Global Versus Local Computations: Fast Computing with Identifiers

Mikaël RABIE

LIP, ENS de Lyon
69007 Lyon, France
mikael.rabie@ens-lyon.org

Abstract. This paper studies what can be computed by using probabilistic local interactions with agents with a very restricted power in polylogarithmic parallel time.
It is known that if agents are only finite state (corresponding to the Population Protocol model by Angluin *et al.*), then only semilinear predicates over the global input can be computed. In fact, if the population starts with a unique leader, these predicates can even be computed in a polylogarithmic parallel time.
If identifiers are added (corresponding to the Community Protocol model by Guerraoui and Ruppert), then more global predicates over the input multiset can be computed. Local predicates over the input sorted according to the identifiers can also be computed, as long as the identifiers are ordered. The time of some of those predicates might require exponential parallel time.
In this paper, we consider what can be computed with Community Protocol in a polylogarithmic number of parallel interactions. We introduce the class $CPPL$ corresponding to protocols that use $O(n \log^k n)$, for some k, expected interactions to compute their predicates, or equivalently a polylogarithmic number of parallel expected interactions.
We provide some computable protocols, some boundaries of the class, using the fact that the population can compute its size. We also prove two impossibility results providing some arguments showing that local computations are no longer easy: the population does not have the time to compare a linear number of consecutive identifiers. The *Linearly Local* languages, such that the rational language $(ab)^*$, are not computable.

1 Introduction

Population Protocols, introduced by Angluin *et al.* in 2004 [3], corresponds to a model of finite states devices with a very restricted memory using pairwise interactions to communicate and compute a global result. Predicates computable by population protocols have been characterized as being precisely the semi-linear predicates; i.e. those equivalent to be definable in first-order Presburger arithmetic [1, 3]. Semi-linearity was shown to be sufficient, and necessary. Those predicates use the global multiset of the input.

Later works on population protocols have concentrated on characterizing what predicates on the input configurations can be stably computed in different variants of the models and under various assumptions. Variants of the original

model considered so far include restriction to one-way communications [1], restriction to particular interaction graphs [2]. Various kinds of fault tolerance have been studied for population protocols [12], including the search for self-stabilizing solutions [5]. Some works also include the Probabilistic Population Protocol model that makes a random scheduling assumption for interactions [4, 13].

Some works extend this model. The edges of the interaction graph may have states that belong to a constant-size set. This model called the *mediated population protocol* is presented in [19]. The addition on Non-Determinism has been studied in [8]. The research of Self-Stabilization (over some fairness assumption) has been explored in [5, 7]. An extension with sensors offering a *cover-time* notion was also studied in [6]. A recent study in [18] also focused on finding the median agent in an extension of the model called Arithmetic Population Protocols.

More generally, the population protocol model shares many features with other models already considered in the literature. In particular, models of pairwise interactions have been used to study the propagation of diseases [17], or rumors [11]. In chemistry, the chemical master equation has been justified using (stochastic) pairwise interactions between the finitely many molecules [15, 20]. The variations over the LOCAL model [14] can be seen as a restriction over the interactions (using a graph) but with a set of possible improvements in agents' capacities.

Agents have been endowed with even stronger tools in different models. The *passively mobile protocols* introduced by Chatzigiannakis *et al.* [10] constitutes a generalization of the population protocol model where finite state agents are replaced by agents that correspond to arbitrary Turing machines with $O(S(n))$ space per-agent, where n is the number of agents. As agents remain initially anonymous, only functions over the global input can be computed.

The *community protocols* introduced by Guerraoui and Ruppert [16] are closer to the original population protocol model, assuming *a priori* agents with individual very restricted computational capabilities. In this model, agents are no longer anonymous: each agent has a unique identifier and can only remember $O(1)$ other agent identifiers. Guerraoui and Ruppert [16] using results about the so-called storage modification machines [21], proved that such protocols simulate Turing machines: Predicates computed by this model with n agents are precisely the predicates in $NSPACE(n \log n)$. The sorted input symbols according to the identifiers can be analysed locally by the protocols to compute the right output. In [9], the possibility that identifiers are no longer unique is explored through the *homonym population protocols* model.

Motivation

Angluin *et al.*, in [4], prove that any computable predicate by a Population Protocol can be computed in $O(n \log^5 n)$ expected interactions, as long as there is a unique leader at the beginning. This article includes some arguments leading to the idea that there might exist protocols computing a leader election in $O(n \log n)$ expected interactions. Doty *et al.* proved in [13] that there cannot be a protocol computing a leader so fast. They proved that a protocol needs

$\Omega(n^2)$ expected interactions to get to a configuration with a single leader, if every agent is a potential candidate at the beginning.

The exact characterization of what can be computed by populations having unique leaders gave the motivation to look to what can be computed in $O(n \log^k n)$ expected interactions (for any $k > 0$), with the Community Protocols model [16]. We consider, as in [4], that each pair of agents (or identifiers) have the same probability to be chosen at each step of a computation. In [4], it is considered that dividing the number of expected interaction by n provides the expected number of parallel interactions.

Community protocols can be seen as interactions controlled by devices in a social group. For example, identifiers can correspond to phone numbers, and the devices can be applications on smartphones. In this vision, it seems intuitive to consider that a group of individuals do not want to stay too long together to compute some global information. Sorting a group of people depending on phone numbers to look for patterns does not seem intuitive, and hence useful.

This paper introduces the class *CPPL*, corresponding to what can be computed with Community Protocols in a polylogarithmic number of expected parallel interactions (which corresponds to a number of expected interactions bounded above by $n \log^k n$ for some k), or a polylogarithmic number of epidemics or broadcasts. We introduce some protocols, proving that the size of the population (or some subset) can be computed in some sense to be explained.

We then show the weakness of this model based on the fact that local computation cannot be performed over the whole input. More precisely, we prove that only a polylogarithmic number of agents can find the next or previous identifier to their own. We also introduce the class of linearly local languages, containing the rational language $(ab)^*$, and prove that none of its elements cannot be computed.

We finish with some comparisons with other computational classes. We introduce a class of Turing Machine trying to match the expressive power of $CPPL$. Those machines use a polylogarithmic space of computation, and is able to use the tools we found. This machine has access to global informations of the input, but can focus locally only on a polylogarithmic number of regions of the input.

The paper is organized as follows: Section 2 provides the Community Protocol model introduced in [16] and includes some examples. Section 3 provides some elements and results about fast computing with Population Protocols from [4]. Section 4 explains a way to keep the fairness of our protocols and describes a way to compute the size of the population. Section 5 introduces the notion of Linearly Local languages and proves that these languages are not in $CPPL$. Section 6 provides some complexity bounds on the class $CPPL$.

2 Model

We present now the model introduced by Guerraoui and Ruppert in [16]: Agents have unique identifiers, and can store a fixed number of them. Agents can compare 2 identifiers. We consider that, unlike in [9], agents cannot know when two identifiers are consecutive.

This model has been proved in [16] to correspond to $NSPACE(n \log n)$, even when we add a fixed number of byzantine agents. We will not consider byzantine agents in this paper.

