Streaming Communication Protocols*

Lucas Boczkowski†, Iordanis Kerenidis‡, and Frédéric Magniez§

CNRS, IRIF, Univ Paris Diderot, Paris, France.

Abstract

We define the Streaming Communication model that combines the main aspects of communication complexity and streaming. We consider two agents that want to compute some function that depends on inputs that are distributed to each agent. The inputs arrive as data streams and each agent has a bounded memory. Agents are allowed to communicate with each other and also update their memory based on the input bit they read and the previous message they received.

We provide tight tradeoffs between the necessary resources, i.e. communication and memory, for some of the canonical problems from communication complexity by proving a strong general lower bound technique. Second, we analyze the Approximate Matching problem and show that the complexity of this problem (i.e. the achievable approximation ratio) in the one-way variant of our model is strictly different both from the streaming complexity and the one-way communication complexity thereof.

*This work has been partially supported by the ERC project QCC and the French ANR Blanc project RDAM

†E-mail: lucas.boczkowski@liafa.univ-paris-diderot.fr

‡E-mail: iordanis.kerenidis@liafa.univ-paris-diderot.fr

§E-mail: frederic.magniez@cnrs.fr

1 Introduction

In the last decade we have witnessed a big shift in the way data is produced and computation is performed. First, we now have to deal with enormous amounts of data that we cannot even store in memory (internet traffic, CERN experiments, space expeditions). Second, computations do not happen in a single processor or machine, but with multi-core processors and multiple machines in cloud architectures. All these real-world changes necessitate that we revisit and extend our models and tools for studying the efficiency and hardness of computational problems.

Imagine the following situation : some input is spread among two or more agents. The agents want to compute some function f which depends on everybody's input. This is an archetypal problem of Communication Complexity (CC) [25], which offers a way to estimate the number of bits that need to be exchanged, under various settings, in order to achieve that goal. There are many different CC models, depending whether the agents can speak directly between them or through a referee, and whether they can use multiple rounds of communication or just a single one. Communication complexity has found a variety of applications both in networks and distributed computing but also in other areas of theoretical computer science, including, circuit lower bounds, fomulae size, VLSI design, etc. All these communication models however have a feature in common. They assume the agents are computationally unbounded and that the input is delivered all at once. In a distributed context as that of sensor networks, not only are there several computing agents, the input might not even be given all at once. The *streaming model* [1] has been defined precisely to capture the fact that the input of an agent is so big it cannot be stored or read several times. Instead it comes bit by bit. Some function of the stream needs to be computed, but the available space is not big enough to store the entire input. The streaming model has been extensively studied in recent years with a plethora of interesting upper and lower bounds on the necessary memory to solve specific streaming problems [23]. More recently, the turnstile model has received a lot of attention. In this model, streams are made of both insertions and deletions, and the function to be computed depends on the remaining elements (and eventually their respective frequencies). Indeed, any streaming algorithm in the turnstile model can be turned into an algorithm based solely on the updates of linear sketches [20].

The Streaming Communication model. We would like to combine the two above mentioned models to include both that inputs are distributed among different agents and also are coming as streams at each agent. Each agent is given a bounded memory to store what she sees. We refer to this extension as the *Streaming Communication (SC) model*. Even though communication arguments have often been invoked in proving lower bounds for regular streaming models, as in the seminal work of [1, 3], this model has not been rigorously defined previously, in spite of its theoretical appeal and relevance for actual communication networks. More formally, in the SC model consider two agents, Alice and Bob, want to compute some function f that depends on inputs (x, y) that are respectively distributed to each agent, x to Alice and y to Bob. Both inputs arrive as data streams and each agent has a bounded memory of a given size S. Agents may or may not speak every time they receive a bit. They can also update their memory based on the previous bit they read, the previous message they received and of course the actual content of their memory.

Additionally to the memory size S of Alice and Bob, the other relevant parameters we consider are the number R of communication rounds and the number T of bits in the full transcript (the concatenation of all messages). In the *one-way* SC model, there is only a single message from Alice to Bob at the end of the streams. Observe that we do not bound the size of each message, since we show that those can always be assumed to be of at most $S + 1$ bits (**Proposition 2.5**).

Related models Before we present our results in the streaming communication model, we review some related works in the communication and streaming models. As explained previously, our goal is to provide rigorous tradeoffs between the two resources: memory storage and communication between the players, in a model where inputs are coming as streams.

The most relevant works to ours are the papers [10, 11]. There, two parties receive two streams and at the end of the streams each party sends their workspace to a referee which uses both workspaces to compute some function of the union of the two streams. In this model, we can also see elements both from streaming algorithms and communication complexity, albeit of the restricted form of simultaneous message passing. Here we provide a more general framework for communication and we look at a much wider variety of problems and protocols.

One of the powerful and often used techniques in streaming algorithms is linear sketches, which naturally provide very simple SC protocols even in the one-way setting, by just combining the linear sketches at the end of the protocol. In fact, in the turnstile model, when streams are made of sequences of insertions and deletions and the function to be computed is a function of the remaining elements (and eventually their respective frequencies), any streaming algorithm can be turned into an algorithm based solely on the updates of linear sketches [20]. Hence, here we focus on other stream models, such as the one of insertion only, which are more challenging in the context of SC protocols.

In the distributed functional monitoring setup initially proposed by [9], k servers receiving a stream have to allow a coordinator to continuously monitor a given quantity. Several works have expanded the results on this model (see e.g. [8, 13, 21, 24]). These all focus on communication and do not consider *both* resources, memory storage and communication simultaneously. They can be viewed as extending [10, 11] with greater number of players. The communication model however is still restricted to a referee scenario.

Another line of work studies bounded-memory versions of communication [5, 7, 19]. Several models have been proposed that share the same structure. The input is given all at once, but the players only have bounded space to store the conversation while further restrictions can be placed on the algorithms used by the players, for example to be straight line programs [19], or branching programs [5].

Our results. Our first results, detailed in Section 2.4, show connections between our new model and its two parent models, Communication Complexity and Streaming. We show that the total transcript size T and the product RS (of the number of rounds and the memory size) are both lower bounded by the communication complexity $C(f)$ of the function f we wish to compute, up to logarithmic factors (**Proposition 2.6**). Those factors come from an inherent notion of clock in our SC model.

The comparison with streaming algorithms is more subtle. Since the i-th bit of Alice's input arrives at the same time as the i-th bit of Bob's input, the correct comparison is with a single streaming model where the stream is the one we get by interleaving Alice's and Bob's stream. We denote by $\mathcal{S}^{\textbf{int}}(f)$ the memory required by a streaming algorithm for computing a function f, when the two streams are interleaved in a single stream. We first observe than interleaving streams instead of concatenating them can lead to an exponential blow up (**Theorem 2.8**). Then we show that $\mathcal{S}^{\textbf{int}}(f)$ is a lower bound on twice the memory size S of players (**Proposition 2.9**).

Then it is natural to ask if there is always a polynomial relation between, on one hand, the parameters of a protocol in the SC model (S, R and T), and on the other hand, the communication complexity $C(f)$ (randomized or deterministic) of the function when the input is all given in the beginning and the memory $\mathcal{S}^{\textbf{int}}(f)$ necessary in the single stream model. We show that this is not true in general by providing an example for which $\mathcal{S}^{\textbf{int}}(f) = C(f) = O(\log n)$ but $R \cdot S = \Omega(n)$, when $S = \Omega(\log n)$ (**Theorem 2.10**). This implies that the SC complexity of a function f may not be immediately derived neither by its communication

complexity nor by its streaming complexity. This is one of the main reasons why our model is interesting and necessitates novel techniques for its study.

The first of our two main results is a general technique for proving tradeoffs between memory and communication in our model. The smaller the memory, the more frequent communication has to be. For instance, one expects that for functions whose communication complexity is n, i.e. where all bits are necessary, players with a memory of size S *have to* speak at least every S rounds (either deterministic or randomized), since if they remain silent for more than S rounds, they start to lose information about their input. More precisely, assume any function f that can be written as $f(x,y) = G(g_1(x^1,y^1), g_2(x^2,y^2), \ldots, g_L(x^L,y^L))$, where G is a function satisfying some assumptions. Then, any randomized protocol computing f must have $R \cdot S = \Omega(\sum_{\ell \in L} C(g_\ell))$ (**Theorem 3.3**). We can apply our theorem to many canonical communication functions, including $IP_n, DISJ_n$ or $TRIBES_n$, and show that any protocol satisfies $R \cdot S = \Omega(n)$ (**Theorem 3.4**).

