Incremental Coding for Extractable Compression in the Context of Massive Random Access
Thomas Maugey, Aline Roumy, Elsa Dupraz, Michel Kieffer

To cite this version:
Thomas Maugey, Aline Roumy, Elsa Dupraz, Michel Kieffer. Incremental Coding for Extractable Compression in the Context of Massive Random Access. IEEE Transactions on Signal and Information Processing over Networks, 2020, 6, pp.251-260. 10.1109/TSIPN.2020.2981263. hal-02542972

HAL Id: hal-02542972
https://inria.hal.science/hal-02542972
Submitted on 15 Apr 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Incremental coding for extractable compression in the context of Massive Random Access

Thomas Maugey, Member, IEEE, Aline Roumy, Member, IEEE, Elsa Dupraz, Member, IEEE, Michel Kieffer, Senior Member, IEEE

Abstract—In this paper, we study the problem of source coding with Massive Random Access (MRA). A set of correlated sources is encoded once for all and stored on a server while a large number of clients access various subsets of these sources. Due to the number of concurrent requests, the server is only able to extract a bitstream from the stored data: no re-encoding can be performed before the transmission of the data requested by the clients.

First, we formally define the MRA framework and propose to model the constraints on the way subsets of sources may be accessed by a navigation graph. We introduce both storage and transmission costs to characterize the performance of MRA. We then propose an Incremental coding Based Extractable Compression (IBEC) scheme. We first show that this scheme is optimal in terms of achievable storage and transmission costs. Second, we propose a practical implementation of our IBEC scheme based on rate-compatible LDPC codes. Experimental results show that our IBEC scheme can almost reach the same transmission costs as in traditional point-to-point source coding schemes, while having a reasonable overhead in terms of storage cost.

Index Terms—Data compression, Source coding, Random Access, Channel coding.

I. INTRODUCTION

In source coding for Massive Random Access (MRA), the data generated by several correlated sources is compressed, stored on a server and later partly requested by many clients. We propose to model the potential client’s requests by a navigation graph that represents the application constraints. For instance, the sources may only be requested in a sequential order. In this setup, the large data volume and the high number of clients prevent re-encoding at the server. This implies that the whole data must be coded before clients can make requests and that only simple data extraction operations are allowed after receiving the requests. We consider the MRA problem from a pure source coding perspective, which differs from problems of massive multiple access to some communication resources [1].

Examples of MRA applications are free-viewpoint television (FTV) [2], access to data collected by a network of sensors [3] or a crowd-sensing platform [4]. In these applications, the items of the database (e.g., the frames of a video, or the frames of the different views in multiview video, or the signals captured by different sensors) show strong correlations with each other. This suggests to jointly compress the sources [5, Th. 2.6.6] to exploit the statistical dependencies, which is unfortunately not compatible with the requirement of having random access. Tile-based approaches [6], [7] divide the sources into subsets, called tiles. Sources within a tile are jointly compressed, such that the correlation between sources can be partly exploited while, at the same time, enabling an access to a subset of sources. However, the random access granularity remains at the tile level, which implies that a lot of unrequested sources are usually transmitted.

Instead, several solutions have been proposed when random access is a key constraint. For instance, in video compression, picture-wise access is necessary, either for editing [8] or for interactive communication [9]. In the schemes of [8], [9], all frames are coded independently (All intra mode). When higher compression performance is targeted (for broadcasting or streaming applications), the sequence is encoded jointly and an access frame (intra coded, i.e., independently of the others) is introduced to enter the decoding of the sequence roughly every second [10]. To allow to resynchronize faster than every second, some frames are encoded and stored twice (once as intra, and once as a complement to the previously coded frames) [11]. Multiple storage of the same frame has also been proposed in the context of interactive communication of multiview videos in [12], [13], [14]. This solution provides an increased accessibility to the video with almost the same transmission rate, but at the expense of an increased storage cost. A tradeoff between storage and transmission costs has been proposed in the context of multiview videos [15][16], where a frame is no more encoded several times. Instead it is stored and sent in a single worst case version (called a Merge frame). Similar techniques have been proposed to allow random access to compressed sensor data. For instance, separated compression of blocks of data, similar to the All Intra video compression, has been proposed for volumetric data [17] and digital power meter [18]. The idea to encode some data points as a reference, as in [11][19], has been proposed for Internet of Things databases [20] and for the genomes [21]. However, these methods generally come with a significant transmission or storage rate penalty.

