Streaming Communication Protocols*†

Lucas Boczkowski[1], Iordanis Kerenidis[2], and Frédéric Magniez[3]

1 **CNRS, IRIF, Univ Paris Diderot, Paris, France**
`lucas.boczkowski@irif.fr`
2 **CNRS, IRIF, Univ Paris Diderot, Paris, France**
`iordanis.kerenidis@irif.fr`
3 **CNRS, IRIF, Univ Paris Diderot, Paris, France**
`frederic.magniez@irif.fr`

Abstract

We define the Streaming Communication model that combines the main aspects of communication complexity and streaming. Input arrives as a stream, spread between several agents across a network. Each agent has a bounded memory, which can be updated upon receiving a new bit, or a message from another agent. We provide tight tradeoffs between the necessary resources, i.e. communication between agents 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.

1998 ACM Subject Classification C.2.2 Network Protocols

Keywords and phrases Networks, Communication complexity, Streaming algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.130

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 across a network. The agents want to compute some function f which depends on everybody's input.This is an archetypal problem of Communication Complexity (CC) [29], 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

* Full version available at `https://arxiv.org/abs/1609.07059`.
† This work has been partially supported by the ERC project QCC and the French ANR Blanc project RDAM.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 130; pp. 130:1–130:14


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 [27]. 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 this model can be turned into an algorithm based solely on the updates of linear sketches [25].

1.1 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 5**).

1.2 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 work is [15, 16]. 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 [25]. 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 streaming functional monitoring setup initially proposed by [11], k servers receiving a stream have to allow a coordinator to continuously monitor a given quantity (see also [10, 9, 13] for earlier works in the database community on the monitoring topic). Several follow-up works studied this model (see e.g. [8, 26, 18, 12, 28]). These all focus on communication and do not consider *both* resources, memory storage and communication simultaneously. They can be viewed as extending [15, 16] with greater number of players. The communication model however is still restricted to simultaneous messages to a referee.

Another line of work studies bounded-memory versions of communication [24, 5, 7]. 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 [24], or branching programs [5].

Last, [14] studies another model that deals with distributed parallel streaming platforms, where now, the stream arrives in parallel and arbitrarily partitioned to a set of different agents that communicate in order to solve the task.

1.3 Our results

As a first step, we restrict ourselves in this work to the 2-player case. 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 (number of rounds times memory size) are both lower bounded by the communication complexity $C(f)$ of the f we wish to compute, up to logarithmic factors (**Proposition 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 $\mathcal{S}^{\mathbf{int}}(f)$ 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 8**). Then we show that $\mathcal{S}^{\mathbf{int}}(f)$ is a lower bound on twice the memory size S of players (**Proposition 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 $\mathcal{S}^{\mathbf{int}}(f)$ necessary in the single stream model. We show that this is not true in general by providing an example for which $\mathcal{S}^{\mathbf{int}}(f) = C(f) = O(\log n)$ but $R \cdot S = \Omega(n)$, when $S = \Omega(\log n)$ (**Theorem 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, 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 13**). 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 14**).

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 [25], 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 [21].

It is known that any streaming algorithm for Approximate Matching using $\widetilde{O}(n)$ memory cannot achieve a ratio better than $\frac{e}{e-1}$ [20], 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 [17]. 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 17**), 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 16**), 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, 22], 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 [17, 20]. 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 21**).

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 [29]. For more details about the communication complexity model, please refer to [23]. In the communication complexity models, generically denoted by CC, 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 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** ([23]). *For any $\varepsilon, \delta > 0$, it holds that, $\delta \cdot C_{\varepsilon+\delta}(f) \leq C_\varepsilon^{avg}(f) \leq C_\varepsilon(f)$.*

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 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 [27] 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 [25].

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 5 that their size can be assumed to be $S+1$ without loss of generality.

$\sigma_{i+1}^{A/B} := \Phi^{A/B}(m_i^{B/A}, \sigma_i^{A/B}, r)$ 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 4** (SC protocols)**.** An SC protocol Π uses S bits of memory, R rounds, and T bits when

1. The memory size of each player is at most S bits;
2. The (expected) number of time slots where either $m_i^A \neq \emptyset$ or $m_i^B \neq \emptyset$ is at most R;
3. 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 [19]. 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 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 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 7.** Let $\mathcal{S}^{\mathbf{int}}(f)$ be the amount of memory required for a streaming algorithm to compute f where the input stream is x, 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. A proof of the following result is provided in the full version of this paper.

▶ **Theorem 8.** *There is a function f such that $\mathcal{S}^{\mathbf{int}}(f) = \Omega(n)$, whereas there is a streaming algorithm to compute f with memory $O(\log n)$ when streams are concatenated.*

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

▶ **Proposition 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 $2S \geq S^{\mathbf{int}}(f)$.*

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 $\mathcal{S}^{\mathbf{int}}(f)$ and the communication complexity $C(f)$, at least when $S = O(\mathcal{S}^{\mathbf{int}}(f))$. This appears to not hold in general. The following result is shown in the full version of the paper.