In Section 4, we study problems arising in the context of graph streaming. We work in the insert only model, meaning that the graph is presented as a stream of its edges in an arbitrary order. Indeed, as opposed to the turnstile model, where any algorithm can be turned into a linear sketches based on [20], the situation is much more intriguing for problems where linear sketches are not used. In particular, in the context of streaming algorithms for graph problems, *Approximate Matching* has been extensively studied, and its streaming complexity is still unknown. Given a stream of edges (in an arbitrary order) of an n-vertex graph G and some space restriction, the goal is to output a collection of edges from G forming a matching, as big as possible in G. The matching size estimation is a different and somehow easier problem [16].
It is known that any streaming algorithm for Approximate Matching using $\widetilde{O}(n)$ memory cannot achieve a ratio better than $\frac{e}{e-1}$ [15], whereas the best known algorithm is a simple greedy algorithm which provides a 2-approximation. In the one-way CC model, without memory constraints, it has been also showed that a $\frac{3}{2}$-approximation is the tight bound when Alice's message is restricted to $\widetilde{O}(n)$ bits [12]. Both these works use in a clever way the so-called Ruzsa-Szemerédi graphs.
We study both the general SC model and its one-way variant. Our main bounds are for the one-way variant, the weaker model combining the restrictions of both CC and streaming models: we show a lower bound of $\frac{e+1}{e-1} \approx 2.16$ for the approximation factor unless $S = n^{1+\Omega(\frac{1}{\log\log n})}$ (**Corollary 4.3**), which is strictly higher than both the single stream lower bound of $\frac{e}{e-1} \approx 1.58$ with same space constraints and the one-way communication lower bound of 1.5. We also provide a one-way SC protocol achieving an approximation ratio of 3 with the same space constraints (**Theorem 4.2**), thus leaving as an open question the optimal approximation ratio. Moreover, we show that how often the players communicate makes a big difference, namely we show how to implement the simple greedy algorithm when Alice and Bob can communicate during the protocol that provides a ratio of 2, strictly better than our lower bound for the one-way SC model. Let us emphasize that all previous lower bounds, including the ones in the turnstile models [2, 17], do not readily apply to the one-way SC model for the Approximate Matching problem. However, our main lower bound in the one-way SC model uses as a black-box the hard distributions of graph streams of [12, 15]. Therefore, further improvements in the streaming context may lead to improvements in our model. Given a hard distribution μ of graphs for the approximate matching for streaming algorithms, we show how to extend this distribution to produce a hard distribution μ_2 in our one-way SC model (**Theorem 4.7**).

2 The streaming communication model

We provide some background on communication complexity and streaming and then, we define our model and describe some initial results.

2.1 Communication Complexity

We start by reviewing some results in the usual models of communication complexity (CC), defined by Yao [25]. For more details about the communication complexity model, please refer to [18]. In the communication complexity models, generically denoted by *CC*, two players aim at computing some function which depends on their disjoint inputs, by communicating. Each player determines her message based on previous messages and her input. The goal is to minimize the total length of the protocol transcript.
In the randomized case, we will allow the players to share public randomness. Allowing for public randomness makes our lower bounds stronger, while the protocols we provide will be deterministic. We will also consider the expected, rather than maximal, length of transcripts and define the average randomized communication complexity of a function.

Definition 2.1. *For a given protocol Π, we denote by $\Pi(x, y, r)$ the transcript with inputs x, y and public randomness r. The worst case (resp. expected) communication complexity of a function f with error ε is defined as $C_\varepsilon(f) = \min_\Pi \max_{x,y} \max_r |\Pi(x, y, r)|$ and $C_\varepsilon^{avg}(f) = \min_\Pi \max_{x,y} \mathbb{E}_r(|\Pi(x, y, r)|)$, where the minimum is taken over protocols computing f with error ε, and the expectation on the second line is with respect to the randomness r used in Π.*

The following proposition relates the average and worst case randomized communication complexities.

Proposition 2.2 ([18])**.** *For any $\varepsilon, \delta > 0$, it holds that, $\delta \cdot C_{\varepsilon+\delta}(f) \leq C_\varepsilon^{avg}(f) \leq C_\varepsilon(f)$.*

Depending on the context, we denote by $C(f)$ either the deterministic communication complexity of f or the randomized complexity for a fixed $\varepsilon = 1/3$.
Some of the canonical functions studied in communication complexity are the equality problem, denoted EQ_n where the players output 1 iff their inputs $x, y \in \{0, 1\}^n$ are equal, the disjointness problem, denoted $DISJ_n$ where the goal is to check whether the n-bit strings interpreted as sets intersect or not, and the inner product problem IP_n where the players need to output the inner product of their inputs modulo 2.
The functions $DISJ_n$ and IP_n are "hard" functions for *CC*, in the sense that almost all the input must be sent even when we allow for randomization, error and expected length. The following two bounds, which we will need later, can be derived for example from [4], where the notion of information cost is used.

Theorem 2.3. *Any protocol for $DISJ_n$ or IP_n with error $1/2 - \varepsilon$ has communication complexity $\Omega(\varepsilon^2 n)$.*

2.2 Streaming algorithms

In the streaming model, the input comes as a stream to an algorithm whose task is to compute some function of the stream while using only a limited amount of memory and making a single or a few passes through the input stream. See [23] for a general introduction to the topic. If possible, the updates should also be fast. It was defined in the seminal work of [1] where the authors provided upper and lower bounds for computing some stream statistics. Since then, a plethora of results have appeared for computing statistics of the stream, as well as for graph theoretic problems. For the graph problems, we will assume that the graph is revealed to the algorithm as a stream, one edge at a time. In the more recent turnstile model, streams are made of both

insertions and deletions, and the goal is to compute some function that depends only the remaining elements (and eventually their respective frequencies). As we have said, any problem in the turnstile model can be solved via linear sketches [20].

2.3 The new model of Streaming Communication protocols

We show how to extend the original model of communication complexity to account for streaming inputs. In the Streaming Communication SC model we consider that the inputs x, y are not given all at once to the two players Alice and Bob but rather come as a stream. Moreover each player only has limited storage, S bits of memory. In the randomized case, the players also have access to a shared random bit string r which may be infinite. They may use as many coins as they like from these strings.
A *protocol* Π in the streaming communication model is specified by four functions $\Phi^A, \Phi^B, \Psi^A, \Psi^B$. Each time slot i is divided in two phases:

1. Each party receives a message from the other party (m_i^B and m_i^A resp.) and updates their memory (that was in state σ_i^A and σ_i^B resp.) according to the function Φ^A and Φ^B resp. This function also depends and the shared random string r, which is not restricted in size.
2. Messages m_{i+1}^A and m_{i+1}^B are produced using the functions Ψ^A and Ψ^B resp., that depend on the current memory states σ_{i+1}^A and σ_{i+1}^B resp., the newly read input bit, and the randomness r. The messages might be empty and they could also be arbitrarily big in principle, though we will see in Proposition 2.5 that their size can be assumed to be $S + 1$ without loss of generality.

The memory state $\sigma_{i+1}^{A/B} \in \{0,1\}^S$ and the next message $m_{i+1}^{A/B} \in \{0,1\}^*$ can be defined recursively as $\sigma_{i+1}^{A/B} := \Phi^{A/B}(m_i^{B/A}, \sigma_i^{A/B}, r)$ and $m_{i+1}^{A/B} := \Psi^{A/B}(\sigma_{i+1}^{A/B}, x_{i+1}, r)$. Moreover, we assume that the streams end with a special EOF symbol and that once the streams are finished, the players only get one last round of communication, and then they have to output something.

Definition 2.4 (SC protocols)**.** *An SC protocol* Π *uses* S *bits of memory,* R *rounds, and* T *bits when*

- *The memory size of each player is at most* S *bits;*
- *The (expected) number of time slots where either* $m_i^A \neq \emptyset$ *or* $m_i^B \neq \emptyset$ *is at most* R*;*
- *The (expected) size of all exchanged messages is at most* T *bits.*

The expectation is over the randomness of the protocol and worst-case over the inputs. An SC protocol is said to be one-way *if there is a single message from Alice to Bob after the streams have been received, and only Bob computes the function.*

Note, that our model carries an implicit notion of time due to the players reading their streams synchronously, and hence, the ability to send empty messages can be used to reduce communication [14]. However the gain is only logarithmic in the number of available time slots (see Section 2.4). We could have avoided such extra power, by defining a model where agents know when they should speak or read a bit, based on the previous messages they received and their memory content. Nevertheless, we opted for our model, as it is simpler to state and the necessary resources do not change by more than a factor logarithmic in the input size.
When we prove lower bounds or communication-memory tradeoffs, we do not consider the complexity of $\Phi^{A/B}$. These functions could be of arbitrarily high complexity. To make things simple, we assume they are the same functions for every round $i \in [n]$, but they can depend on n. This framework captures the streaming model as a special case, when the output depends on the stream of Alice only.

2.4 Properties of the SC model

Several times in our proofs, we will consider an SC protocol and use it to solve problems in the standard models of CC. It is convenient to have a bound on how big the messages $m^{A/B}$ can be.
The length of the messages $m^{A/B}$ could be very big in the SC protocol, but we now show that the SC protocol can be simulated replacing them by length $S+1$ messages.

Proposition 2.5. *In the SC model, we may always assume that the size of the messages is at most $S+1$ bits, up to redefining the transition functions $\Phi^{A/B}$.*

Proof. Consider a protocol Π with associated functions $\Phi^{A/B}, \Psi^{A/B}$. The players can exchange their S size memory and the last input symbol instead of the actual messages. Hence, it is possible to redefine functions $\Phi^{A/B}, \Psi^{A/B}$ and directly assume messages have length $\leq S+1$. The new equations with $S+1$ bit messages would read $\sigma_{i+1}^{A/B} := \Phi^{A/B}(\Psi^{B/A}(\sigma_i^{B/A}, y_i, r), \sigma_i^{A/B}, r)$ and $m_{i+1}^{A/B} := \Psi^{A/B}(\sigma_{i+1}^{A/B}, x_{i+1}, r)$. □

Any protocol in the SC model can be simulated with another protocol in the usual CC model with a small overhead. Note that due to the implicit time in the SC model, we cannot immediately conclude that the SC model is harder than the usual communication model. Nevertheless, this time issue induces only an extra logarithmic factor.