The goal of this paper is to propose a novel coding ar-
architecture that is competitive in terms of storage and transmission costs while considering the application constraints incorporated into the navigation graph. To do so, we first formally introduce the MRA problem in Sec. II, as a multi-source coding problem where clients are allowed to access successively to some of the sources. Then, we propose to model the possible successive requests by a navigation graph, whose structure is imposed by application constraints. In the navigation graph, a directed edge exists from node $j$ to node $\ell$, if it is possible, for a client, to request source $X_\ell$ after source $X_j$. Hence, when coding a source $X_\ell$, its neighboring sources in the navigation graph might be used to achieve lower compression cost, because one of them is present at the client’s side, i.e., obtained from previous requests. However, exploiting the correlation is a difficult task in the MRA context, because, when coding a given source $X_\ell$, the encoder does not know which sources will be available at the decoder.

Then, we propose to evaluate the compression efficiency of MRA schemes in terms of two costs: the storage cost and the transmission cost, the latter being request-dependent. In Sec. III, we use these two costs to evaluate the coding performance of state-of-the-art methods, when these methods must satisfy the constraints defined by the navigation graph.

In Sec. IV, we propose a novel Incremental coding Based Extractable Compression (IBEC) scheme. For each source $X_\ell$, incremental coding builds a bitstream that enables to decode $X_\ell$ based on any possible source potentially available at the decoder. More precisely, the stored bitstream is composed of different parts that are potentially transmitted depending on the level of correlation between $X_\ell$ and on the previously requested sources present at the client’s side. For a storage cost that is as small as the most performing existing scheme, we show that the transmission cost achieved by the proposed IBEC scheme is the same as the cost that is obtained in the absence of random access. In Sec. IV-B, a practical implementation of the IBEC scheme is proposed based on rate-adaptive Low-Density Parity-Check (LDPC) codes.

Finally, Sec. V describes experimental comparisons of the proposed IBEC scheme with conventional coding architectures.

II. MASSIVE RANDOM ACCESS (MRA)

The source coding problem with MRA is shown in Fig. 1. Consider a database containing the signals $\{X_\ell\}_{1 \leq \ell \leq L}$ generated by $L$ correlated sources denoted $\{X_\ell\}_{1 \leq \ell \leq L}$. Access to this database is made randomly, according to the client’s choice, but also according to some restrictions imposed by the application. In Sec. II-A, we propose to model these restrictions with a navigation graph. The database is first compressed offline into a set of separable bitstreams denoted by $\{b_\ell\}_{1 \leq \ell \leq B}$. Then, a client requests some of the signals $X_\ell$ in the form of a vector $v$ of source indices. Here, the signal $X_\ell$ corresponds to the smallest entity that can be accessed, such that when the client requests the source of index $\ell$, information about all the samples of this source are sent. Note that a real source of information might be split into several sources, to offer more flexibility in the access to the database. Upon reception of a request, the server extracts part of the stored bitstream and sends it to the client for decoding.

In Sec. II-B, we formally define two criteria to characterize the coding performance of a solution to the MRA problem, namely the storage and rate costs $S$ and $R$. For ease of presentation, we consider in this work a lossless compression scheme. A lossy extension may be obtained, e.g., with a quantization of the input sources [22].

A. Modeling Random Access by means of a navigation graph

Access to a database is usually proposed with some restrictions. For instance, in FTV, the client observes a scene by navigating from one viewpoint to another. But, to offer a smooth client experience, the navigation might be limited to neighboring viewpoints only. In the sensor network application, a client might be allowed to only request signals that have been acquired in some spatio-temporal windows.

