots, X_n) = T(Y_1, \ldots, Y_n)$, then $T(Z_1, \ldots, Z_n) = T(X_1, \ldots, X_n)$ whenever $Z_i \in \{X_i, Y_i\}$. The proof is the same: if not, there must be some player who is the first to send a different message. But their input matches their input in one of the two cases $(X_1, \ldots, X_n), (Y_1, \ldots, Y_n)$, so this is a contradiction.

The fooling set argument also extends, but we'll use the same concepts slightly differently below. We'll first show that no protocol can use the same 1 transcript for more than n^k inputs. Consider some set F of inputs that share the same transcript, and assume for contradiction that there exists an ℓ such that for all i, there exists an input $(X_1, \ldots, X_n) \in F$ with $\ell \in X_i$. Then by the rectangle argument, this same transcript is used for the input $(Y_1, \ldots, Y_n)$, where $\ell \in Y_i$ for all i, contradicting the correctness of the protocol. This means that for every ℓ, there must exist a player who does not have element ℓ in any of their inputs in F. So now we can count such inputs: for every such input, define $p(\ell)$ to be the lowest-indexed player who never has item ℓ. Then every input in F has $\ell \notin X_{p(\ell)}$. There are $n-1$ other players who all may or may not be interested in item ℓ, but all sets are pairwise disjoint, so at most one player can have $\ell \in X_i$. So there are n choices (one of the $n-1$ players, or none at all) done independently over k items, and there are at most n^k instances covered by a single transcript.

Finally, we just want to show that there are $(n+1)^k$ instances where the protocol must output 1, meaning that there are at least $\log_2((1+1/n)^k)$ different transcripts. To count the number of instances where the protocol must output 1, observe that each such instance can be selected by choosing, for each item, a player (or none at all) who has that item in X_i. There are $n+1$ choices, independently for all items, so this is $(n+1)^k$.

So immediately we conclude a lower bound of $n \log_2(1+1/n) = \Theta(k/n)$. □

The $\Omega(k/n)$ hardness also holds against randomized communication and is known to be tight. See references for more.

Finally, we want to use the above hardness to conclude a lower bound on the communication necessary to get very good approximations for combinatorial auctions (essentially showing that the algorithms we saw in the previous two lectures are the best possible unless we make some assumptions about the valuation functions).

Example 4 (Welfare Maximization)**.** *Instead of thinking of the input the players as sets, think of these bitstrings as defining a valuation function for each bidder. Then Welfare-Maximization is the following promise problem:*

- *If for all partitions of the items, the welfare is at most* 1*, the protocol must output* 1*.*
- *If there exists a partition of the items for which the welfare is at least* n*, the protocol must output* 0*.*
- *Otherwise, the protocol can behave arbitrarily.*

Theorem 5. *The communication complexity of welfare maximization is* $2^{\Omega(m/n^2)}$*. Note that this implies that exponential communication among the bidders is necessary in order to guarantee that an allocation is within a factor of* n *of optimal when* $n = m^{1/2-\varepsilon}$*.*

Proof. First, assume that we have the following construction of partitions: $\{T_1^\ell, \ldots, T_n^\ell\}_{\ell \in [k]}$, where for all ℓ, $T_i^\ell \cap T_j^\ell = \emptyset$, but for all $\ell \neq \ell'$, $T_j^\ell \cap T_i^{\ell'} \neq \emptyset$. That is, we can assign each bidder T_i^ℓ for the same ℓ, but not mix and match across different ℓs.

Now, consider the following class of valuations, called *multi-minded*, or *coverage* valuations: bidder i has some subset $X_i \subseteq [k]$, and $v_i(S) = 1$ if and only if $T_i^\ell \subseteq S$ for some

$\ell \in X_i$ (and $v_i(S) = 0$ otherwise). That is, bidder i "likes" all the sets T_i^ℓ, for $\ell \in X_i$, and gets value 1 as long as they get some set they like.

Observe that if there exists a single $\ell \in \cap_i X_i$, then it is possible to achieve welfare n: simply give each bidder set T_i^ℓ for the $\ell \in \cap_i X_i$. Also, observe that if it is possible for both bidder i and bidder j to get value 1, then the sets that they receive and are interested in must be disjoint, and therefore must have the same ℓ. So if for all i, j, $X_i \cap X_j = \emptyset$, then there is no way for both bidder i and bidder j to get non-zero welfare, and the maximum possible welfare is 1. So by a direct reduction to MultiDisjointness, the communication complexity of determining whether the welfare is $\geq n$ or ≤ 1 is at least $\Omega(k/n)$.

Now, we just need to figure out how big of a k we can have while still guaranteeing a construction of the following form. Consider k random partitions of $[m]$. That is, each item is independently and uniformly awarded to a bidder, and T_i^ℓ is the set of items awarded to bidder i in partition ℓ. Clearly, this satisfies the first property: for all ℓ, the T_i^ℓs indeed form a partition, by definition. We just need to make sure we don't take too many random partitions so that we have bad luck and some $T_i^\ell \cap T_j^{\ell'} = \emptyset$.

So consider a fixed $T_i^\ell, T_j^{\ell'}$. What is the probability that they don't intersect? This is the probability that for each item x, it is either not put into T_i^ℓ or not put into $T_j^{\ell'}$. This happens with probability $1 - 1/n^2$. So the probability that this happens for every item is exactly $(1 - 1/n^2)^m \approx e^{-m/n^2}$. Now just take a union bound over all $(kn)^2$ sets, and we get that the probability that any two sets happen to be disjoint by bad luck is $\approx k^2 n^2 e^{-m/n^2}$. Therefore, we can take $k = 2^{\Omega(m/n^2)}$ and get a non-zero probability of success (even if we can't find this instance efficiently, it exists, so we can use that for the construction and proof).

So to wrap up: we need a construction of the form at the beginning of the proof for large k. The argument above provides a construction for $k = 2^{\Omega(m/n^2)}$, which completes the proof. □

Quick side note: the proof approach above where we showed that a randomized procedure succeeds in providing a desired construction with non-zero probability is called the *probabilistic method*. It's very useful to show combinatorial structures exist when you only care about existence and not about actually finding them (although there's a ton of work on "constructive" probabilistic method arguments too, not discussed in this class).

Bibliography

1. Communication Complexity (for Algorithm Designers). Tim Roughgarden, 2015. http://theory.stanford.edu/∼tim/w15/l/w15.pdf