Proposition 2.6. *We can simulate any protocol Π in the SC model with parameters S, R, T with another protocol Π' in the normal communication model such that its communication cost $C(\Pi')$ is bounded as $C(\Pi') \leq T(1+2\log n)$ and $C(\Pi') \leq R(S+2\log n+1)$.*

We now compare the SC model to streaming algorithms, that is when there is a single player and a single stream. There are various ways to combine streams x and y in a single stream. Since x_i is presented to Alice at the same time as y_i to Bob, in a single player model x_i should be presented just before y_i to the player. This explains why we consider the interleaved streaming model.

Definition 2.7. *Let $\mathcal{S}^{\textbf{int}}(f)$ be the amount of memory required for a streaming algorithm to compute f where the input stream is x and y interleaved, that is to say, $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$.*

It turns out that interleaving streams instead of concatenating streams may affect the memory requirement of the function for a standard streaming algorithm by an exponential factor (see Appendix A.2).

Theorem 2.8. *There is a function f such that $\mathcal{S}^{\textbf{int}}(f) = \Omega(n)$, whereas there is a streaming algorithm to compute f with memory $O(\log n)$ when streams are concatenated.*

First let us observe that $S^{\textbf{int}}(f)$ provides a lower bound in the SC model.

Proposition 2.9. *Let f be a function. Then, any protocol in the SC model for the function f, where Alice and Bob use memories of size S, must have $2S \geq S^{\textbf{int}}(f)$.*

It is natural to ask if there is a polynomial relation bounding the parameters S, R, T of a protocol in the SC model in terms of the streaming complexity $\mathcal{S}^{\textbf{int}}(f)$ and the communication complexity $C(f)$, at least when $S = O(\mathcal{S}^{\textbf{int}}(f))$. This appears to not hold in general (See Appendix A.3).

Theorem 2.10. *There exists a function f such that $\mathcal{S}^{\textbf{int}}(f) = C(f) = O(\log n)$ but any protocol computing f in the SC model must have $R \cdot S = \Omega(n)$. This holds for $S = \Omega(\log n)$.*

3 Communication primitives

We provide a general theorem that provides tight tradeoffs both in the deterministic and randomized case for a variety of functions, including $DISJ_n, IP_n, TRIBES_n$.

3.1 A general lower bound

In this section we show a general result that gives a lower bound for a large class of functions. We will obtain the lower bounds for the usual primitives $DISJ, IP, TRIBES$ as a corollary. We treat ε as a constant. Let us start with a high level description of the result.
Assume f can be written as the composition of an outer function G and inner gadgets g_ℓ. We want to lower bound the communication needed to compute f in the SC model in terms of the communication complexity of each g_ℓ and the available memory.

Definition 3.1. *We call a function G on L variables* non trivial *if the following holds. There exists a word $a \in \{0,1\}^L$ such that for all ℓ there exists a postfix $b_\ell \in \{0,1\}^{L-\ell-1}$ such that $G(a_{\leq \ell} u b_\ell)$ depends on the bit u. More formally $G(a_{\leq \ell} 0 b_\ell) \neq G(a_{\leq \ell} 1 b_\ell)$.*

This may look as a restrictive condition. In fact, most natural functions that depend on every bit are "non trivial" in this sense. For the function $\bigoplus$, a, b can be chosen arbitrarily. The functions OR and AND are also non trivial. For instance for AND, $a = b = 1^L$ will do.
We borrow the next definition from [4] (extending it slightly).

Definition 3.2 (Block-decomposable functions.)**.** *Let $I_1, \ldots, I_L$ be an interval partition of $[n]$, which we refer to as* blocks*. For $\ell \in [L]$, let $t_\ell = |I_\ell|$ be the length of I_ℓ. Given strings $x, y \in \{0,1\}^n$, write x^ℓ (resp. y^ℓ) for the restriction of x (resp. y) to indices in block I_ℓ. We say $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is* G-decomposable *with primitives (g_ℓ), where $G : \{0,1\}^L \to \{0,1\}$ and $g_\ell : \{0,1\}^{t_\ell} \times \{0,1\}^{t_\ell} \to \{0,1\}$, if for all inputs x, y we have $f(x,y) = G(g_1(x^1, y^1), g_2(x^2, y^2), \ldots, g_L(x^L, y^L))$.*

Theorem 3.3. *Assume the function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is G-decomposable with primitives $(g_\ell)_{\ell \in L}$, and that G is non-trivial. Let $C_{\varepsilon+\delta}(g_\ell)$ be the worst-case randomized communication complexity of g_ℓ in the usual communication model. Then, any randomized protocol computing f with error ε in the SC model with S bits of memory, R expected communication rounds and T expected bits of total communication, must have*

$$R \geq \sum_{\ell \leq L} \frac{\delta C_{\varepsilon+\delta}(g_\ell) - S}{S + 2\log t_\ell + 1}, \quad T \geq \sum_{\ell \leq L} \frac{\delta C_{\varepsilon+\delta}(g_\ell) - S}{1 + 2\log t_\ell}.$$

We can get a similar bound in the deterministic case, where we use the deterministic communication complexity of the g_ℓ's. Last, if G is $\bigoplus$ we may remove δ from the above bounds, changing the complexities $C_{\varepsilon+\delta}(g_\ell)$ to $C_\varepsilon^{avg}(g_\ell)$.

3.2 Applications

Before proving Theorem 3.3, we give a few corollaries. Note that the upper bounds are trivial. Remind the function $TRIBES$, which is an AND of Set Intersections is defined by $TRIBES_n(x,y) := AND_{i \leq \sqrt{n}} \circ OR_{j \leq \sqrt{n}} (x_{ij} \bigwedge y_{ij})$.

Theorem 3.4. *Any randomized protocol in the SC model that computes the function IP_n, $DISJ_n$, or $TRIBES_n$ and uses S memory and R communication rounds must have $R \cdot S = \Omega(n)$.*

Proof. We start by the argument for $DISJ_n$. We write $DISJ_n(x,y) = AND_{\ell=1}^{n/k}\left(DISJ_\ell(x^\ell, y^\ell)\right)$, and use the previous result with $G = AND$ and $g_\ell = DISJ_k$. It follows from Theorem 2.3 that $R_{\varepsilon+\delta}(DISJ_k) = \Omega(k)$. We omit the dependency on ε and δ in the term $\Omega()$, treating these parameters as fixed constants. The number of rounds for any randomized protocol in the SC model is at least $\sum_{\ell=1}^{\frac{n}{k}} \frac{\Omega(k)-S}{S+2\log k}$. We get the result choosing $k = \Omega(S)$.
In the case of IP_n, the function $f = IP_n$ is the composition of $G = \bigoplus$ over $\frac{n}{k}$ coordinates with $g_\ell = IP$ (for each $\ell \leq \frac{n}{k}$), over k coordinates. Theorem 2.3 gives $C_\varepsilon^{avg}(g_\ell) \geq \Omega(k)$. Theorem 3.3 yields the bound, taking $k = 10S$. We omit the case of $TRIBES_n$ as it follows from a similar argument. □

4 Approximate Matching in the Streaming Communication model

The main problem we are going to consider is that of computing an approximate matching. The stream corresponds to edges (in an arbitrary order) of a bipartite graph $G = (P, Q, E)$ over vertex set P, Q, and the algorithm has to output a collection of edges which forms a matching. All edges in the output have to be in the original graph. In the vertex arrival setting, each vertex from Q arrives together with all the edges it belongs to. Our goal is to understand what is the best approximation ratio we can hope for, for a given memory (and message size).
We start by some notations. In a graph $G = (P, Q, E)$, if $U \subseteq P \cup Q$ and $V \subseteq P \cup Q$ are subsets of the vertices, we denote by $E(U, V) \subseteq E$ the edges with endpoints in U and V. We also denote by $OPT(G)$ the maximum size of a matching in G.
Observe now that when communication can occur at any step, the greedy algorithm, which is currently the best algorithm in the standard streaming model, can be implemented easily by having Alice communicate to Bob every time she adds an edge to her matching.

Proposition 4.1. *The greedy algorithm, which achieves a* 2*-approximation, can be implemented using* $n \log n$ *bits of communication and* n *rounds.*

Thus, we now focus on the one-way SC model, where the communication is restricted to happen once the streams have been fully read. In this setting, we will get different lower and upper bounds than in the streaming model. We start by a positive result.