Before showing how to integrate these restrictions, we first model a request as a vector of source indices. Indeed, in FTV, a frame of one view is modeled as the data of a source, and the navigation of the client is therefore equivalent to a request of ordered source indices. In the case of sensor networks, a source corresponds to the data of one sensor in a temporal window. Again, the request of a spatio-temporal window can be seen as a sequence of source indices, up to an ordering of the requested sources. This ordering can either be predefined (e.g., navigation to the east and the south only) or optimized. In the latter case, the ordering needs to be sent with the data.

Now, to describe the set of allowed requests to the database that may be performed by a client, we introduce the oriented navigation graph $G = (N, E)$. $N$ is a set of $L + 1$ nodes and $E$ is a set of directed edges between these nodes. The nodes represent the $L$ sources plus a (dummy) source $X_0$ used to initiate the navigation. A directed edge $e_{j,\ell}$ from node $j$ to

---

**Fig. 1.** The Massive Random Access system: $L$ signals are jointly compressed and stored on a server, a client requests a sequent of source indices $v$, the server extracts part of the stored bitstream and sends it to the client for decoding.
node $i$ indicates that the source $X_i$ can be accessed by a client, once $X_i$ has been previously requested and stored in the clients’ memory. The only source that may be directly accessed, without having previously requested another source, is the (dummy) source $X_0$ which corresponds to node 0. To summarize, the graph $G$ introduces the constraints on the way sources may be accessed. The set of all possible requests consistent with $G$ is denoted by $V$.

\section*{B. Rate and storage costs}

To reduce as much as possible the amount of data required to serve a client’s request, a wise coding scheme must exploit the correlation between the requested sources. Based on the navigation constraints introduced above, we know that for a given requested source of index $i$, at least one of its neighbors in the graph $G$ has been already accessed and can serve as a helper for decoding. We can thus model the request $v = [v_1, v_2, \ldots]$ as a tree whose root is the node 0, and the edge indicates that the preceding source helps the decoding of the successive source (see Fig. 2). In general, there are several ways to extract and decode the data for a given request. For example, when $v_j$ has two parents $v_{i_1}$ and $v_{i_2}$, both previously requested, the server can either extract the compressed bitstream of the source $X_{v_j}$ with either $X_{v_{i_1}}$ or $X_{v_{i_2}}$ taken as a reference. To avoid any ambiguity, the neighbor used for the extraction is sent by the client together with the request. In the rest of the paper, $\pi_v(v_i)$ denotes the parent of the source of index $v_i$ used in the extraction.

As described in Fig. 1, the aim of the offline compression is to compute a set of bitstreams representing the data generated by the $L$ sources. Let $B = \{b_i\}_{1 \leq i \leq B}$ be the resulting set of $B$ independently extractable bitstreams. A subset of necessary bitstreams will be transmitted as a response to each client’s request $v$. Several compression algorithms exist that satisfy these constraints (see Sec. III for a detailed presentation). To compare these schemes, we first introduce the storage cost, which represents the size of the bitstreams stored on the server:

\begin{equation}
S = \frac{1}{L} \sum_{i=1}^{B} |b_i|, \tag{1}
\end{equation}

where $|.|$ denotes the sub-stream size expressed in number of bits, and where the normalization factor $L$ is in order to have a cost per source.

The second criterion used in order to compare compression schemes is related to the efficiency of serving requests. Indeed, at the other side of the chain, the client sends its request $v = [v_1, \ldots, v_{|v|}]$ to the server. The server then extracts from $B$, the bitstreams $\{b_{i}\}_{i \in \mathcal{I}(v)}$ necessary to recover the signals requested in $v$. The set $\mathcal{I}(v)$ contains the indices of the $b_i$ that are necessary to recover the sources whose index is in $v$.