Definition 1 *A* Community Protocol *is given by seven elements* $(U, B, d, \Sigma, \iota, \omega, \delta)$ *where:*

- U *is the infinite ordered set of identifiers.*
- B *is a finite set of basic states.*
- $d \in \mathbb{N}$ *is the number of identifiers that can be remembered by an agent.*
- Σ *is the finite set of entry symbols.*
- ι *is an input function* $\Sigma \to B$.
- ω *is an output function* $B \to \{True, False\}$.
- δ *is a transition function* $Q^2 \to Q^2$, *with* $Q = B \times U \times (U \cup \{\_\})^d$.

The set $Q = B \times U \times (U \cup \{\_\})^d$ *of possible states each agents can have is such that each agent carries three elements: its identifier, its state, and* d *slots for identifiers.*

The transition function δ *has two restrictions: Agents cannot store identifiers that they never heard about, and the transitions must only depend on relative position of the identifiers in the slots and on the state in* B. *More formally, we have:*

1. *if* $\delta(q_1, q_2) = (q_1', q_2')$, *and id appears in* q_1' *or* q_2' *then id must appear in* q_1 *or in* q_2.
2. *whenever* $\delta(q_1, q_2) = (q_1', q_2')$, *let* $u_1 < u_2 < \cdots < u_k$ *be the distinct identifiers that appear in any of the four states* q_1, q_2, q_1', q_2'. *Let* $v_1 < v_2 < \cdots < v_k$ *be distinct identifiers. If* $\rho(q)$ *is the state obtained from* q *by replacing all occurrences of each identifier* u_i *by* v_i, *then we require that* $\delta(\rho(q_1), \rho(q_2)) = (\rho(q_1'), \rho(q_2'))$.

We also add the fact that δ cannot change the identifier of an agent.

As a convention, we will often call an agent of initial identifier $id \in U$ the agent id. We will sometimes write Id_k for the kth identifier present in the population. An agent with identifier id, in state q and with a list of d identifiers $L = id_1, \ldots, id_d$ will be written in what follows $q_{id,id_1,\ldots,id_d}$.

Example 1 (Leader Election). It is possible to compute a Leader Election (a protocol where all agents start in state L from which we want to reach a configuration with a single L: the leader), where the leader will be the agent with the smallest identifier, with $O(n \log n)$ expected interactions. As a reminder, without identifiers, a protocol needs $\Omega(n^2)$ expected interactions to elect a leader [13].

Agents will store the identifier of their leader. Here is the protocol, using above notations for rules:

- $B = \{L, N\}$.
- $d = 1$.
- $\Sigma = L$, $\iota(L) = L$ and $\omega(L) = \omega(N) = True$.
- δ is such that the non-trivial rules (i.e. where at least one state changes) are:

$$
\begin{array}{rrcll}
L_{id_a,\_} & L_{id_b,\_} & \to & L_{id_a,\_} & N_{id_b,id_a} \text{ with } id_a < id_b \\
L_{id_a,\_} & N_{id_b,id_c} & \to & L_{id_a,\_} & N_{id_b,id_a} \text{ with } id_a < id_c \\
L_{id_a,\_} & N_{id_b,id_c} & \to & N_{id_a,id_c} & N_{id_b,id_c} \text{ with } id_c < id_a \\
N_{id_a,id_b} & N_{id_c,id_d} & \to & N_{id_a,id_b} & N_{id_c,id_b} \text{ with } id_b < id_d
\end{array}
$$

To determine the speed of this protocol, it suffices to realize that the final leader actually does an epidemic to spread its identifier (epidemic is defined in Definition 4). An epidemic takes $O(n \log n)$ expected interactions. Thus, the leader election can be performed in $O(n \log n)$ expected interactions. The notions of time and computation are defined in what follows.

Remark 1. To ensure that at some point, a single leader remains in the population, Gerraoui *et al.* uses the notion of Fairness introduced in the Population Protocols model [3]. As we work here with probabilistic interaction (each pair of agents has the same probability to interact), the fairness notion will not be needed.

Definition 2 *An* Input *is a subset of $U \times \Sigma$ such that any element of U (the elements of U being called* Identifiers*) can appear at most once. Inputs will often be seen as words of Σ^*, as it is possible to sort the input elements according to the identifiers (recall that we consider that U is ordered). An input $u = u_1 \dots u_n$ is such that the agent with the smallest identifier has input u_1, the second has input $u_2 \dots$*

The Initial State *of an agent assigned with the identifier id and the input s is $(\iota(s), id, \_^d)$, where $\_^d$ states for d repetitions of the empty slot $\_$.*

A Configuration *is a subset of Q where two elements cannot have the same first identifier (i.e. two agents must have two distinct identifiers).*

A Step *is the transition between two configurations $C \to C'$, where only two agents' states may change: we apply to the two agents a_1 and a_2 the rule corresponding to their respective state q_1 and q_2, i.e. if $\delta(q_1, q_2) = (q_1', q_2')$ (also written by rule $q_1\ q_2 \to q_1'\ q_2'$), then in C' the respective states of a_1 and a_2 are q_1' and q_2'. All other agents have the same state in C and C'.*

A configuration has an Output *$y \in \{True, False\}$ if for each state $b \in B$ present in the population, $\omega(b) = y$. A configuration C is said* Output Stable *if it has an output y and if, for any C' reachable from C, C' has also the output y.*

An input $w \in \Sigma^*$ has an Output *$y \in Y$ if from any reachable configuration from the initial configuration, we can reach an output stable configuration of output y. It means that from the input, the protocol will reach with probability 1 an output stable configuration, and there is a single output y reachable. The input is* Accepted *if and only if it has output $True$.*

A protocol Computes *a set L if, for any input word $w \in \Sigma^*$, the protocol provides an output, and the protocol accepts w if and only if $w \in L$. We then say that L is* Computable*. We will sometimes say that the protocol is* Las Vegas*, as it will always succeed to provide an output with probability 1.*

A language is Computed in *$f(n)$* Expected Interactions *if, for any input w, the expected number of interactions to reach an output stable configuration is bounded above by $f(|w|)$.*

The Community Protocols model has been fully characterized:

Theorem 1 ([16]). *The decisions problems computable by community protocols correspond exactly to the class* $NSPACE(n \log n)$.

The set of languages computable by community protocols is $NSPACE(n \log n)$.

Let us introduce now the class we will work with in this paper:

Definition 3 *We define the class* CPPL *as the sets of languages that can be recognized by a Community Protocol with* $O(n \log^k n)$ *expected interactions for some* $k \in \mathbb{N}$*, where each pair of agents has the same probability to interact at each moment.*

We say that a function f *is* n-polylog *if there exists some* k *such that we have* $f(n) \leq n \log^k n$.

3 Fast Computing Known Results

We introduce here some of the elements and results in [4] by Angluin *et al.*. These elements are on the Population Protocols model. It corresponds to the case where agents do not have identifiers.

The results are based under the assumption that the population starts with a unique leader. With community protocols, this assumption will no longer be used, we will always consider the leader to be the agent with the smallest identifier (see Example 1).

We introduce the main result and some tools from [4] that will be used in this paper. We first introduce the notion of epidemics, which will be our main tool to perform computations. We will quickly talk about the Phase Clock Protocol that permits to be sure with high probability that an epidemic had the time to happen. We finish with a complexity result.

3.1 Epidemics

The epidemic is the most important probabilistic protocol. Its purpose is to spread or gather information. It will permit for example to get an identifier, to check the state of an agent of a given identifier, to check if there exists some agent in a given state...