Theorem 4.2 (Greedy matchings)**.** *If Alice sends Bob a maximal matching of her graph, then Bob can compute a* 3*-approximation. In particular a* 3*-approximation can be computed in a deterministic one-way SC protocol using* $O(n \log n)$ *memory and message size.*

Proof. Let G_1, G_2 be the respective graphs that Alice and Bob get, and let M_1, M_2 be their respective computed maximal matchings. We show that there is a matching in $M_1 \cup M_2$ of size at least $|OPT(G)|/3$. The proof goes in two steps. Let ℓ be the size of $V(M_1 \cup M_2)$. We prove first that there is a matching of size $\geq \frac{\ell}{3}$ in the graph $M_1 \cup M_2$, and then that the number of edges in $OPT(G)$ is at most ℓ,
The first part is easy. Observe that $M_1 \cup M_2$ has maximal degree 2. Then there must be a matching of size at least $\ell/3$ from Theorem 7 in [6].
For the second part, we construct an injection $OPT(G) \hookrightarrow V(M_1 \cup M_2)$. Let $e = (u_1, u_2) \in OPT(G)$. Assume without loss of generality that $e \in G_1$. Then either u_1 or u_2 is matched in M_1 (maybe both), by maximality of M_1. Map e to (one of) its matched endpoint in M_1. □

Our lower bound is obtained using a black box reduction that we develop in the following sections. It is a direct consequence of the combinaison of Theorem 4.7 and Theorem 4.5.

Corollary 4.3 (A $(e+1)/(e-1)$ lower bound)**.** *Any protocol achieving a ratio of $\frac{e+1}{e-1} - \eta \approx 2.16 - \eta$, for some constant η, in the vertex arrival setting needs communication $n^{1+\Omega(1/\log\log n)}$ where the hidden constant in the $\Omega(.)$ depends on η.*

4.1 Hard distributions for streaming algorithms

Our notion of hard distribution is tailored to capture the distributions appearing in [12, 15]. They are distributions over streams of graphs, that is over graphs and edge orderings.
We will use the following definition for constructing families of hard distributions when $n \to \infty$ and α, η are fixed. Therefore $O()$ and $o()$ notations have to be understood in that context.

Definition 4.4 (Hard distribution)**.** *A distribution μ over streams of bipartite graphs $G = (P, Q, E)$ is an* $(\alpha, n, m(n), \eta)$-hard distribution *when P and Q are sets of size n and the following holds*

1. *There is a cut $W, \overline{W}$ of vertices such that $|W \cap Q| + |\overline{W} \cap P| \leq (1 - \alpha + \eta)n$.*
2. *There is a matching M of size $(1-\eta)n$ in G that can be decomposed into $M_0 \cup M'$ such that*
   (i) $P_0 := V(M_0) \cap P$ and $Q_0 := V(M_0) \cap Q$ are of fixed size (in the support of μ) larger than $(\alpha - \eta)n$ and smaller than αn; and
   (ii) $P_0 \subseteq W$ and $Q_0 \subseteq \overline{W}$.
3. *Every streaming algorithm **Alg** with $o(m)$ bits of memory that outputs E^* with $E^* \subseteq E$ must satisfy $|E^* \cap E((W \cap P) \times (\overline{W} \cap Q))| = o(n)$ with probability $1 - o(1)$.*

In particular, a streaming algorithm with small memory can only maintain on hard distributions a small fraction of edges $E((W \cap P) \times (\overline{W} \cap Q))$ and therefore of M_0. In addition, observe that for every matching E^* and cut $(W, \overline{W})$ (see Figure 1) $|M| \leq |W \cap Q| + |\overline{W} \cap P| + E((W \cap P) \times (\overline{W} \cap Q))$. Thus edges from $E(W \cap P \times \overline{W} \cap Q)$ are also crucial for obtaining a good matching.
The existence of hard distributions is ensured by [12, 15]. The hard distributions families are not exactly presented as we present them here. The following Theorem follows from results in [15] involving more parameters, for the purpose of the construction itself. We disregard those since we use the existence of hard distribution families as a black box.

Theorem 4.5 ([15])**.** *For all $\eta > 0$ and n, there is a $(1/e, n, m(n), \eta)$-hard distribution μ over graphs of n vertices with $m(n) = n^{1+\Omega(\frac{1}{\log\log n})}$, where the notation $\Omega(.)$ hides a dependency on η.*

Sketch of proof. Specifically, point (1) of our definition follows from [15, Lemma 13], point (2) follows from [15, Claim 12], and point (3) follows from [15, Lemma 14]. □

4.2 Lifting hard distributions to streaming communication protocols

Let μ be an (α, n, m, η)-hard distribution for streaming algorithms. We will show how to extend it to a distribution μ_2 for the two party version of the approximate matching problem. At a high level, we give the players two copies of the same graph G randomly chosen according to μ, but embedded into two different but overlapping subsets of vertices (see Figure 2). The non-overlapping parts correspond to edges from $E(W \cap P \times \overline{W} \cap Q)$ (see Definition 4.4), and are therefore hard to maintain, but necessary to setup a large matching.
From now on, identify P with the set $[n]$. Set $\beta := 1 + \alpha$. Our labelings are defined over vertex set $P' \times Q' := [\beta n] \times [\beta n]$, and are encoded by injections from $[n]$ to $[\beta n]$ (where for simplicity βn is understood as an integer). Given a hard distribution μ, we define a distribution μ_2 as follows.

Definition 4.6 (The distribution μ_2)**.** *Let μ be a hard distribution, where P and Q are identified with $[n]$. Then sampling a bipartite graph over vertex set $P' \times Q' = [\beta n] \times [\beta n]$ from μ_2 is defined as follows*

- *Sample $G \sim \mu$. Let $(W, \overline{W})$ and P_0, Q_0 be the corresponding cut and sets from Definition 4.4.*
- *Sample σ, τ uniformly at random such that $\sigma(P_0) \cup \tau(P_0) = \emptyset = \sigma(Q_0) \cup \tau(Q_0)$ are disjoint, and σ, τ are equal on $P \setminus P_0$. Such injections σ, τ are called G-*compatible*.*

In addition, define $G_\sigma := (\sigma(P), \sigma(Q), E_\sigma)$, where $E_\sigma = \{(\sigma(u), \sigma(v)) \mid (u, v) \in E\}$, and G_τ similarly. Alice is given G_σ and Bob is given G_τ with the same order as under μ.

In this construction, observe that edges sent to Alice and Bob may overlap. In fact the distribution can be tweaked to make edges disjoint using a simple gadget, while preserving the same lower bound. We present this in Appendix C. We can now state our main result for the reduction.

Theorem 4.7 (Generic reduction)**.** *If there exists an (α, n, m, η)-hard distribution for approximate matching, then any protocol in the one-way SC model whose approximation ratio is $\frac{1-\alpha}{1+\alpha} - O(\eta)$ has to use $\Omega(m)$ memory.*

Proof. Define a cut for the two player instance as $W' := \sigma(W) \cup \tau(P_0)$, $\overline{W}' := \sigma(\overline{W}) \cup \tau(Q_0)$. It follows from the max-flow min-cut argument that any matching E^* has size at most $|E^*| \leq |\overline{W}' \cap Q'| + |W' \cap P'| + |E^* \cap E(W' \cap P' \times \overline{W}' \cap Q')|$.
Moreover note that by construction $|\overline{W}' \cap Q'| = |\sigma(\overline{W}) \cap Q'|$. Indeed $\tau(P_0) \cap Q' = \emptyset$. Then we can write $|\sigma(\overline{W}) \cap Q'| = |W \cap Q|$ and similarly for $|W' \cap P'|$ (see also Figure 2). It follows that

$$|\overline{W}' \cap Q'| + |\overline{W}' \cap P'| = |\overline{W} \cap Q| + |\overline{W} \cap P| \leq (1 - \alpha + \eta)\, n.$$

The set of edges the protocol outputs is included in $E^*_A \cup E^*_B$ by definition. Under the high probability event that these sets only have an overlap of $o(n)$ with the "important edges" $E(W' \cap P \times \overline{W}' \cap Q)$ (see Lemma 4.8 below) then, if E^* is the output matching by a protocol using $o(m)$ memory, then using Lemma 4.8 the matching $E^* \subseteq E^*_A \cup E^*_B$ is of size at most $(1 - \alpha + \eta)\, n + o(n) \leq (1 - \alpha + 2\eta)\, n$ (for large enough n).
On the other hand, there is a matching between $\sigma(P)$ and $\sigma(Q)$ of size $(1 - \eta)n$ and a matching of size $(\alpha - \eta)n$ between $\tau(P_0)$ and $\tau(Q_0)$ (using Point (2) in the definition of a hard distribution for approximate matching, Definition 4.4). We identified $\sigma(P) \sqcup \tau(P_0)$ with $[\beta n]$ and hence under μ_2 there is a matching of size $(\alpha - \eta)n + (1 - \eta)n = \beta n - 2\eta n$. This shows that the approximation ratio of a protocol using $o(m)$ memory is at most $\frac{\beta n - 2\eta n}{(1-\alpha+2\eta)} = \frac{1+\alpha}{1-\alpha} - O(\eta)$. □

Lemma 4.8. *Let **Alg** be a protocol for the two party case, i.e. a pair of algorithms for Alice and Bob **Alg** $= (\mathbf{Alg}_A, \mathbf{Alg}_B)$. Let E^*_A (resp. E^*_B) denote the edges $\mathbf{Alg}_A$ (resp. $\mathbf{Alg}_B$) outputs, assuming only $o(m)$ memory is used. With probability $1 - o(1)$ over the choice of σ, τ, G, or alternatively with probability $1 - o(1)$ under μ_2, it holds that*

$$|E^*_A \cap E(W' \cap P' \times \overline{W}' \cap Q')| = o(n), \quad \textit{and similarly} \quad |E^*_B \cap E(W' \cap P' \times \overline{W}' \cap Q')| = o(n).$$