The \textit{per request transmission cost} is defined as the cumulated size of the bitstreams sent to a client for a given request:

\begin{equation}
R(v) = \frac{1}{|v|} \sum_{i \in \mathcal{I}(v)} |b_i|, \tag{2}
\end{equation}

where the normalization leads to a per source criterion. Finally, to obtain a criterion that does not depend on the client’s request, we assume that a probability distribution $p$ over the clients’ requests is available. This leads to the expected transmission cost:

\begin{equation}
R = \mathbb{E}_v(R(v)) = \sum_{v \in V} p(v) R(v). \tag{3}
\end{equation}

\section*{III. Conventional architecture limitations}

Building a coding scheme that solves the MRA problem consists in finding an off-line encoding function able to generate the bitstreams $\{b_i\}_{1 \leq i \leq B}$ associated to the source realizations $\{X_i\}_{1 \leq i \leq L}$, and an online extraction function, able to select sub-streams $\{b_i\}_{i \in \mathcal{I}(v)}$ that satisfy any request $v \in V$.

This section provides the costs achieved by conventional coding and extraction schemes in the context of MRA (summarized in Fig. 3). For that purpose, we introduce the following notations $\forall i \in [1, L], \forall j \in [0, L]$:

\begin{equation}
h_{i,j} = H(X_i^n_i | X_j^{n_j}) \quad \text{and} \quad h_i = H(X_i^{n_i}), \tag{4}
\end{equation}

where $n_i$ and $n_j$ stand for the signal length of the sources $X_i$ and $X_j$ respectively, $X_i^{n_i}$ denotes the random vector $(X_{i,1}, X_{i,2}, \ldots, X_{i,n_i})$, and $H(X)$ and $H(X | Y)$ denote the entropy and the conditional entropy respectively. We also recall that $\forall i$, $H(X_i | X_0) = H(X_i)$, since $X_0$ is a dummy source used for initialization only.

A first solution to compress the database while allowing various access to subsets of sources is to code each source \textit{independently}. In this case, there is one $b_i$ per source, and $B = L$. This scheme is called All Intra (AI). It has been
proposed to facilitate either picture-wise editing of compressed videos [8], or picture-wise access in the context of interactive video communication [9]. The interest of this method resides in the fact that it is totally flexible, in the sense that it can straightforwardly satisfy any client request. The drawback of this solution is that the correlation between the sources is not exploited. More precisely, the storage cost achieved by the AI scheme is:

\[ S_{AI} = \frac{1}{L} \sum_{i=1}^{L} h_i \]  

and the transmission cost is:

\[ R_{AI} = \sum_{v \in V} p(v) \frac{1}{|V|} \sum_{i \in v} h_i. \]  

Therefore, the AI scheme achieves low compression performance, both in terms of storage and transmission, especially when the sources are strongly correlated.

Alternatively, one can exploit the correlation between a source and the previously requested ones, i.e., the parents in the navigation graph \( G \), at the coding stage. This is for example the case of the predictive coding scheme: if \( x_j \) is in memory when decoding a vector \( x_i \) (because it has been already requested), the predictive encoder first builds a prediction \( \hat{x}_{ij} \) of the signal \( x_i \), computes the residue \( x_i - \hat{x}_{ij} \) and stores the coded residue as a bitstream \( b_i' \). If the prediction is good, the compression performance of such a scheme is high. However, the computed residue depends explicitly on the reference source used to build the prediction (\( X_j \) in the previous example), and does not allow random access. Thus, to use predictive coding in the MRA context, one needs to store on the server, for each source \( X_i \), one residue per possible parent \( j \) of node \( i \) in the graph \( G \). This scheme is thus called the Multiple Prediction (MP) scheme [11], [12], [13], [14], and the storage cost is:

\[ S_{MP} = \frac{1}{L} \sum_{i=1}^{L} \sum_{j \in \pi_j, e \in E} h_{ij}. \]  

When serving a request \( v \), only the useful residues are sent to the client. The transmission cost is thus:

\[ R_{MP} = \sum_{v \in V} p(v) \frac{1}{|V|} \sum_{i \in v} h_i |\pi_v(i)|. \]  

Since \( h_{ij} \leq h_i \), the transmission cost is reduced with respect to the AI scheme, and the cost reduction increases with the correlation of the requested sources. On the other hand, the storage cost increases significantly with the averaged degree of the graph, i.e., with the flexibility offered to the client to navigate within the database.