The important element with this tool is that an epidemic takes $O(n \log n)$ expected interactions. Intuitively, in parallel, at each step, the number of agents aware of the epidemic doubles, using $O(\log n)$ parallel steps to spread.

Definition 4 ([4]) *An* Epidemic Protocol *is a protocol who spreads some information through an epidemic. The purpose is, for a leader, to* Infect *each agent. More formally, if 0 represents the not infected state and 1 the infected one, there is just a non trivial rule:*

$$1\ \ 0 \to 1\ \ 1$$

Most of the time, it will be a leader who will start a spreading of some information. The computation will start in the configuration 10^{n-1} *(one agent in state* 1*, the others being in state* 0*), where 1 represents the leader.*

Proposition 1 ([4]). *Let T be the expected number of interactions before an epidemic protocol starting with a single infected agent infects all the other ones. For any fixed $c > 0$, there exist positive constants c_1 and c_2 such that, for sufficiently large n, with probability at most $1 - n^{-c}$:*

$$c_1 n \log n \leq T \leq c_2 n \log n$$

From this theorem, we know that any epidemic protocol will take $\Theta(n \log n)$ expected interactions. If we are (almost) sure that more than $c_2 n \log n$ interactions occurred, we will be (almost) sure that an epidemic has finished.

To be almost sure that at least $c_2 n \log n$ interactions have occurred, [4] introduced the Phase Clock Protocol. The leader runs a clock between 0 and m for some $m > 0$. Each agent tries to store the current time, following some updating rules. Each time the clock loops (i.e. the leader reaches m and resets the clock), the population is almost sure that at least $c_2 n \log n$ interactions have occurred.

Proposition 2 ([4]). *For any fixed $c, d_1 > 0$, there exist two constants m and d_2 such that, for all sufficiently large n, with probability at least $1 - n^{-c}$ the phase clock protocol with parameter m, completes n^c rounds, where the minimum number of interactions in any of the n^c rounds is at least $d_1 n \log n$ and the maximum is at most $d_2 n \log n$.*

This result permits to be sure, with high probability, that for n^c rounds, in each round, an epidemic had the time to occur.

3.2 Presburger's Arithmetic

The main result from [4] is that, if the population starts with a unique leader, any computable predicate by a population protocol can be computed with $O(n \log^5 n)$ expected interactions.

Theorem 2 ([4]). *For any predicate P definable in Presburger's Arithmetic, and for any $c > 0$, there is a probabilistic population protocol with a leader to compute P without error that converges in $O(n \log^5 n)$ interactions with probability at least $1 - n^{-c}$.*

As a reminder, those predicates correspond to boolean combinations of:

- *Threshold Predicate*: $\sum a_i x_i \geq b$, with $a_1, \ldots, a_n, b \in \mathbb{Z}^{n+1}$.
- *Modulo Predicate*: $\sum a_i x_i \equiv b[c]$, with $a_1, \ldots, a_n, b, c \in \mathbb{Z}^{n+2}$.

where x_i corresponds to the number of agents with input $i \in \Sigma$. This also corresponds to semilinear sets.

Corollary 1. *Any predicate definable in Presburger's Arithmetic is in $CPPL$.*

Proof. We use the two following facts:

- The Leader Election can be performed in $O(n \log n)$ (see Example 1).
- Any predicate definable in Presburger's Arithmetic can be computed in $O(n \log^5 n)$ expected interactions (see Theorem 2), as long as there is a single leader.

Each agent stores the smallest identifier it has heard about in its Leader slot. It links its internal clock to the leader: if it meets an agent storing a smallest identifier, it acts as if its own clock was at 0, and performs the interaction with the other agent accordingly. Hence, each agent will act as in the protocols of [4] as soon as it hears about the right leader's identifier (in [4], agents start their role in the computation as soon as they get instruction from the leader, or from someone who transmits leader's instruction through an epidemic).

4 Some Computable Protocols

We are now able to introduce some probabilistic protocols, including a complex one that encodes the size of the population. Let first introduce the following notion:

Definition 5 *We will often talk about* Next *and* Previous*. Those are two functions $U \to U$ that provides, to a given identifier, the next one/previous one present in the population. More formally:*

- $Next(id_a) = \min\{id_b : id_b > id_a\}$.
- $Previous(id_a) = \max\{id_b : id_b < id_a\}$.

By convention, $Next$ of the highest identifier is the smallest, and $Previous$ of the smallest identifier is the highest one. Thus, these two functions are bijective.

Sometimes, $Next$ and $Previous$ will be slots in protocols, with the purpose to find the right identifier corresponding to the function. "Finding its $Next$" means that the agent needs to put the right identifier in its slot $Next$.

4.1 From Monte Carlo to Las Vegas Protocols

We considered in the previous section Monte Carlo protocols (i.e. protocols having eventually a probability of failure). We accept that the protocols might have some probability of failure, as long as we can minimize it as much as needed (we use the same bound of $1 - n^{-c}$ as in [4]). Those protocols alone do not compulsory compute any set.

We provide in this paper Monte Carlo descriptions of the protocols. We consider that the protocols also run in parallel a Las Vegas protocol providing the right output with probability 1 (the corresponding Las Vegas protocols exist, as a consequence of Theorem 1). The protocol detects, as in [4], when the Las Vegas protocol should have finished to find the output. At this point, each agent switches its output from the Monte Carlo protocol's to the Las Vegas protocol's. With probability at least $1 - n^{-c}$, this will not change the output.

Here is a small result to justify that we can transform our protocols presented in this paper in Las Vegas ones by multiplying the expected number of interactions by n^3:

Proposition 3. *Let be a population where all agents has found their $Next$ (see definition 5). There exists a protocol that simulates an epidemic spread from an agent taking $O(n^3)$ expected interactions, with a success of probability 1. In the new protocol, the agent meets all the other ones in the population.*

Proof. We suppose that all agents have already found their $Next$, and we suppose all agents know the leader's identifier, being the smallest identifier in the population. The agent needs to meet the leader, then remembers the $Next$ of the leader, meets it, remembers its $Next\ldots$ until it finds the agent having as $Next$ the leader's identifier. At this point, the agent has met all agents in the population.

Each step takes $\frac{n(n-1)}{2}$ expected interactions, and we have n steps. Hence, this protocol takes $O(n^3)$ expected interactions to derandomize the epidemic from the initial agent.

Finding each $Next$ needs at most $O(n^2 \log n)$ expected interactions (which corresponds to the number of interactions expected before every possible interaction has occurred at least once). Detecting when an agent found a new $Next$ is easy: the corresponding agent goes to find the leader to give the information. This latter then resets its computation, spreading the information as in the previous proof. With probability 1, at some point, all agents will have found the right $Next$ and the leader will then reset for the last time the computation.

We will also use some protocols of [4]. Even though some parts use only epidemics, others are trying to detect when something has finally occurred (for example, detect when some state no longer appears in the population). When our Las Vegas protocol will run this detection, it will iterate the epidemic part until it detects the desired fact. In [4], those elements are proved to happen with high probability in a single epidemic. Hence, our expectation will not grow here.

We can prove that this protocol takes at most $O(n^2 \log n + n^2 \log n + n^3) = O(n^3)$ expected interactions to reset for the last time the computation. Then, we add a factor of n^3 to the expected number of interactions taken by the Monte Carlo protocol to make it Las Vegas.