Proof. The proof consists in building from $\mathbf{Alg}_A$, and similarly $\mathbf{Alg}_B$, a streaming algorithm for μ. Indeed, for inputs over vertices $[n]$ distributed according to μ, simply pick a random σ apply it to the input, and run $\mathbf{Alg}_A$ on the graph G_σ. Then output $E^*_0 := \sigma^{-1}(E^*_A)$.
First observe that the distribution of σ and the distribution of G are independent. Therefore, conditioned on G, the injection σ is uniform. It follows that μ_2's first marginal is also the distribution of G_σ, where $G \sim \mu$ and σ is uniform and independent.
Then, using Definition 4.4, with probability $1 - o(1)$ over the choice of σ and $G \sim \mu$, we obtain that $|E^*_A \cap E(W' \cap P' \times \overline{W}' \cap Q')| = |E^*_0 \cap E(W \cap P \times \overline{W} \cap Q)| = o(n)$, which concludes the proof. □

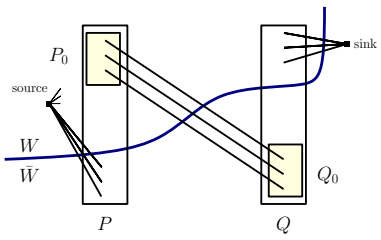

Figure 1: The figure shows how the maxflow mincut theorem is used to argue about the size of a matching in a bipartite graph G over vertex set $P \times Q$ drawn from a distribution μ. Only edges from the cut are drawn. The source and sink are added for the sake of the argument, they are not part of G.

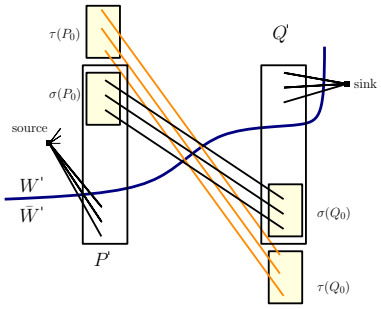

Figure 2: The construction of μ_2.

References

[1] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

[2] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms*, pages 1345–1364, 2016.

[3] László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory. In *Proceedings of the 27th IEEE Foundations of Computer Science*, pages 337–347, 1986.

[4] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

[5] Paul Beame, Martin Tompa, and Peiyuan Yan. Communication-space tradeoffs for unrestricted protocols. In *Proceedings of 31st Foundations of Computer Science*, pages 420–428, 1990.

[6] Therese C. Biedl, Erik D. Demaine, Christian A. Duncan, Rudolf Fleischer, and Stephen G. Kobourov. Tight bounds on maximal and maximum matchings. *Discrete Mathematics*, 285(1-3):7–15, 2004.

[7] Joshua E. Brody, Shiteng Chen, Periklis A. Papakonstantinou, Hao Song, and Xiaoming Sun. Space-bounded communication complexity. In *Proceedings of the 4th Innovations in Theoretical Computer Science*, pages 159–172, 2013.

[8] Ho-Leung Chan, Tak-Wah Lam, Lap-Kei Lee, and Hing-Fung Ting. Continuous monitoring of distributed data streams over a time-based sliding window. *Algorithmica*, 62(3):1088–1111, 2011.

[9] Graham Cormode, S. Muthukrishnan, and Ke Yi. Algorithms for distributed functional monitoring. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms*, pages 1076–1085, 2008.

[10] Phillip B. Gibbons and Srikanta Tirthapura. Estimating simple functions on the union of data streams. In *Proceedings of the 13th ACM Symposium on Parallel Algorithms and Architectures*, pages 281–291, 2001.

[11] Phillip B. Gibbons and Srikanta Tirthapura. Distributed streams algorithms for sliding windows. In *Proceedings of the 14th ACM Symposium on Parallel Algorithms and Architectures*, pages 63–72, 2002.

[12] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the 23d ACM-SIAM Symposium on Discrete Algorithms*, pages 468–485, 2012.

[13] Zengfeng Huang, Ke Yi, and Qin Zhang. Randomized algorithms for tracking distributed count, frequencies, and ranks. In *Proceedings of the 31st Symposium on Principles of Database Systems*, pages 295–306, 2012.

[14] Russell Impagliazzo and Ryan Williams. Communication complexity with synchronized clocks. In *Proceedings of the 25th IEEE Conference on Computational Complexity*, pages 259–269, 2010.

[15] Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms*, pages 1679–1697, 2013.

[16] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms*, pages 734–751, 2014.

[17] Christian Konrad. Maximum matching in turnstile streams. In *Proceedings of 23rd European Symposium on Algorithms*, pages 840–852, 2015.

[18] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, New York, NY, USA, 1997.

[19] Tak Wah Lam, Prasoon Tiwari, and Martin Tompa. Trade-offs between communication and space. *Journal of Computer and System Sciences*, 45(3):296–315, 1992.

[20] Yi Li, Huy L. Nguyen, and David P. Woodruff. Turnstile streaming algorithms might as well be linear sketches. In *Proceedings of the 46th ACM Symposium on Theory of Computing*, pages 174–183, 2014.

[21] Zhenming Liu, Bozidar Radunovic, and Milan Vojnovic. Continuous distributed counting for non-monotonic streams. In *Proceedings of the 31st ACM Symposium on Principles of Database Systems*, pages 307–318, 2012.

[22] Frédéric Magniez, Claire Mathieu, and Ashwin Nayak. Recognizing well-parenthesized expressions in the streaming model. *SIAM Journal on Computing*, 43(6):1880–1905, 2014.

[23] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 2(1):117–236, 2005.

[24] David P. Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In *Proceedings of the 44th ACM Symposium on Theory of Computing*, pages 941–960, 2012.

[25] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing. In *Proceedings of the 11th ACM Symposium on Theory of Computing*, pages 209–213, 1979.

A Missing proofs from Section 2.4

A.1 Proof of Proposition 2.6 and Proposition 2.9

Proof of Proposition 2.6. The following protocol Π' is a simple simulation of the protocol Π in the usual communication model. We assume that both players know when the protocol has ended. Both players update a variable named t supposed to emulate time. Initially, t is set to $t := 0$.

1. If Alice and Bob know that the protocol Π has ended, then they output as in Π and exit.
2. Otherwise, Alice and Bob both declare at which time after t, say t^A, t^B they would send a nonempty message in Π given the current transcript and time t.
3. Whoever has the minimum of (t^A, t^B), sends the message in Π corresponding to this time. If $t^A = t^B$, then both send their message.
4. They update $t = \min(t^A, t^B)$ and go to 1.

The protocol Π' outputs exactly as in Π and we have $C(\Pi') \leq R \times 2 \log n + B$. The factor 2 in the expression $2 \log n$ comes from both Alice and Bob sending a time at step (2) in Π'. We conclude by noticing that $R \leq T$ and $T \leq R \cdot (S + 1)$ (by Proposition 2.5). □

Proof of Proposition 2.9. The streaming problem associated to f can be solved, using an SC protocol and only the memory of the players needs to be written down. The input to this problem is x and y interleaved, by definition of the SC model. The algorithm does not need to store the messages $m_i^{A/B}$. It simply computes them on the fly and does the updates of σ^A, σ^B. Note $2S$ bits suffice to write (σ^A, σ^B). □

A.2 A function that is easy in Streaming and CC but not for Streaming Communication protocols - proof of Theorem 2.8

Rather than exhibiting a function f with the desired properties, in the proof, we consider languages. This is without loss of generality as we can freely move from languages to functions by considering the indicator of the language (for a fixed size of input, n).

Let DYCK(2) be the language consisting of all well parenthesized words with two different types of parenthesis. We will restrict our attention to $L := DYCK(2)_3 \subseteq DYCK(2)$, which consists of words with only 3 alternations. Namely, $x \in L$ iff x can be written in the form $x = u_1 v_1 u_2 v_2 u_3 v_3$ and the u_i's are only made of opening parenthesis whereas the v_i's are only made of closing parenthesis.

Using linear hashing, an algorithm for L is known in the standard streaming model. It only needs $O(\log n)$ memory (see [22]) so that $\mathcal{S}(L) = O(\log n)$. However, if the word x is delivered in an interleaved fashion, much more memory is needed.

We now show that $\mathcal{S}^{\textbf{int}}(L) = \Omega(n)$. The proof follows by providing a direct reduction to a generalization of the INDEX problem from communication complexity.

Definition A.1. *In the INDEX problem, Alice is given a string $u \in \{0,1\}^n$, and Bob is given $u_1, \ldots, u_k, b \in \{0,1\}^{k+1}$ for some k which varies. The goal is for Bob to output $b = x_{k+1}$*

It is known that any one-way protocol where only Alice is allowed to speak, requires at least n bits

$$R^{one-way}(INDEX) = n.$$

We use the notation $\overline{u}$ to denote the same word as u but with all open parenthesis replaced by closing ones and viceversa.
Let $n_1 \in \mathbb{N}$ and set $n := 3n_1 + 2$. We will restrict inputs to the following set

$$A_n = \{x \in \{0,1\}^n, x = uvb\bar{b}\bar{v}w\overline{wu}w'\overline{w'} \mid b \in \{0,1\}, u \in \{0,1\}^{n_1}, v = u_1, \ldots, u_k, k \leq [n_1]\}$$

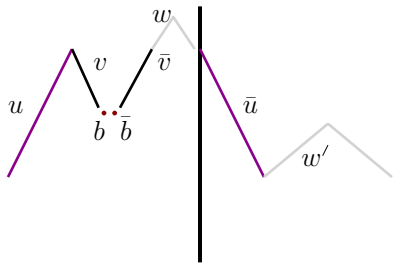

Figure 3: A word from the class A_n. It is represented according to the conventions of [22].