In the Compound (C) scheme proposed in [15], [23], for each source \( x_i \), all the possible predictions \( \hat{x}_{ij} \) are built (with \( e_{ji} \in E \)). Then, considering \( \hat{x}_{ij} \) as noisy versions of \( x_i \), a bitstream \( b_i \) able to correct the errors in all the predictions is generated (for example based on channel coders). In order to tackle all the predictions possibly generated by the clients, the stored bitstream corresponds to the worst one, i.e., to the largest \( h_{ij} \). The storage cost is thus:

\[ S_{C} = \frac{1}{L} \sum_{i=1}^{L} \max_{j, e_{ji} \in E} h_{ij}. \]  

Similarly to the AI scheme, the extraction is simple since there is only one bitstream per source. The transmission cost is thus:

\[ R_{C} = \sum_{v \in V} p(v) \frac{1}{|V|} \sum_{i \in v} \max_{j, e_{ji} \in E} h_{ij}. \]  

The C scheme is thus a good way of achieving reasonable transmission cost (between \( R_{AI} \) and \( R_{MP} \)) while having a smaller storage cost than those of MP and AI schemes. However, while it is reasonable to consider that one needs to anticipate the worst scenario at the server side, this is unfortunate to transmit this bitstream in all the cases, even if the prediction is of a better quality.

The storage and transmission costs are summarized in Fig. 4. In the next section, we introduce the proposed coder that enables to reach the transmission cost of the MP scheme and storage cost of the C scheme.
In Sec. III and IV, the proposed IBEC architectures, i.e., smallest storage and transmission costs of all the conventional schemes are illustrated in Fig. 3. They are then sorted from the best to the worst \( (IBEC) \). Incremental coding based Extractable Compression (IBEC).

A. Proposed coding scheme

In [24], [25], we have shown that when compressing one single source with several potential side informations available at the decoder, it is possible to use incremental coding in order to only send the necessary amount of bits. Based on the graph-based client’s navigation formalism and the transmission/storage costs proposed in Sec. II, we now generalize this result to the multi-source scenario. We call this new coding scheme Incremental coding Based Extractable Compression (IBEC).

For each signal \( x_i \) to be compressed, we first identify the parents of the source of index \( i \) in the navigation graph \( G \). These neighbors are used to build potential predictions \( x_{ij} \). They are then sorted from the best to the worst (i.e., from the smallest \( h_{ij} \) to the largest). Then, we build a first bitstream \( b_1 \) able to decode the best prediction assuming that \( x_{ij} \) is already decoded at the receiver. For the second bitstream, the coding scheme is able to use \( b_1 \) plus an additional bitstream \( b_2 \) of size \( h_{ij} \). This incremental construction is applied in the same way to all predictions. As a result, the stored bitstream, for each source, has the same size as the \( C \) scheme, i.e., the one corresponding to the highest \( h_{ij} \), but is split into several sub-streams so that only the necessary information can be extracted. All schemes are illustrated in Fig. 3.

The global performance of the IBEC scheme is:

\[
S_{IBEC} = \frac{1}{L} \sum_{j=1}^{L} \max_{j \in \mathcal{E}} h_{ij},
\]

and

\[
R_{IBEC} = \sum_{v \in V} p(v) \frac{1}{|V|} \sum_{i \in v} h_{i|\pi_v(i)}.
\]

As it can be seen from Fig. 4, based on the costs, derived in Sec. III and IV, the proposed IBEC scheme obtains the smallest storage and transmission costs of all the conventional architectures, i.e.,

\[
S_{IBEC} = S_C \leq S_{AI} \leq S_{MP}
\]

and

\[
R_{IBEC} = R_{MP} \leq R_{C} \leq R_{AI}.
\]

In the next section, we explain how the incremental coding is built in practice.