As the Monte Carlo protocol fails with probability at most n^{-c} and that the expected number of interactions of the Las Vegas protocol is polynomial, the parallel expectation is still polylogarithmic.

4.2 The Size of the Population

The purpose of the following section is to find a way to compute the size of the population. As each agent can only contain a finite state, each agent will store one bit, and the $\log n$ first agents (according to their identifiers) will ultimately have the size written in binary when you align their bits according to their order. This way to encode an input size was also used in [9].

The protocol uses a sub-protocol that computes the median identifier of a given subset of agents. Used on the whole population, we get the first bit of the size (depending on if we have the same number of identifiers bigger and smaller to this identifier or not). We can then work on half the population. We iterate the protocol on the new half to get a new bit and a new half.

Theorem 3. *Finding the median identifier can be done in a polylogarithmic number of parallel interactions. The median identifier is the identifier Med*

such that:

$$|\{id : id \leq Med\}| - |\{id : id > Med\}| \in \{0, 1\}$$

Proof. We will give an idea here of the protocol. A better description can be found in the appendix.

The protocol works by dichotomy. It keeps and updates two identifiers Min and Max that bounds the median identifier. Here is a quick description of the steps of the protocol:

1. We initialize Min and Max by finding through an epidemic the smallest and the highest identifier present in the population.
2. The leader takes at random an identifier $Cand$ in $]Min, Max[$, by picking the first identifier in the interval it hears about (spreading the search of such an identifier and the reception takes two epidemics).
3. The leader performs the predicates $|x_{\leq Cand} - x_{>Cand}| = 0$, $|x_{\leq Cand} - x_{>Cand}| = 1$ and $|x_{\leq Cand} - x_{>Cand}| \geq 2$, using protocols from [4] (see Theorem 2), where $x_{\leq Cand}$ is the number of agents with an identifier smaller or equal to $Cand$ and $x_{>Cand}$ is the number of agents with an identifier higher than $Cand$.
   - If the answer is $True$ for one of the two first predicates, $Cand$ is the median identifier. The algorithm is over.
   - If the answer for the third predicate is $True$, we have $Min < Cand < Med < Max$. We replace Min with $Cand$ and go back to Step 2.
   - Else, we know that $Min < Med < Cand < Max$. We replace Max with $Cand$ and go back to Step 2.

We prove in the appendix that there is a probability greater than $\frac{1}{4}$ to divide by $\frac{4}{3}$ the number of identifiers in the interval $]Min, Max[$ after one loop of the algorithm. This permits to conclude that this algorithm will take $O(\log n)$ expected iterations.

Each iteration using $O(n \log^5 n)$ expected interactions, we get that this protocol is in $CPPL$.

The previous protocol will be used as a tool to write the size of the population on the $\log n$ first agents. It still work on any subset of agents.

Theorem 4. *There exists a protocol that writes in binary on the first* $\log n$ *agents the size of the population.*

Proof. To build this protocol, we first adapt the previous one as follows:

- The Median protocol can be used on a segment:
  Instead on working on the whole population, we accept to launch it with two identifiers A and B. We will look on the median identifiers among those who are in $[A, B]$.
- The protocol needs to check if the number of agents in the segment $[A, B]$ is even or odd. This corresponds to check if $|\{A \leq id \leq Med\}| - |\{B \geq id > Med\}|$ is equal to 0 or 1.

Each agent stores a bit $Size$ set to 0. The bits of the size are computed from the right to the left as follows:

1. Let min (resp. max) be the smallest (resp. higher) identifier present in the population. We initialize A and B with, respectively, min and max.
   We also initialize an identifier C to min, it will represent the cursor pointing to which agent we write the bit of the size of the population when it is computed.
2. We compute Med, the median agent in $[A, B]$, and write the parity on the bit $Size$ of agent C.
3. We update the identifiers as follows: $B \leftarrow Med$, $C \leftarrow Next(C)$.
4. If $A \neq B$, we come back to step 2, else the computation is over.

When this protocol is over, we have

$$n = \sum_{i=0}^{\log n} 2^i Size_{Next^i(min)}.$$

where $Size_{Next^i(min)}$ is the bit $Size$ of the $(i+1)$th agent.

The Median protocol will be iterated exactly $\log n$ times. This concludes the proof.

5 Impossibility Results

In this section, we provide two results that motivate the idea that the population cannot take into consideration precisely the sub-words in the population (and hence, focus locally on the input). More precisely, only a polylogarithmic number of agents may know what there is exactly on their "neighbors". It is supported by the fact that only a polylogarithmic number of agents will know the identifier next of their own (Theorem 5).

The proof that Linearly Local Languages (see Definition 6) are not in $CPPL$ (Theorem 6) is based on the fact that there is a pair of consecutive identifiers such that, with high probability, the population will not be able to differentiate them, as these identifiers will not appear in a common interaction during the computation.

Theorem 5. *Any population protocols needs at least $\Omega(n\sqrt{n})$ expected interactions until each agent has found its $Next$.*

The proof is in the appendix.

We bring now another impossibility result. We show that Community Protocols cannot link a linear number of consecutive identifiers in $CPPL$. To prove this, we introduce a new class of languages:

Definition 6 *Let $u = u_1 \dots u_N$ a word of size N and $i < N$. We call $\sigma_i(u)$ the word u where the ith letter is permuted with the next one. More formally, we have:*

$$\sigma_i(u) = u_1 \dots u_{i-1} u_{i+1} u_i u_{i+2} \dots u_N$$

We say that a language L is Linearly Local *if there exists some $\alpha \in]0, 1]$ such that, for any n, there exists some $u \in L$ and some $I \subset \mathbb{N}$ such that:*

$u = u_1 \dots u_N$ with $N \geq n$, $\exists I \subset [1, N-1]$, $|I| \geq \alpha N$ and for all $i \in I$, $\sigma_i(u) \notin L$.

These languages are said linearly local as, for any size of input, we can find words that have a linear number of local regions where a small permutation of letter leads to a word not in the language.

Theorem 6. *There is no linearly local language in CPPL.*

To prove this result, the idea is to prove that for any protocol, and for any n, there exists some u in the language of length at least n and $i \in I$ such that there is a high enough probability that the protocol acts the same way on the inputs u and $\sigma_i(u)$.

Let $\alpha \leq 1$, and let $(I_n)_{n\in\mathbb{N}}$ be a sequence such that, for any $n \in \mathbb{N}$, we have $I_n \subset [1, n]$ and $|I_n| \geq \alpha n$.

We work on pairs $(Id_i, Id_{i+1})_{i\in I_n}$. We want to prove that, for any n, there is some $i \in I_n$ such as, with high probability, the identifiers Id_i and Id_{i+1} never appeared in the same interaction after any n-polylog number of interactions. In the proof, Id_i meets Id_{i+1} means both identifiers appear in the slots of two interacting agents when the interaction occurs.

To prove that, we first introduce the 3 following lemmas. The proof of the two first ones can be found in the appendix.

Lemma 1. *Let f a n-polylog function and let $\alpha > 0$.*

To each identifier id, we define the set E_{id} and value M_{id} as $E_{id} = \{$Agents having had id in one of its register after $f(n)$ steps$\}$ and $M_{id} = |E_{id}|$.