The words w, w' are fixed and made of zeros only. This corresponds to a padding. By setting the length of w, w' such that $|u| + 2(|v| + 1 + |w|) = |u| + 2|w|' = 3n_1 + 2$, we enforce that all words in A_n have the same length $n := 3n_1 + 2$. Also, note that the middle of the word $x = uvb\bar{b}\bar{v}w\overline{wu}w'\overline{w'}$ is exactly after $\overline{w}$. We easily see that $x = uvb\bar{b}\bar{v}w\overline{wu}w'\overline{w'} \in A$ is in L iff $b = x_{k+1}$. In the interleaved problem, the first part of the stream corresponds to $(u, \overline{u})$. The second part corresponds to $(vb\bar{b}\bar{v}w\overline{w}, w'\overline{w'})$. The core lemma is

Lemma A.2. *Assume there is an interleaved streaming algorithm that recognizes A, using m bits. Then, there is a one-way protocol to solve INDEX$_{n_1}$ with m bits, hence $m \geq n_1$.*

Proof. We show how to derive such a protocol for INDEX. Let $u \in \{0,1\}^{n_1}$ be Alice input and $u_1, \ldots, u_k, b$ be Bob's input. Let $v = u_1, \ldots, u_k$. Alice pretends she is getting $(u, \overline{u})$ in the interleaved streaming problem. She then sends the content of her memory to Bob, which pursues as if he was getting $(vb\bar{b}\bar{v}w\overline{w}, w'\overline{w'})$ and he answers as the streaming algorithm would. The correctness follows from the remark that $x = uvb\bar{b}\bar{v}w\overline{wu}w'w' \in A_n$ iff $b = x_{k+1}$. □

Notice that, on the class of inputs A_n we considered $n_1 \sim \frac{n}{3}$. It follows from the lemma, that any algorithm for $DYCK(2)_3$ must use $\Omega(n)$ bits of memory.

A.3 A function that is easy for streaming protocols but not in the interleaved variant - proof of Theorem 2.10

First note that the memory S available to an SC protocol should be $\geq \log n$, otherwise we know from 2.4 the problem is not solvable. Let $X, Y', Y, X' \in \{0,1\}^{\frac{n}{2}}$. Let $f(XY', YX')$ be the function

$$f(XY', YX') := IP(X,Y) \bigwedge X = X' \bigwedge Y = Y'.$$

Proposition A.3. *The randomized communication complexity of f is $O(\log n)$.*

Proof. A communication protocol to solve f, has Alice and Bob check the two equalities $X = X'$ and $Y = Y'$ using $O(\log n)$ bits. The inner product can then be evaluated without communicating, since both players know X and Y. □

Proposition A.4. *In the interleaved streaming model, which corresponds to the two streams combined coming as one, the problem can be solved using $O(\log n)$ bits of memory*

$$\mathcal{S}^{int}(f) = O(\log n).$$

Proof. A streaming protocol checks $IP(X, Y)$ bit by bit, on the fly as they arrive. It also produces a linear hash of length $O(\log n)$ for X and Y so as to check $X = X'$ and $Y = Y'$. □

Proposition A.5. *Any protocol in the SC model that computes f using S bits of memory and R communication rounds must have*

$$R \cdot S = \Omega(n).$$

The proof is very similar to the one of Theorem 3.3 (see Appendix B), so we omit some details and advise the reader to read that proof first.
Inputs will be partitioned into blocks of equal size $\alpha S = \Omega(S)$. The parameter α is a constant to be determined later. Previously we refered to the ℓth block as indices in $[\ell\alpha S, (\ell + 1)\alpha S]$. Now we call the ℓth block indices in $[\ell\alpha S, (\ell + 1)\alpha S] \cup [\frac{n}{2} + \ell\alpha S, \frac{n}{2} + (\ell + 1)\alpha S]$, for reasons that will soon be clear. The parameter n stands for the size of the input, which corresponds to $|X| + |Y|$.
The construction is by induction. More precisely, we argue that there must exist inputs x, y such that some communication happens on each of the ℓ blocks *on average over the randomness*, otherwise we obtain a contradiction by deriving a protocol for $IP_{\alpha S}$ using less than αS bits.
Let us fix Π a protocol for f in the SC model.

Lemma A.6. *For any given $x^{\leq \ell}, y^{\leq \ell}$, the inputs on block $\ell + 1$ denoted $x^{\ell+1}, y^{\ell+1}$ can be chosen such that when running Π on $(x^{\leq \ell+1} w y^{\leq \ell+1} w', y^{\leq \ell+1} w' x^{\leq \ell+1} w)$, where w is some arbitrary word, the expected number of rounds on block $\ell + 1$ is at least $\Omega(1)$, if α is chosen big enough.*

The proof follows from the lemma by using it repeatedly to build inputs such that the lower bound holds simultaneously for every block.

Proof of Lemma A.6. The proof of Proposition A.5 proceeds by reducing to IP in the standard model of CC.
The following two paragraphs assume the reader is familiar with the Section B.2. When adapting the strategy we used in Section B.2, the protocol needs to be changed after step 3, it is not clear that Bob can finish the simulation. Computing f using Π, after step 3, Bob needs to plug in values for X', Y'. By definition of the function f, the protocol will output $IP(X, Y) = IP(x^{\ell+1}, y^{\ell+1})$ only if Bob is also able to pretend he is getting $X' = X$ and $Y' = Y$ or in other words if he can plug $x'^{\ell+1} = x^{\ell+1}$ and $y'^{\ell+1} = y^{\ell+1}$ as SC inputs for f.
However, he only knows his input $y^{\ell+1}$ and not $x^{\ell+1}$. We get around this issue by the following trick. After step 3, Bob also sends Daniel's state to Alice so that *she* does the updates of his memory pretending he is getting $X' = X$ (she knows X). In parallel Bob updates Carole's memory pretending she is seeing Y'.

Let us now give more details. Consider Π over inputs starting with $x^{\leq \ell+1}, y^{\leq \ell+1}$. As before, we denote by $R_{\ell+1}$ the expected number of rounds, on block $\ell+1$.
We are going to find a protocol $\Pi_{\ell+1}$ for $IP_{\alpha S}$, in the usual model (without streams and bounded memory) using less than $R_{\ell+1} \times (S + log\alpha S) + 20S$ bits in expectation and error $\leq \varepsilon$. In turn, using Theorem 2.3 this quantity is greater than $c\alpha S$ if ε is small enough and c is a small enough constant. Choosing a fixed α bigger than $100c^{-1}$ leads to $R_{\ell+1} \geq 80\frac{S}{S+\log \alpha S} = \Omega(1)$ and this concludes the proof.
It only remains to explain how protocol $\Pi_{\ell+1}$ is derived from Π, and this is what we do next.

Algorithm 1: The protocol $\Pi_{\ell+1}$ solving $IP_{\alpha S}$, based on Π.

1 Over inputs $x^{\ell+1}, y^{\ell+1}$, Alice simulates the SC protocol Π on $x^{\leq \ell}, y^{\leq \ell}$ for players Carole and Daniel.
2 Then Alice sends Bob the current memory state of Daniel and message of the round from Carole to Daniel.
3 Alice and Bob jointly simulate Π on block $\ell+1$ pretending they are Carole and Daniel receiving $x^{\ell+1}, y^{\ell+1}$
4 Alice sends Carole's memory state to Bob
5 Bob sends Daniel's memory to Alice
6 Alice and Bob on their own update Carole and Daniel memory until reaching indices in $[\frac{n}{2} + \ell\alpha S, \frac{n}{2} + (\ell+1)\alpha S]$. At this point, they jointly simulate the protocol Π again.
7 Again Alice sends Daniel's memory to Bob.
8 Bob knows and outputs $IP(x^{\ell+1}, y^{\ell+1})$ (details below).

We will now analyze the correctness and cost of $\Pi_{\ell+1}$.
Cost : Using 2.4, we can already evaluate the cost to be in expecatation $R_{\ell+1} \times (S + \log(\alpha S))$ for parts 3 and 6 together and less than $20S$ for all the other memory exchanges at 2,4,5,7. The total cost in expectation is thus less than

$$R_{\ell+1} \times (S + \log(\alpha S)) + 20S.$$

Correctness : Ultimately, at line 6, Alice sends back the memory of Daniel and Bob has the final memory of Carole and Daniel, as if they each received $X' = X$ and $Y = Y'$, so he can answer $f(XY, YX) = IP(X, Y) = IP(x^{\ell+1}, y^{\ell+1})$.
The protocol $\Pi_{\ell+1}$ errs only if the initial procedureΠ errs. Hence $\Pi_{\ell+1}$ solves $IP_{\alpha S}$ in the usual model of CC with expected number of bits $\leq \frac{2\alpha S}{3}$, which is a contradiction. □

B The composition Theorem

We start by exposing the idea of our general bound by providing a tight tradeoff for communication rounds and memory in the SC model for the EQ function. Note that for the public coin randomized case EQ is a trivial problem. Then we show Theorem 3.3 in Section B.2.