B. Practical incremental coder

**Incremental coding principle:** The source coding problem with one source and one side information at the decoder can be solved in practice by channel codes [26], [27], [28]. Similarly, to solve the MRA problem, we propose to construct a coding scheme based on channel codes. However, the channel code needs to tolerate variable rate to adapt to all the potential side informations. In practice, rate adaptation is achieved by choosing a rate among a finite set of predefined source coding rates: \( R \in \{ \frac{1}{M}, \ldots, \frac{m}{M}, \ldots, \frac{M}{M} \} \).

Assume that the decoder requests the source \( X_i \), and has previously requested the source \( X_j \), with \( j \in \{ j_1, ..., j_J \} \), see the navigation graph of Fig. 5(a). Note that the size \( J \) of the neighborhood depends on the node \( i \), but for ease of presentation, we remove the dependence with respect to \( i \) in the notation \( J \). Let us further assume that the sources \( X_j \), with \( j \in \{ j_1, ..., j_J \} \) are sorted in increasing order of conditional entropy, i.e., from the most to the least correlated source \( X_j \),

\[
h_{i|j_1} \leq h_{i|j_2} \leq \ldots \leq h_{i|j_J}.
\]

We now explain how to encode the source vector \( x_i \) into an extractable bitstream \( b_1 = (b_1^{j_1}, b_1^{j_2}, \ldots, b_1^{j_J}) \).

**Data encoding:** let us consider that all correlated sources are marginally i.i.d., binary with uniform distribution, and that each source \( X_i \) generates a vector of length \( n \). We model the pairwise correlation between the correlated sources by a channel with transition probability \( p(x_i|x_j) \), see Fig. 5(b). Further assume that the correlation channel is a binary symmetric channel. We use the rate-adaptive code called Low Density Parity Check Accumulate (LDPCA) Code introduced in [29]. Given a set of predefined target rates \( \{ \frac{1}{M}, \ldots, \frac{m}{M}, \ldots, \frac{M}{M} \} \) and a source vector length \( n \), the LDPCA construction provides \( M \) parity check matrices denoted \( (K_1, \ldots, K_m, \ldots, K_M) \), where \( K_m \) is of size \( n \frac{m}{M} \times n \) and where

\[
\forall x, K_1x \subseteq K_2x \subseteq \ldots \subseteq K_Mx
\]

meaning that \( K_1x \) is a subvector of the vector \( K_2x \). In our simulations, we considered the 6336_irregDeg2to21 LDPCA code, whose parameters are available at [30].

We now explain how to encode the source vector \( x_i \). First, the so-called accumulated syndromes are computed as

\[
\forall m, \ a_{i,m} = K_mx_i,
\]

where \( a_{i,m} \) is of length \( n \frac{m}{M} \). Then, for each possible side information \( x_j \), and for each accumulated syndrome \( a_{i,m} \), a reconstruction is performed according to the maximum \textit{a posteriori} criterion, i.e., \( \forall j \in \{ j_1, ..., j_J \}, \forall m \in \{ 1, ..., M \} \)

\[
\hat{x}_{i,j,m} = \arg \max_{x, a_{i,m} = K_mx} p(x_i|x_j).
\]

Note that this defines a modified channel decoder, since the search space is the coset of syndrome \( a_{i,m} \) and not the coset.
of syndrome $\mathbf{0}$, as in classical channel coding. When a LDPC code is used, decoding is performed with the modified belief propagation (BP) algorithm proposed in [31]. Then, for each possible side information $x_j$, we select the shortest accumulated syndrome $a_i^j$ such that the BP decoder recovers $x_i$ perfectly, i.e., $\forall j \in \{j_1, \ldots, j_J\}$, $\forall m \in \{1, \ldots, M\}$

$$m^*(j) = \arg \min_m \{a_{i,m} = \frac{n}{M} s.t. \hat{x}_{i,j,m} = x_i\} \quad (19a)$$

$$a_i^j = a_{i,m^*(j)} \quad (19b)$$

From the inclusion property of the accumulated syndromes (16), and from the ordering of the side information vectors (15), the optimal accumulated syndromes satisfy $a_i^1 \subseteq a_i^2 \subseteq \cdots \subseteq a_i^j$.