▶ **Theorem 10.** *There exists a function f such that $\mathcal{S}^{\mathbf{int}}(f) = C(f) = O(\log n)$ 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. The assumption we make is of a structural kind. Namely, as explained in Definition 12 we assume the function to be computed can be written in a depth-2 fashion, as a composition of an outer function G with inner gadgets g_ℓ.

▶ **Definition 11.** 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 12** (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 $f(x,y) = G(g_1(x^1, y^1), g_2(x^2, y^2), \ldots, g_L(x^L, y^L))$.

Assuming f is G-decomposable, our goal is to show 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.

▶ **Theorem 13.** *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 $C^{avg}_{\varepsilon}(g_\ell)$.

3.2 Applications

Before proving Theorem 13, 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 14.** *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 with $DISJ_n$. 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 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 3 gives $C_\varepsilon^{avg}(g_\ell) \geq \Omega(k)$. Theorem 13 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 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 15.** *The greedy algorithm, which achieves a 2-approximation, can be implemented using $n \log n$ bits of communication and n rounds in the SC model.*

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 16** (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 w.l.o.g. $e \in G_1$. Then either u_1 or u_2 is matched in M_1 (or both), by maximality of M_1. Map e to (one of) its matched endpoints in M_1. ◀

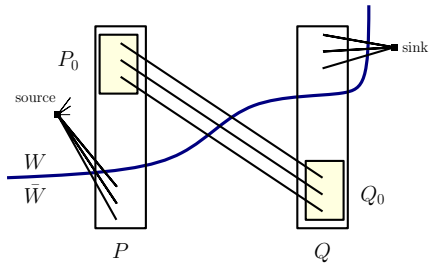

Figure 1 The figure shows how the maxflow mincut theorem is used to argue about the size of a matching in a bipartite graph G over vertex set $P \times Q$ drawn from a distribution μ. Only edges from the cut are drawn. The source and sink are added for the sake of the argument, they are not part of G.

Our lower bound is obtained using a black box reduction that we develop in the following sections. It is a direct consequence of the combination of Theorem 21 and Theorem 19.

▶ **Corollary 17** (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 [17, 20]. 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 18** (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) it holds that $|M| \leq |W \cap Q| + |\overline{W} \cap P| + E((W \cap P) \times (\overline{W} \cap Q))$. 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 [17, 20]. The hard distributions families are not exactly presented as we present them here. The following Theorem follows from results in [20] 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.

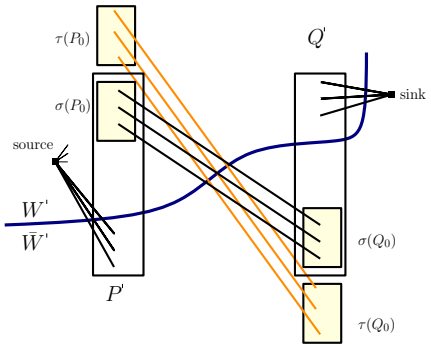

Figure 2 The construction of μ_2.

▶ **Theorem 19** ([20]). *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 [20, Lemma 13], point (2) follows from [20, Claim 12], and point (3) follows from [20, 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 18), 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 20** (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 18.
- 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 (see the full version of this paper). We can now state our main result for the reduction.

▶ **Theorem 21** (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 22 below) then, if E^* is the output matching by a protocol using $o(m)$ memory, then using Lemma 22 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 18). 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 smaller than $\frac{\beta n - 2\eta n}{(1 - \alpha + 2\eta)} = \frac{1+\alpha}{1-\alpha} - O(\eta)$. ◀

▶ **Lemma 22.** *Let* $\boldsymbol{Alg}$ *be a protocol for the two party case, i.e. a pair of algorithms for Alice and Bob* $\boldsymbol{Alg} = (\boldsymbol{Alg}_A, \boldsymbol{Alg}_B)$*. Let* E^*_A *(resp.* E^*_B*) denote the edges* $\boldsymbol{Alg}_A$ *(resp.* $\boldsymbol{Alg}_B$*) 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 18, 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. ◀

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 and Minos N. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In *International Conference on Very Large Data Bases*, pages 13–24, 2005.

10 Graham Cormode, Minos N. Garofalakis, S. Muthukrishnan, and Rajeev Rastogi. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *Special Interest Group on Management of Data*, pages 25–36, 2005.

11 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.

12 Graham Cormode, S. Muthukrishnan, Ke Yi, and Qin Zhang. Continuous sampling from distributed streams. *J. ACM*, 59(2):10:1–10:25, 2012.

13 Graham Cormode, S. Muthukrishnan, and Wei Zhuang. What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *Proceedings of the 22nd International Conference on Data Engineering*, page 57, 2006.

14 Jon Feldman, S. Muthukrishnan, Anastasios Sidiropoulos, Clifford Stein, and Zoya Svitkina. On the complexity of processing massive, unordered, distributed data. *CoRR*, abs/cs/0611108, 2006.

15 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.

16 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.

17 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.

18 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.

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

20 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.

21 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 734–751, 2014. doi:10.1137/1.9781611973402.55.

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

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

24 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.

25 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.

26 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.

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

28 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.

29 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.