There exists some polylogarithmic function g such that, for n large enough, after $f(n)$ steps:

$$\mathbb{E}(|\{id : M_{id} \leq g(n)\}|) \geq \left(1 - \frac{\alpha}{2}\right) n$$

With this first result, we deduce that at most a small fraction of the pairs (Id_i, Id_{i+1}) could have met after n-polylog number of steps. This means that Id_i and Id_{i+1} never appeared in the slots of two agents that interacted together, when they interacted.

Lemma 2. *Let f be a n-polylog function. For n big enough, after $f(n)$ steps:*

$$\mathbb{E}(|\{i \in I_n : Id_i \text{ and } Id_{i+1} \text{ were in a same interaction}\}|) \leq \frac{3}{4}\alpha n$$

Lemma 3. *Let f be a n-polylog function. For any n large enough, there exists $i \in I_n$ such as:*

$$\Pr(Id_i \text{ met } Id_{i+1}) \leq \frac{3}{4}$$

Proof. Let suppose that for any i, $\Pr(Id_i \text{ met } Id_{i+1}) \geq \frac{3}{4}$.

That implies, $\mathbb{E}(N) = \sum_{i\in I_n} \Pr(Id_i \text{ met } Id_{i+1}) \geq \frac{3}{4}\alpha n$.

This is a direct contradiction of the previous lemma.

To prove our theorem, we need to prove the following proposition:

Proposition 4. *For any protocol, for any n-polylog function f, for any input of size n large enough, there exists some $i \in I_n$ such that the probability that the identifiers Id_i and Id_{i+1} never appeared on a same interaction after $f(n)$ steps is greater than $\frac{1}{4}$.*

This proposition is a direct corollary of previous lemma. With this proposition, the proof of Theorem 6 can be done as follows:

Proof. Let L be a linearly local language with parameter $\alpha > 0$. Let P be a protocol computing L in less than $n \log^m n$ expected interactions for some $m \in \mathbb{N}$.

Let choose n large enough to have the property of Proposition 4 with $f(n) = 9n \log^m n$. Let u be a word of size $N \geq n$ such that the corresponding I has a size greater than αN.

We have, from Markov's Inequality that :

$\Pr(\text{number of steps to compute } u \leq 9N \log^m N) \geq \frac{8}{9}$.

It implies that at least $\frac{8}{9}$ of the sequences of configurations of length $9N \log^m N$ provides the right output.

By applying the previous proposition, we obtain the existence of some $i \in I$ such that the probability that the identifiers Id_i and Id_{i+1} never appeared on a same interaction after $9N \log^m N$ steps is greater than $\frac{1}{4}$.

This implies that in at least $\frac{1}{4}$ of the sequences of configurations of length $9N \log^m N$, Id_i and Id_{i+1} were never in a common interaction.

Hence, if Id_i and Id_{i+1} never appear on a same interaction, then P will not see any difference between the two inputs u and $\sigma_i(u)$.

Between the $\frac{8}{9}$ of the sequences that provides the right output on these two inputs, at least $\frac{7}{9}$ are common (two sequences are here said to be common if the sequence of the interacting identifiers are equals).

As $\frac{7}{9} \geq 1 - \frac{1}{4}$, amongst those common sequences, some of them does not involve Id_i and Id_{i+1} in a same interaction. As the protocol cannot differentiate those two inputs during those sequences, it cannot bring the right output.

This provides a contradiction. There is no protocol in $CPPL$ that computes L.

Corollary 2. *The rational language $(ab)^*$, the rational language of words not containing the subword (ab), the well-formed parenthesis language and the palindrome language are not in $CPPL$.*

Proof. For the first language, to each n we can associate $u = (ab)^n$, with $\alpha = 1/2$. Same thing with the third one, replacing a with the opening parenthesis and b with the closing one. For the fourth, $(ab)^n a$ works the same way. Finally, for the second one, $(bac)^n$ and $\alpha = 1/3$ works.

6 Set Considerations

We provide finally, in this section, set comparisons with $CPPL$. We first give a large upper bound:

Theorem 7.

$$CPPL \subset NSPACE(n \log n) \cap \bigcup_{k \in \mathbb{N}} SPACE(n \log^k n)$$

The result is a combination of Theorem 1 with a lemma proved in the appendix.

We now provide a class of Turing Machines that computes everything we found to be computable yet. This class of machines is capable of computing global properties, through the ability to work on subsets of agents. It is capable to compute the size of sets of agents. It can perform any polylogarithmic number of steps of a regular Turing Machine.

This machines are capable of focusing only on a polylogarithmic regions of agents. It motivates the belief that Community Protocols are not capable of local knowledge on too much places.

Theorem 8. *Let M_T a Turing Machine on alphabet Γ recognizing the language L having the following restrictions. There exists some $k \in \mathbb{N}$ such that*

- *M_T has 4 tapes. The first one is for the input x.*
- *The space of work is restricted as follows:*
  - *The first tape uses only the input space of $|x|$ cells.*
  - *The 2nd and the 3rd use at most a space of $\log |x|$ cells.*
  - *The 4th uses at most a space of $\log^k |x|$ cells.*
- *M_T can only do at most $\log^k |x|$ unitary operations among the following ones:*
  1. *A regular Turing Machine step.*
  2. *Mark/Unmark the cells that have the symbol $\gamma \in \Gamma$.*
  3. *Write in binary on the 2nd tape the number of marked cells.*
  4. *Go to the cell of the number written on the 3rd tape if this number is smaller than $|x|$.*
  5. *Mark/Unmark all the cells left to the pointing head on the first tape.*
  6. *Turn into state γ' all the marked cells in state $\gamma \in \Gamma$.*
  7. *Select homogeneously a random number between 1 and the number written on the 3rd tape if this number is smaller than $|x|$.*

Then we have $L \in CPPL$.

Proof. The proof is in the appendix. All the items are proved using previous results.

References

1. D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing, DISC*, 2007.
2. Dana Angluin, James Aspnes, Melody Chan, Michael J. Fischer, Hong Jiang, and René Peralta. Stably computable properties of network graphs. In *Distributed Computing in Sensor Systems*, June 2005.
3. D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *Principles of Distributed Computing, PODC*, July 2004.
4. D. Angluin, J. Aspnes, and D. Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing, DISC*, 2008.
5. D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing population protocols. In *on Principles of Distributed Systems OPODIS*, December 2005.
6. J. Beauquier, P. Blanchard, J. Burman, and S. Delaët. Computing time complexity of population protocols with cover times - the zebranet example. In *Stabilization, Safety, and Security of Distributed Systems, SSS*, 2011.