Finally, for the source $X_i$, the stored sequence of bitstreams $b_i = (b_i^1, b_i^2, \ldots, b_i^j)$ is constructed from the $a_i^j$ as follows. First, $b_i^1 = a_i^1$. Then, the second bitstream $b_i^2$ is obtained by retaining the bits in $a_i^2$ that are not in $a_i^1$, i.e., $b_i^2 = a_i^2 \setminus a_i^1$. More generally, we have $b_i^k = a_i^k \setminus a_i^{k-1}$.

The resulting storage cost for the source $X_i$ is $S_i = |a_i^j|/n$, where the overall storage cost is $S = \sum_{i=1}^{n} S_i$.

Data extraction: Upon request of the source $X_i$, and knowing that the source $X_j$ is available at the decoder, the server extracts from $b_i$ the subsequence $(b_i^j, b_i^2, \ldots, b_i^1) = a_i^j$, and sends it to the decoder. This leads to a transmission cost $R_i^j = |a_i^j|/n$. Then, the transmission cost of a request, $v = (v_1, \ldots, v_M)$ is

$$R(v) = \frac{1}{|v|} \sum_{i \in v} R_i^j \quad (20)$$

Data decoding: Upon request of the source $X_i$, the decoder receives $(b_i^j, b_i^2, \ldots, b_i^1) = a_i^j$. The decoder then performs BP decoding taking into the previously received side information $x_j$. From the rate adaptation performed at the encoder (19), the reconstruction is performed without any error.

IBEC vs C scheme: The C scheme shares similarities with the proposed IBEC scheme since, in both cases, a channel code is used to perform data encoding. In the IBEC scheme, the index of the previous request $j$ is used to adapt the transmission and send the complement information only, as shown in (19). In the C scheme, this knowledge is not used. The sent accumulated syndrome is the one that allows perfect reconstruction for any possible side information (21a), in particular for the worst one (21b).

$$m^* = \arg \min_m \{a_{i,m} = \frac{n}{M} s.t. \forall j \hat{x}_{i,j,m} = x_i\} \quad (21a)$$

$$= \arg \min_m \{a_{i,m} = \frac{n}{M} s.t. \hat{x}_{i,j,m} = x_i\} \quad (21b)$$

$$\forall j, a_i^j = a_{i,m^*} \quad (21c)$$

**IBEC and C schemes vs MP:** The IBEC and C schemes use channel coding to perform data encoding. By contrast, in the MP scheme all possible residues $x_i - x_{ij}$, $\forall (i, j)$ are encoded with a variable length source code and then stored. Upon request of the source of index $i$, after having requested the source of index $j$, only the compressed bitstream of $\hat{x}_i - x_{ij}$ is sent.

**V. Performance comparison**

**Datasets:** In order to evaluate the performance of the proposed incremental scheme compared to the baseline methods reviewed in Section III, we generate five different datasets according to Algorithm 1. We pick one graph structure described by an adjacency matrix $\mathbf{A}$ (available at [32]). Depending on the cases, this graph structure is randomly generated or corresponds to a predefined structure. To generate the sources $X_i$, $L$ independent random vectors $\mathbf{z}_i$ of size $n = 12672$ are first generated, forming a matrix $\mathbf{Z}$ of size $L \times n$. Then, in order to generate correlated sources, we iteratively perform a filtering of $\mathbf{Z}$ with the following filter $\mathbf{M} = \mathbf{D}^{-1/2}(\mathbf{A} + \mathbf{I}_L)\mathbf{D}^{-1/2}$, as in the distributed averaging method [33]. After a large number of iterations, the sources will converge to a consensus, where all sources have the same value equal to the data average. Since it is not our goal here, we intentionally perform less iterations in order to bring some correlation between the sources, without reaching complete consensus though. The variable $\mathbf{Z}$ is finally binarized in order to build the data $\mathbf{X}$ to encode.

