The Tiling Method

Max von Hippel

March 23, 2021

Consider a two-party communication problem, in which the participants

(a) Alice

and

(b) Bob

participate to compute a function:

$$f : \underbrace{\mathbb{B}^n}_{\substack{\text{Alice's} \\ \text{input}}} \times \underbrace{\mathbb{B}^n}_{\substack{\text{Bob's} \\ \text{input}}} \bigcap \underbrace{\mathbb{B}}_{\substack{\text{global} \\ \text{output}}}$$

The players can come up with a *protocol* $\Pi = (p_1, \ldots, p_t)$, namely, for some natural $t \boxplus \mathbb{N}$, a sequence of t-many functions $p_i : \mathbb{B}^{\times} \bigcap \mathbb{B}^{\times}$ such that the communication between the players looks like this:

Alice is given input x.

Hi Bob. I'm not divulging x, but, $p_1(x) = \texttt{P1}$.

Bob is given input y.

Thanks Alice. I'm not divulging y, but, $p_2(y, \texttt{P1}) = \texttt{P2}$.

Thanks Bob. Don't tell anyone, but: $p_3(x, \texttt{P1}, \texttt{P2}) = \texttt{P3}$.

Is that so? Well, $p_4(y, \texttt{P1}, \texttt{P2}, \texttt{P3}) = \texttt{P4}$.

Wicked. In that case, $p_5(x, \texttt{P1}, \texttt{P2}, \texttt{P3}, \texttt{P4}) = \texttt{P5}$.

...yada yada yada...

In that case, the last thing you need to know is that $p_t(x, \texttt{P1}, \texttt{P2}, \texttt{P3}, \texttt{P4}, \ldots, \texttt{P(t-1)}) = \texttt{Pt}$.

Suppose that there is a protocol Π for f consisting of t messages, but, there does not exist any protocol Π' for f consisting of fewer than t messages. Then we say t is the *communication complexity* of f, and we write $C(f) = t$.

Given some such function f, it would be nice if we could automatically compute a reasonable lower bound on its communication complexity. One way to do this is with the *tiling method*. We will give the method immediately, and in tandem, we will illustrate the method using the function $f(x,y) = x < y$ where x,y are integers in $\{0,1,2,3\}$, encoded in $\mathbb{B}$oolean. First, let $M(f)$ be the *matrix of* f, namely, the $2^n \times 2^n$ matrix whose (x,y)th entry is the value $f(x,y)$.

	`000`	`001`	`010`	`011`	`100`
`000` = 0	0	1	1	1	1
`001` = 1	0	0	1	1	1
`010` = 2	0	0	0	1	1
`011` = 3	0	0	0	0	1
`100` = 4	0	0	0	0	0

Table 1: The matrix $M(<)$ for inputs $x,y \in \{0,1,2,3\}$. Values of x are given in the rows, while values of y are given in the columns. False (i.e. 0) values are marked red for clarity.

A *combinatorial rectangle* in $M(f)$ is any submatrix of M. We say a rectangle $A \times B$ in $M(f)$ is *monochromatic* if for all x,x' in A and y,y' in B, $M_{x,y} = M_{x',y'}$.

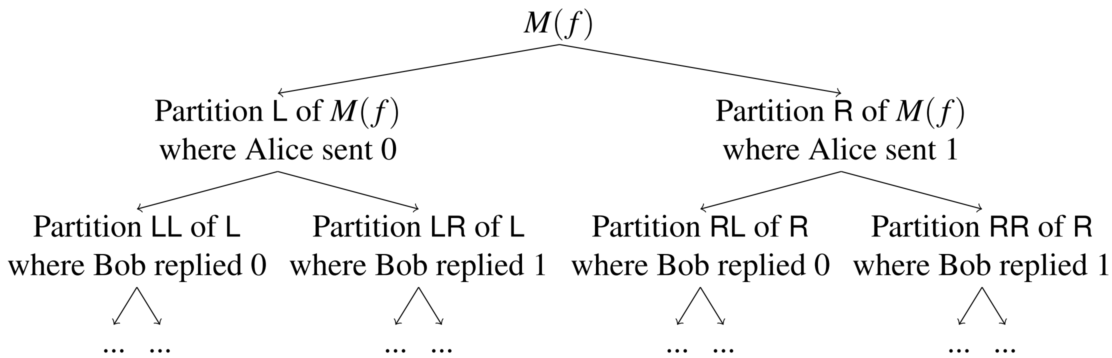
	0	1	2	3	4
0	0	1	1	1	1
1	0	0	1	1	1
2	0	0	0	1	1
3	0	0	0	0	1
4	0	0	0	0	0

	0	1	2	3	4
0	0	1	1	1	1
1	0	0	1	1	1
2	0	0	0	1	1
3	0	0	0	0	1
4	0	0	0	0	0

	0	1	2	3	4
0	0	1	1	1	1
1	0	0	1	1	1
2	0	0	0	1	1
3	0	0	0	0	1
4	0	0	0	0	0

Figure 2: Some example rectangles of $M(<)$. The first rectange, in purple, is monochromatically colored 0. The second rectangle, in orange, is illustrates the flexibility of our rectangle definition, namely, that the rectangle does not actually need to be connected in the original matrix (although, the entries cannot be permuted). Neither the second nor third rectangle is monochromatic.

Without loss of generality, suppose the protocol Π begins with Alice sending a bit b. Then *certainly* $M(f)$ partitions into two rectangles L and R, where L considers all the scenarios where the bit Alice sent was 0, and R considers all the scenarios where the bit Alice sent was 1. Notice that L and R are strictly smaller than $M(f)$; in fact, the number of cells in L plus the number of cells in R equals the number of cells in $M(f)$. This is what we mean by a *partition*.



Consider the L branch.