B.1 Deterministic EQ

Theorem B.1. *Any deterministic protocol in the SC model computing EQ_n with R communication rounds and S memory, must have $R \cdot S = \Omega(n)$.*

Proof. We are going to use a protocol for EQ in the SC model with inputs of length n, to solve EQ on smaller inputs of length $k \leq n$, in the standard communication model.
Assume we have a protocol Π in the SC model with R number of rounds. Let $k := \frac{n}{R}$ and for simplicity we assume that k is integer. With this choice of k, for any pair of inputs, X, Y in the original protocol there is a silence of length k, by the pigeonhole principle. By *a silence* of length k, we mean a period of length k where all messages sent are empty. However, note that the position of this silent period might be different for different inputs.
We will describe a deterministic protocol for EQ_k in the usual communication model, for some appropriate choice of k, using only $2S + \log k$ bits, which will imply $2S + \log k \geq k$ using the bound for EQ_k. We can assume $k \geq S$, otherwise the desired bound holds, and for k large enough $k \leq 2.1S$, which implies $R \cdot S = \Omega(n)$.

Algorithm 2: Let Π_k be the following protocol for EQ_k, on inputs $x, y \in \{0,1\}^k$.

1 Set $X := x^{\frac{n}{k}} \in \{0,1\}^n$, the concatenation of x, $\frac{n}{k}$ times.
2 Alice simulates Π on X, X, without communication: she simulates *both sides*, call them Carole and Daniel, assuming they *both* have input X.
3 She sends Bob the index $i \in [k]$ of x she is reading one step after the beginning of the silent period.
4 She sends as well Daniel's initial and final state $\sigma_{init}, \sigma_{finish}$.
5 Bob then outputs 1 iff (1) Daniel would also have remained silent if he was getting y (shifted by i) and (2) starting at σ_{init} Daniel would have reached σ_{finish} by the end of the silent period.

We now prove the correctness of the protocol.
If $x = y$, then conditions (1) and (2) hold. If $x \neq y$ then conditions (1) and (2) cannot hold. Assume they do, then we can build two inputs X, Y to the original protocol such that the answer on (X, X) and (X, Y) are the same, which is a contradiction to the correctness of Π.
The word $X = x^{\frac{n}{k}}$ is chosen to be the same as above, and Y is equal to X except on the first k bits of the silent period where x is replaced by y.
All together, we have $x = y$ iff conditions (1) and (2) are correct. In other words, Π_k is a valid protocol for EQ_k. □

B.2 Proof of Theorem 3.3

In this section we prove Theorem 3.3. The bounds are proven in the same way in the deterministic/randomized setting and for bits or rounds. The difference between bits and rounds comes from the difference in the bounds in Proposition 2.6. In what follows, we focus on lower bounding the rounds in the randomized setting and leave to the reader the slight modifications needed for the other cases. The modifications required for the case $G = \bigoplus$ are discussed in the end of the proof.
The idea of the proof is the following: we start by any protocol Π for f in the SC model and we demonstrate that there exist inputs x, y that require a lot of communication rounds on average under Π. More precisely, we argue that there must exist inputs x, y such that some communication happens on each of the L blocks *on average over the randomness*. This is achieved by deriving a protocol, denoted Π_ℓ in the usual communication model for g_ℓ with error $\leq \varepsilon + \delta$ and that uses less than (roughly) $R_\ell \cdot S$ bits, where R_ℓ is the expected number of rounds of communication under Π at block ℓ. This shows a lower bound on R_ℓ. The protocol Π_ℓ is obtained by embedding inputs of g_ℓ in bigger inputs for f, and running the purported efficient protocol Π for f.

More precisely, let Π be a protocol for f in the SC model. Let a, b be the strings given by the assumption that G is non trivial.

Our goal is to find inputs $x = x^1, \ldots, x^L$ and $y = y^1, \ldots, y^L$ for f with some specific properties. To find these inputs, we first find (x^1, y^1), then given the values of (x^1, y^1) we find (x^2, y^2) and so forth, until we complete the inputs for f.

For each $\ell \in [L-1]$, we will prove that given any so-far picked inputs $x^{\leq \ell} = x^1, \ldots, x^\ell$ and $y^{\leq \ell} = y^1, \ldots, y^\ell$ that satisfy $g_1(x^1, y^1), \ldots, g_\ell(x^\ell, y^\ell) = a_{\leq \ell}$, we can always find $(x^{\ell+1}, y^{\ell+1})$ with the following two properties. First, $g_{\ell+1}(x^{\ell+1}, y^{\ell+1}) = a_{\ell+1}$ for the string a that witnesses G is nontrivial. Note that $a_{\ell+1} \in \{0, 1\}$ and there should always be inputs for which $g_{\ell+1}(x^{\ell+1}, y^{\ell+1}) = a_{\ell+1}$, otherwise $g_{\ell+1}$ is constant and we can ignore it from the computation. Second, in the protocol Π for f, the expected number of communication rounds on block $\ell + 1$ is at least $\frac{\delta C_{\varepsilon+\delta}(g_{\ell+1}) - S}{S + 2\log t_{\ell+1} + 1}$. In Lemma B.2 we prove that it is indeed possible to find such inputs for all ℓ and for any prefix. This concludes the proof of the theorem since we first use the lemma to pick (x^1, y^1) with high expected communication, then we use it again to pick (x^2, y^2) with high expected communication (given the prefix (x^1, y^1)) and so forth until we pick the entire inputs x, y in a way that the bounds in the theorem hold. For the case when $G = \bigoplus$, we can remove the δ from the bound by using Lemma B.3.

In the following lemma, observe that we ask for a lower bound on the expected (over public randomness r in the protocol) number of rounds instead of the more standard *maximum* number of rounds over r. This is because, given Π and inputs x, y, the strings r's which give high communication can a priori be different on each block. What we are after is one fixed string r for which the communication is high on many blocks. Of course, proving our lower bound for the expected number of rounds also implies it for the worst case.

Lemma B.2. *For any $\ell \in [L-1]$ and for any pair of prefixes $x^{\leq \ell}, y^{\leq \ell}$ such that $g_1(x^1, y^1), \ldots, g_\ell(x^\ell, y^\ell) = a_{\leq \ell}$ it is possible to find a pair of inputs for block $\ell + 1$, $(x^{\ell+1}, y^{\ell+1})$ such that*

- *$g_{\ell+1}(x^{\ell+1}, y^{\ell+1}) = a_{\ell+1}$;*
- *The expected number of rounds of communication on block $\ell + 1$ under Π, for inputs starting with $x^{\leq \ell+1}, y^{\leq \ell+1}$ is at least $R_{\ell+1} := \frac{\delta C_{\varepsilon+\delta}(g_{\ell+1}) - S}{S + 2\log t_{\ell+1} + 1}$.*

Proof of Lemma B.2. Note that we have assumed without loss of generality that the functions g_ℓ are not constant. For any $\ell \in [L-1]$, denote by $b_\ell \in \{0,1\}^{L-\ell-1}$ the input such that $G(a_{\leq \ell} 0 b_\ell) \neq G(a_{\leq \ell} 1 b_\ell)$.

We now describe a protocol $\Pi_{\ell+1}$ for $g_{\ell+1}$ in the usual communication model that computes $g_{\ell+1}$ with error $\leq \varepsilon + \delta$ and worst case communication cost in bits

$$A_{\ell+1} := \frac{1}{\delta}\left(R_{\ell+1}\left(S + 2\log t_{\ell+1} + 1\right) + S\right). \tag{1}$$

But this quantity has to be greater than $C_{\varepsilon+\delta}(g_{\ell+1})$ and this implies the lower bound on $R_{\ell+1}$ given in the statement. Note that we need to describe a protocol $\Pi_{\ell+1}$ that works *for all inputs* even though the guarantee we have for Π on block $\ell + 1$ is only for inputs $x^{\ell+1}, y^{\ell+1}$ such that $g_{\ell+1}(x^{\ell+1}, y^{\ell+1}) = a_{\ell+1}$. The protocol $\Pi_{\ell+1}$ is described in pseudocode below. Let us call Alice and Bob the players that want to use $\Pi_{\ell+1}$ to compute $g_{\ell+1}$ and Carole and Daniel the fictitious players simulating Π.

We now analyze the protocol $\Pi_{\ell+1}$. Bob has all the data to simulate exactly the protocol Π, which tries to compute the function $f(x, y) = G\left(a_{\leq \ell} g_\ell(x^{\ell+1}, y^{\ell+1}) b_\ell\right)$. Note that it holds that $G\left(a_{\leq \ell} g_\ell(x^{\ell+1}, y^{\ell+1}) b_\ell\right) = g_\ell(x^{\ell+1}, y^{\ell+1})$ when $G(a_{\leq \ell} 0 b_\ell) = 0 \neq G(a_{\leq \ell} 1 b_\ell)$ (this is why Bob outputs c), and $G\left(a_{\leq \ell} g_\ell(x^{\ell+1}, y^{\ell+1}) b_\ell)\right) = 1 - g_\ell(x^{\ell+1}, y^{\ell+1})$ when $G(a_{\leq \ell} 0 b_\ell) = 1 \neq G(a_{\leq \ell} 1 b_\ell)$ (this is why Bob outputs $1 - c$).