7. J. Beauquier, J. Burman, and S. Kutten. Making population protocols self-stabilizing. In *Stabilization, Safety, and Security of Distributed Systems, SSS*, 2009.
8. J. Beauquier, J. Burman, L. Rosaz, and B. Rozoy. Non-deterministic population protocols. In *Principles of Distributed Systems, OPODIS*, 2012.
9. O. Bournez, J. Cohen, and M. Rabie. Homonym population protocols. *NETYS*, 2015.
10. I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P. G. Spirakis. Passively mobile communicating machines that use restricted space. In *International Workshop on Foundations of Mobile Computing*, FOMC '11, 2011.
11. DJ Daley and DG Kendall. Stochastic Rumours. *IMA Journal of Applied Mathematics*, 1965.
12. C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. When birds die: Making population protocols fault-tolerant. In *Distributed Computing in Sensor Systems, Second IEEE International Conference, DCOSS* , 2006.
13. D. Doty and D. Soloveichik. Stable leader election in population protocols requires linear time. In *Distributed Computing, DISC*, 2015.
14. P. Fraigniaud, A. Korman, and E. Lebhar. Local MST computation with short advice. In *Symposium on Parallelism in Algorithms and Architectures, SPAA*, 2007.
15. D.T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A*, 1992.
16. R. Guerraoui and E. Ruppert. Names trump malice: Tiny mobile agents can tolerate byzantine failures. *Automata, Languages and Programming*, 2009.
17. H. W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, December 2000.
18. G. B. Mertzios, O. E. Nikoletseas, C. L. Raptopoulos and P. G. Spirakis. Stably Computing Order Statistics with Arithmetic Population Protocols *Mathematical Foundations of Computer Science, MFCS*, 2016.
19. O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Mediated population protocols. *Theor. Comput. Sci.*, 2011.
20. J. Dickson Murray. *Mathematical Biology. I: An Introduction.* Springer, third edition, 2002.
21. A. Schönhage. Storage modification machines. *SIAM Journal on Computing*,1980.

A Proof of Theorem 3

Here is a better description of the median identifier protocol.

As said, this algorithm is based on dichotomy. Let construct the following recursive protocol that has as input two identifiers Min and Max and tries to find med knowing that it is in $]Min, Max[$.

We do not provide a full description of the protocol but all the ideas to build it. We first describe the d slots of identifiers. Each agent will also store some bits also described bellow. We then describe the algorithm of the protocol. We finally provide a lemma that proves that the protocol is in $CPPL$.

Each agent stores four slots of identifiers and three bits:

- Slot $Leader$ containing the leader's identifier.
- Slots Min and Max (that will be initialized in steps 0.1 and 0.2, before launching the first time the dichotomy protocol).
- Slot $Cand$ that contains the current candidate to be the median identifier.
- Bit C equals to 1 if and only if the agent's identifier is in $]Min, Max[$.
- Bit D equals to 1 if and only if $Cand$ is the candidate identifier selected by the leader.
- Bit G that equals 1 if and only if the agent's identifier is strictly greater than $Cand$.
- Bit S that equals 1 if and only if the agent's identifier is smaller or equal to $Cand$.

To an agent with identifier id, its slot $Slot$ will be written $Slot_{id}$ and its bit B will be written B_{id}.

We will not describe formally all the rules of the protocol but the essential steps. Here are of the steps of the protocol (0.1, 0.2 and 0.3 are the initializing steps):

Step 0.1 We start with a Leader Election using the smallest identifier as a Leader. Each agent stores the leader's identifier in its slot $Leader$ and in its slot Min.

Step 0.2 At the same time, the Leader performs an epidemic, each agent updating its slot Max each time they see a bigger candidate.
We know that when the epidemic stops, each agent, with high probability, knows the smallest and the biggest identifier present in the population.

Step 0.3 Every agents that are not Min and Max set their slot $Cand$ to their own identifier, put their bit C to 1 and their bit D to 0.
Min and Max put _ in slot $Cand$ and set their bit C and D to 0.

Step 1. The Leader looks for a candidate to be the medium identifier Med. For this, it propagates an epidemic where each agent such that $Cand =$ _ fills it as soon as it meets an agent having an identifier with this slot filled (putting it in its own slot).

Step 2. When the leader finds a candidate, it starts a new epidemic to spread the candidate's identifier to each agent. When an agent id_1 with $D_{id_1} = 0$ meets and agent id_2 with $D_{id_2} = 1$, it copies $Cand_{id_2}$ in its slot $Cand_{id_1}$ and switches its own bit D_{id_1}. Then it updates its bits G_{id_1} and S_{id_1} (according to its relative position to the identifier $Cand_{id_2}$).

Step 3. The leader launches two protocols described in [4]. The first one decides if $\sum_{id}(S_{id} - G_{id}) \in \{0, 1\}$ and the second one decides if $\sum_{id}(S_{id} - G_{id}) \geq 2$.
Section 3.2 recalled that a subtraction can be performed in $O(\log^3 n)$ expected epidemics.
Note that $\sum_{id} S_{id}$ corresponds to the number of agents having an identifier smaller or equal to slot $Cand$, and $\sum_{id} G_{id}$ corresponds to the number of agents greater.

Step 4. According to the result, the protocol acts as follows:
- If $\sum_{id}(S_{id} - G_{id}) \in \{0, 1\}$, slot $Cand$ corresponds to identifier Med, the protocols ends.
- If $\sum_{id}(S_{id} - G_{id}) \geq 2$, we have $Min < Cand < Med < Max$. The Leader does an epidemic asking each agent to put identifier $Cand$ in their slot Min. Each agent checks if its identifier is smaller than $Cand$. If it is the case, the agent updates $Cand$ to _ and switches C to 0, else it updates $Cand$ to its own identifier.
  Then the Leader goes back to step 1.
- Else, we have $Min < Med < Cand < Max$. The Leader does an epidemic asking each agent to put $Cand$ in their Max. Each agent checks if its identifier is higher than $Cand$. If it is the case, the agent updates $Cand$ to _ and switches C to 0, else it updates $Cand$ to its own identifier.
  Then the Leader goes back to step 1.

Lemma 4. *The protocol described above finishes in $O(n \log^3 n)$ expected interactions.*

Proof. This protocol finishes, as in each step, there is at least one less candidate (formally, $\sum_{id} C_{id}$ is strictly decreasing after each passage in the loop). Each loop uses a polylogarithmic number of epidemics (The decision problem defined in [4] uses $O(\log^2 n)$ expected interactions).

Hence, we only need to prove that we have a polylogarithmic number of loops in expectation.

Let c_k be the number of agents such as $C_{id} = 1$ after the kth loop. We have $c_0 = n$ and $\forall k$, $c_k > c_{k+1}$. We will show now that if $c_k > 1$, $\Pr\left(c_{k+1} \leq \frac{3}{4} c_k\right) \geq \frac{1}{4}$.

We notice first that each agent having its bit C set to 1 has the same probability to be chosen by the Leader, as each of these agents act the same way. We will call $Cand$ the identifier of the selected agent.

We assume that $|\{id : id \leq Med \ \& \ C_{id} = 1\}| \geq |\{id : id \geq Med \ \& \ C_{id} = 1\}|$ (the case $\leq$ is symmetric). We have $\Pr(Cand \leq Med) \geq 1/2$, as we chose to focus on the case where the majority of candidates have a smaller identifier than Med.

We now suppose that $Cand \leq Med$. Let M be the median identifier of the subset $\{id : id \leq Med \ \& \ C_a = 1\}$. We have $\Pr(Cand \geq M | Cand \leq Med) \geq 1/2$, as M is the median identifier among those smaller than Med.

In the case $Cand \geq M$, we have the following inequality on the number candidates that will no longer be one:
$|\{id : id \leq M \ \& \ C_{id} = 1\}| \geq \frac{1}{2}|\{id : id \leq Med \& C_{id} = 1\}| \geq \frac{1}{2} \times \frac{1}{2}|\{id : C_{id} = 1\}| = \frac{1}{4}c_k$.

Hence, if we have $Cand \geq M$, we have $c_{k+1} = c_k - |\{id : id \leq Cand \ \& \ C_{id} = 1\}| \leq \frac{3}{4}c_k$.

The expectation of the number of trials before we are in this case is less than 4 (as we have more than half a chance to be inferior to Med and then have again more than half a chance to be greater than M).

$(3/4)^i c_0 \leq 1 \Leftrightarrow i \ln(3/4) + \ln n \leq 0 \Leftrightarrow i \geq \frac{\ln n}{\ln 4/3}$.

This protocol uses, in expectation, at most $\frac{4}{\ln 4/3} \ln n$ loops before finding the Med identifier.

This protocol uses $O(n \log^3 n)$ expected interactions to compute the median identifier.

B Proof of Theorem 5

Let T be the number of interactions needed in a sequence of configurations $(C_i)_{i \in \mathbb{N}}$ to reach the point where all agents have the right $Next$. We define, for each identifier id and each configuration C_i, $M_{id}(i)$ as the number of agents, $Previous(id)$ excluded, that had id written in one of its d slots of identifiers in a configuration C_j with $j \leq i$. We also define $M_{id} = M_{id}(T)$.

We first define the following elements:

- $E_{id}(i) = \{id_b \in U : id_b \neq Previous(id)$ & id has been in a slot of id_b in a configuration C_j with $j \leq i\}$. It corresponds to the set of identifiers where id appeared in a slot, excluding $Previous(id)$.
- $M_{id}(i) = |E_{id}(i)|$.
- $E_{id} = E_{id}(T)$ and $M_{id} = M_{id}(T)$.

We first have the following lemma:

Lemma 5.

$$\sum_{id \in U} M_{id} \leq d(n + 2T).$$

Proof. From a configuration C_i, we define, for $id \in U$, $p_{id}(i) \subset U$ the set of identifiers appearing in the d slots of id.

Let $L_{id}(i)$ be the set of all of the $p_{id}(j)$, with $j \leq i$ (we have $L_{id}(i) \in \mathscr{P}(U)$, $\mathscr{P}(U)$ being the set of subsets of U).

Let $N_{id}(i)$ be the number of times id appears in each $L_{id_b}(i)$ and $N_{id} = N_{id}(T)$. More formally:

$$N_{id}(i) = \sum_{id_b \in U} |\{p : id \in p \ \& \ p \in L_{id_b}(i)\}|$$

Finally, let $S(i) = \sum_{id \in U} \sum_{p \in L_{id}(i)} |p|$ and $S = \sum_{id \in U} \sum_{p \in L_{id}} |p|$.

We have $S(i) = \sum_{id \in U} N_{id}(i)$, as each element of each set p is counted exactly once in one of the $N_{id}(i)$.

We also have $N_{id}(i) \geq M_{id}(i)$, as, if $id_b \in E_{id}$, there is some $p \in L_{id_b}$ such as $id_b \in p$. We then also get $N_{id} \geq M_{id}$.

Let $Z(i) = \sum_{id \in U} |L_{id}(i)|$ and $Z = \sum_{id \in U} |L_{id}|$.

We know that for any p, $|p| \leq d$ (each agent stores at most d identifiers at a same time). From this, we get

$$S(i) \leq \sum_{id \in U} \sum_{p \in L_{id}(i)} d = d \sum_{id \in U} |L_{id}(i)| = dZ(i).$$

We can notice that $Z(0) = n$ and that for any i, $Z(i+1) \leq Z(i) + 2$, as an interaction can only add at most one element in the $L(i)$ of the interacting identifiers.

From this, we have $Z(i) \leq n + 2i$ et $Z \leq n + 2T$.

We finally obtain $\sum_{id \in U} M_{id} \leq \sum_{id \in U} N_{id} = S \leq dZ \leq d(n + 2T)$.

We consider now T as the random variable corresponding to the end of the arrangement protocol (i.e. the first time where all agents know their $Next$ identifier), and let $\mathbb{E}(T)$ be its expectation.

Lemma 6.

$$\mathbb{E}(T_f) \geq \frac{1}{\sqrt{20d}} n\sqrt{n}.$$

Proof. We work in the case where $T \leq 2\,\mathbb{E}(T_f)$.

Let p_k be the probability that $k+1$ agents found their $Next$, supposing that k agents have already found their $Next$.

As $Previous$ is bijective and $\{id \text{ without } Next\} \subset \{id \in U\}$, we have

$$p_k = \frac{1}{n(n-1)} \sum_{id \text{ without } Next} M_{Previous(id)} \leq \frac{1}{n(n-1)} \sum_{id} M_{id}.$$

By applying Lemma 5 and the hypothesis $T \leq 2\,\mathbb{E}(T_f)$, we obtain:

$$p_k \leq \frac{d}{n(n-1)}(n + 4\,\mathbb{E}(T_f)).$$

We have $\mathbb{E}(T_f) \geq n$, since the quickest way to end the protocol is by having each agent meeting its $Next$ directly. Hence, $p_k \leq \frac{d}{n(n-1)} 5\,\mathbb{E}(T_f)$.

Hence, $\mathbb{E}(T | T \leq 2\,\mathbb{E}(T_f)) = \sum_{k=0}^{n-1} \frac{1}{p_k} \geq \sum_{k=0}^{n-1} \frac{n(n-1)}{5d\,\mathbb{E}(T_f)} = \frac{n^2(n-1)}{5d\,\mathbb{E}(T_f)}$.

In the general case, we have:
$\mathbb{E}(T) = \Pr(T \leq 2\,\mathbb{E}(T_f)) \cdot \mathbb{E}\left(T | T \leq 2\,\mathbb{E}(T_f)\right) + \Pr(T > 2\,\mathbb{E}(T_f)) \cdot \mathbb{E}(T | T > 2\,\mathbb{E}(T_f))$.

Markov's inequality provides the result that: $\Pr(T \leq 2\,\mathbb{E}(T_f)) \geq \frac{1}{2}$. Hence, we deduce:

$\mathbb{E}(T) \geq \Pr(T \leq 2\,\mathbb{E}(T_f)) \cdot \mathbb{E}\left(T | T \leq 2\,\mathbb{E}(T_f)\right) \geq \frac{n^2(n-1)}{10d\,\mathbb{E}(T_f)}$.

From this, we have $\mathbb{E}(T)^2 \geq \frac{n^2(n-1)}{10d} \geq \frac{n^3}{20d}$. This provides the final result:

$$\mathbb{E}(T_f) \geq \frac{1}{\sqrt{20d}} n\sqrt{n}$$

Lemma 6 permits to conclude the proof of Theorem 5: the arrangement protocol must take at least $O(n\sqrt{n})$ expected interactions.

C Proof of Lemma 1

Proof. Let h be a function such as $\mathbb{E}(|\{id : M_{id} \geq h(n)\}|) \geq \frac{\alpha}{2}n$.

We suppose now to be in the case where $|\{id : M_{id} \geq h(n)\}| \geq \frac{\alpha}{2}n$.

For each identifier id, we associate the set $L_{id} \subset \mathscr{P}(U)$ of the list of identifiers the agent id have had during the $f(n)$ first steps.

Let $S = \sum_{id \in U} \sum_{p \in L_{id}} |p|$. We want to prove that $S \geq \frac{\alpha}{2} nh(n)$.

Let N_{id} be the number of occurrences of id in all the L_{id_b}.