It is clear that the communication cost of the protocol $\Pi_{\ell+1}$ is $\leq A_{\ell+1}$, since the protocol exits before at step (6). All we need to do to conclude is justify that the error is less than $\varepsilon + \delta$. There are two cases when

Algorithm 3: The protocol $\Pi_{\ell+1}$ for computing $g_{\ell+1}$ on inputs $x^{\ell+1}, y^{\ell+1}$ based on Π

1. Alice and Bob simulate on their own the SC protocol Π on the lexicographically first inputs $x^{\leq \ell}, y^{\leq \ell}$ such that $g_1(x^1, y^1), \ldots, g_\ell(x^\ell, y^\ell) = a_{\leq \ell}$ for players Carole and Daniel.
2. Then, Alice and Bob jointly simulate Π on block $\ell + 1$ pretending they are Carole and Daniel receiving $x^{\ell+1}, y^{\ell+1}$ respectively.
3. Alice sends Carole's memory state to Bob.
4. Bob finishes the simulation on his own, pretending Carole and Daniel are getting as input some strings x', y' such that $g_{>\ell+1}(x', y') = b_\ell$. Let c be the output of the protocol Π. Bob outputs c if $G(a_{\leq \ell} 0 b_\ell) = 0$ and $1 - c$ if $G(a_{\leq \ell} 0 b_\ell) = 1$.
5. If at any point, the exchanged bits reach $A_{\ell+1}$ (defined in Equation 1), Bob outputs $1 - a_{\ell+1}$ and the protocol stops.

$\Pi_{\ell+1}$ errs. First, when the protocol $g_{\ell+1}\left(x^{\ell+1}, y^{\ell+1}\right) = a_{\ell+1}$ and the protocol exits (and hence outputs $1 - a_{\ell+1}$); second, when the protocol completes the simulation of Π and Π errs. We can upper bound the error probability as

$$\Pr\left[\Pi_{\ell+1} \text{ errs}\right] \leq \Pr[\Pi_{\ell+1} \text{ exits when } g_{\ell+1}(x^{\ell+1}, y^{\ell+1}) = a_{\ell+1}] + \Pr[\Pi \text{ errs}]. \tag{2}$$

We know that $\Pr[\Pi \text{ errs}] \leq \varepsilon$ and hence it remains to show that $\Pr[\Pi_{\ell+1} \text{ exits when } g_{\ell+1}(x^{\ell+1}, y^{\ell+1}) = a_{\ell+1}] \leq \delta$.

First, recall that for all possible inputs $(x^{\ell+1}, y^{\ell+1})$ with $g_{\ell+1}(x^{\ell+1}, y^{\ell+1}) = a_{\ell+1}$, the expected number of communication rounds on block $\ell + 1$ in Π is bounded by $R_{\ell+1}$.

Using the results in section 2.4, the protocol Π on block $\ell+1$ can be simulated by Alice and Bob in the usual communication model with a multiplicative overhead $S+2\log t_{\ell+1}+1$. Hence, the expected communication cost of step 2 is $R_{\ell+1}\left(S + 2\log t_{\ell+1} + 1\right)$.

The communication cost at step 3 is equal to S, and so, the total expected cost of $\Pi_{\ell+1}$, when $g_{\ell+1}(x^{\ell+1}, y^{\ell+1}) = a_{\ell+1}$, is smaller than $R_{\ell+1}(S + 2\log t_{\ell+1} + 1) + S = \delta A_{\ell+1}$. Thus the probability the protocol exits because the transcript reached $A_{\ell+1}$ in this case is smaller than δ by Markov inequality.

Hence we established that $\Pi_{\ell+1}$ is a protocol for $g_{\ell+1}$ with error $\varepsilon + \delta$ using $\leq A_{\ell+1}$ bits in the worst case over inputs. This concludes the proof. □

The case $G = \bigoplus$ In this case, the proof is similar but simpler. In fact, for the $\bigoplus$ function, we do not need to enforce that $g_\ell(x^\ell, y^\ell) = a_\ell$ for any a_ℓ, since basically all prefixes do *not* fix the output of G. The analogous part to Lemma B.2 reads.

Lemma B.3. *For any given SC protocol Π for f, for any $\ell \in [L-1]$ and for any pair of prefixes $x^{\leq \ell}, y^{\leq \ell}$, it is possible to find a pair of inputs for block $\ell + 1$, $(x^{\ell+1}, y^{\ell+1})$ such that the expected number of rounds of communication on block $\ell + 1$ in Π is at least $\frac{C_\varepsilon^{avg}(g_{\ell+1}) - S}{S + 2\log t_{\ell+1} + 1}$.*

Given the lemma, the conclusion of Theorem 3.3 for the case of $G = \bigoplus$ follows easily.

Sketch of proof for Lemma B.3. Again we use Π to derive a protocol for $g_{\ell+1}$ in the usual communication model, using $R_{\ell+1}(S + 2\log t_{\ell+1} + 1) + S$ bits in expectation over block $\ell + 1$, and error ε. This should be at least $C_\varepsilon^{avg}(g_{\ell+1})$ by definition of $C_\varepsilon^{avg}(\cdot)$ and the bound follows. Note that in this case, in step 4, Bob can choose any suffix he wants to finish the simulation of Π. And since we look at the average case complexity,

we do not need step 5 and the protocol $\Pi_{\ell+1}$ can always finish the simulation of Π on block $\ell+1$. Last, for the error, we simply have this time

$$\Pr\left[\Pi_{\ell+1} \text{ errs}\right] = \Pr[\Pi \text{ errs}].$$

□

C Removing overlapping edges

In our previous construction, edges sent to Alice and Bob may overlap. We now explain how to transform a distribution where edges sent to Alice and Bob may overlap to a distribution where edges are disjoint. The transformation is built on a gadget, which replaces G by two copies $G_\times$ or $G_\parallel$ defined as follows.

Definition C.1 (The graph $G_\times$ and the graph $G_\parallel$)**.** *Starting from a graph $G = (P, Q, E)$ with edge set E defined over vertices $P \times Q$ denote by $\ell : P, Q \to P_1, Q_1$ and $r : P, Q \to P_2, Q_2$ two bijections mapping P, Q to disjoint copies P_i, Q_i, for $i = 1, 2$. We define the graph $G_\times$ (resp. $G_\parallel$) over $(P_1 \sqcup P_2, Q_1 \sqcup Q_2)$ by putting the two edges $(\ell(x), r(y)), (r(x), \ell(y))$ (resp. $(\ell(x), \ell(y)), (r(x), r(y))$) for every $(x, y) \in E$ (see Figure 4).*

Corollary C.2. *We may assume in the reduction of Theorem 4.7 that the edges of both players are non overlapping.*

Proof. Assuming we have a hard distribution μ for approximate matching in streaming, we are going to build a distribution μ_2' as in Definition 4.6, but with the extra constraint that the edges of both players are disjoint. We denote by μ_2' the distribution where G is sampled according to μ, then Alice is given $\sigma(G_\times)$ and Bob $\tau(G_\parallel)$ where σ, τ are G-compatible (with the appropriate modification to account for the doubling of the input set).
Note that the marginals of μ_2' have no overlapping edges by construction. Let us explain at a high level why Theorem 4.7 still works using μ_2' rather than μ_2.
Observe that if μ is $(\alpha, n, m(n), \eta)$ hard then so is $\mu_\times$ (resp. $\mu_\parallel$) the distribution under the $\times$ (resp. $\parallel$) transform. Hence Lemma 4.8 follows in the same way as before. On the other hand, the size of the input has doubled but so has the maximum matching.

□

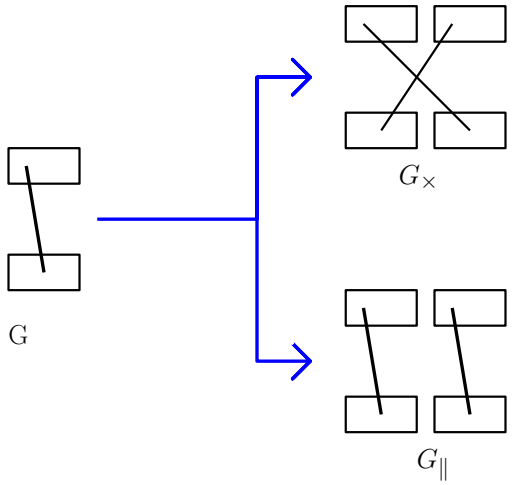

Figure 4: A bipartite graph G, and the associated graphs $G_\times$ and $G_\parallel$.

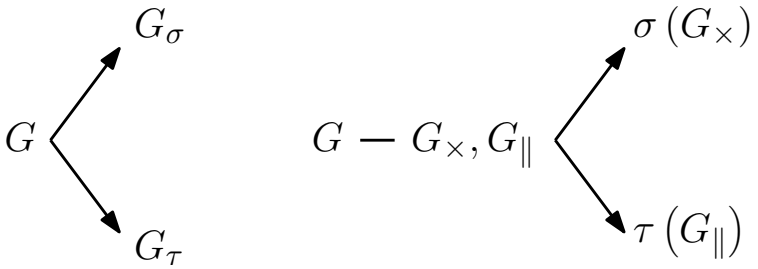

Figure 5: The difference between the two player distribution μ_2 and μ_2' (no overlap).