We have $N_{id} = \sum_{id_b \in U} |\{p : id \in p \ \&\ p \in L_{id_b}\}|$ and $S = \sum_{id \in U} N_{id}$.

We have $N_{id} \geq M_{id}$ (an agent id appearing at least once in id_b appears at least in one of its lists). As we supposed $M_{id} \geq h(n)$ for at least $\alpha n/2$ agents, we get $S \geq \frac{\alpha}{2} nh(n)$.

Our purpose is to get a lower bound of $Z = \sum_{id \in U} |L_{id}|$.

We know that for any $id \in U$ and $p \in Ł_{id}$, $|p| \leq d$. Then,

$$S \leq \sum_{id \in U} \sum_{p \in L_{id}} d = d \sum_{id \in U} |L_{id}| = dZ.$$

Hence, $Z \geq \frac{\alpha}{2d} nh(n)$.

Let Z_i be the value of Z after i steps.

We have $Z_0 = n$ and $Z_{i+1} \leq Z_i + 2$. Hence, $n + 2f(n) \geq Z \geq \frac{\alpha}{2d} nh(n)$.

From this, we get $h(n) \leq \frac{2d}{\alpha}\left(1 + \frac{2}{n} f(n)\right)$.

As f is n-polylog, h must be polylogarithmic.

If we chose h maximal matching our initial postulate, we get the expected result:

$$\mathbb{E}(|\{id : M_{id} \leq h(n) + 1\}|) \geq \left(1 - \frac{\alpha}{2}\right) n$$

D Proof of Lemma 2

Proof. For any j and k, let $L_{j,k}$ be the random variable that is equal to 1 if identifiers j and k appeared in a same interaction, 0 otherwise.

We will work on the number of pairs that interacted $N = \sum_{i \in I_n} L_{i,i+1}$. We want to prove that the expectation of this variable is less than $\frac{3}{4}\alpha n$.

$\mathbb{E}(N) = \sum_{i \in I_n} \mathbb{E}(L_{i,i+1}) = \sum_{i \in I_n} \Pr(Id_i \text{ met } Id_{i+1} \text{ after } f(n) \text{ steps})$

From Lemma 1, we deduce that from at least $\frac{\alpha}{2}n$ pairs, each identifier is present on at most $g(n)$ agents, as only at most $\frac{\alpha}{2}n$ identifiers appeared on more than $g(n)$ agents. Hence, for these pairs,

$\Pr(Id_i \text{ met } Id_{i+1} \text{ after one step}) \leq \frac{g(n)(g(n)-1)}{2} \cdot \frac{2}{n(n-1)} \leq \frac{g(n)^2}{n(n-1)}$.

$\Pr(Id_i \text{ met } Id_{i+1} \text{ after } f(n) \text{ steps}) \leq 1 - \left(1 - \frac{g(n)^2}{n(n-1)}\right)^{f(n)}$.

Let $\beta > 1$. The function $l(x) = x^\beta$ is convex (as $l''(x) = \beta(\beta-1)x^{\beta-2} > 0$). The convexity implies that $l(x) \geq l(y) + l'(y)(x-y)$. With $x = (1-X)$ and $y = 1$, we have:

$(1-X)^\beta \geq 1 + \beta(1-X-1) = 1 - \beta X$.

With $X = \frac{g(n)^2}{n(n-1)}$ and $\beta = f(n)$ (as for n large enough, $f(n)$ is always greater than 1), we have the following inequality:

$\left(1 - \frac{g(n)^2}{n(n-1)}\right)^{f(n)} \geq 1 - \frac{f(n)g(n)^2}{n(n-1)}$.

$\Pr(Id_i \text{ met } Id_{i+1} \text{ after } f(n) \text{ steps}) \leq 1 - \left(1 - \frac{g(n)^2}{n(n-1)}\right)^{f(n)} \leq \frac{f(n)g(n)^2}{n(n-1)}$.

As f is n-polylog and g is polylogarithmic, for n large enough, we have $\frac{f(n)g(n)^2}{n(n-1)} \leq \frac{1}{4}$.

If we sum these probabilities for the half number of pairs we considered, we get the upper bound of $\frac{1}{4}\alpha n = \frac{\alpha}{4}n$ expected pairs that appeared in a same interaction.

Even if all the pairs of identifiers of the not considered half of I_n met, the upper bound for the second half provides the result:

$$\mathbb{E}(N) \leq \frac{\alpha}{2}n + \frac{\alpha}{4}n \leq \frac{3}{4}\alpha n$$

E Proof of Theorem 7

To prove this theorem, it suffices to prove the following lemma:

Lemma 7. *For any protocol in CPPL, there exists some $K \in \mathbb{N}$ such as this protocol is in $SPACE(n \log^K n)$.*

Proof. Any configuration can be described on a space $O(n \log n)$ by writing, for each agent, its state and the list of identifiers (using identifier 1 for the first,... n written in binary for the last).

Let $f(n) = n \log^k n$ for some k being a function bounding by above the number of expected interactions to find the output. We will prove that the result holds considering $K = k + 1$.

The idea is to simulate all the sequences of configurations of length $3f(n)$ (i.e. sequence of $3f(n)$ interactions).

With more details: We can write a sequence by providing the order of the pairs that interacted. Hence, we need a space of $3f(n) \log n$ to encode a sequence on a Turing Machine. Going from one sequence to the next does not require more space. We then just simulate the sequence of interactions on the population. Finding, from a configuration, if it has an output or not does not require more space.

For each possible output, we use a counter initialized at 0. Each time we computed a sequence, if the configuration does have an output, we increase the corresponding counter.

When all the sequences have been simulated, we provide the same output that the one that got the highest counter.

From Markov's inequality, we get that the probability to use more that $3f(n)$ interactions is less than one third. Hence, more than the half of the $3f$ sequences must provide the right output. The output of our Machine is the same that the protocol's.

This machine is deterministic and uses a space $O(f(n)\log n) = O(n\log^{k+1} n)$.

F Proof of Theorem 8

The protocol will write the 2nd and 3rd tapes on the first $\log n$ agents, the 4th one on the first $\log^k n$ agents. To initialize it, it will start by a leader election, then will compute the size of the population (getting from this $\log n$).

Let prove that the 6 items can be simulated:

1. It is easy, as the Leader only needs to know on which agent each lecture head is. When the head goes left, the leader finds with an epidemic the previous identifier (same when the head goes right with the next identifier).
2. This one can be performed in a single epidemic.
3. It corresponds to the process described in Section 4 performed on the agents marked.
4. It is a similar process to finding the median identifier, only we work:
   (a) We keep two identifiers Min and Max.
   (b) We take a random agent $Cand$ in $]Min, Max[$.
   (c) We compute the number of agents with identifier smaller than $Cand$.
   (d) Either it is the right one and it is over, either we have an update of Min or Max. In the second case, we go back to step (b) with the new interval.

   We can see that this process will use a logarithmic number of loops in expectation.
5. We perform an epidemic to mark all agents of identifier smaller of the pointing head's.
6. Again, an epidemic is enough.
7. The leader identifiers the $(x+1)$th identifier in the population. It then spreads an epidemic to look for an identifier smaller or equal to this one, as in the median candidate process. As in the median process, each identifier has the same probability to be selected.

   After than, the leader needs to count how many identifiers are smaller to this one (his own excluded). The given number will be chosen in the right interval according to a homogeneous distribution.