We considered various number of iterations $N_{it}$ to generate sources with different characteristics, which correspond to a variety of scenarios that are encountered when compressing correlated data (e.g., averaged connectivity degree and the heterogeneity of the correlations). They are depicted in Figure 6. The 2D grid graph represents data acquired by sensors arranged in a regular 2D grid, where each source is connected to its direct neighbors. This graph models a navigation encountered in Free Viewpoint Television, where the navigation is performed among multiview videos. Here one node corresponds to one frame and the sources placed...
on the same line as the frames of the same view. Moreover, the coding rates of the sources are heterogeneous as classically observed in video compression. The graph Community Network has 8 clusters, in which nodes are highly connected and correlated between each other. Some sources are also connected to sources in other clusters. The correlation is lower. The graph Sphere is similar to 2D grid, with heterogeneous levels of correlation. However, the averaged connectivity degree is higher. The graph Spiral has a very small averaged connectivity degree. In other words, the randomness of the navigation is reduced. The graph Torus has homogeneous level of correlation and averaged connectivity degree.

**Data:** $n, N_{it}, A$

**Result:** $x_t, P, H$

**Initialization:**

Generate $Z = [z_{i,t}]_{i \leq L, t \leq n}$; $L \times n$ matrix whose rows are $L$ independent random sources with the distribution $N(0, 1)$ of size $n$;

Build the filter matrix $M = D^{-1/2}(A + I_L)D^{-1/2}$, where $D$ is the degree matrix of $A + I_L$;

for $i \in [1, N_{it}]$ do
  $Z = MZ$
end

Build a binary variable:

$X = (Z > 0)$;

Estimate the entropy rates $h_{t_i|t_2}$ between each source and its neighbors in the graph:

for $t_1 \in [1, L]$ do
  $h_{t_1} = 1$;
  for $t_2 \in [1, L]$ s.t. $e_{t_2, t_1} \in E$ do
    Estimate the error probability:
    $p_{t_1|t_2} = p_{err}(x_{t_1}|x_{t_2})$;
    Calculate the entropy:
    $h_{t_1|t_2} = -p_{t_1|t_2} \log_2 p_{t_1|t_2} + (1 - p_{t_1|t_2}) \log_2 (1 - p_{t_1|t_2})$;
  end
end

**Algorithm 1:** Data generation

**Client’s request generation:** In the following, we present storage and transmission costs evaluated on the five graphs generated as explained above. In order to evaluate the transmission cost, that is by nature, client-dependent, one need to simulate different client’s requests, evaluate their transmission cost for the different schemes and average them in order to obtain the expected transmission cost in eq. (3). As explained in Section II, a client’s request is a tree $\subset G$. We thus generate random navigation trees of different lengths. For that purpose, we assume that all the transitions between sources are equally probable.

**Results:** We have implemented the AI, MP, C schemes and our proposed IBEC. For AI, MP schemes the coding of the source or the residue after prediction is done using an arithmetic coder. For C scheme, the encoding is done with the 6336_irregDeg2to21 LDPCA code available at [30] (as in (19), among the set of codes, we choose the LDPC code with minimum rate that allows perfect reconstruction of the vector).

For each graph, we have encoded the corresponding data $X$, and simulated 100 client’s navigation, recording, each time, the transmission cost. We have also calculated, for each case, the theoretical expected performance, based on the entropy calculation, see eq. (5,6) for AI, eq. (7,8) for MP, eq. (9,10) for $C$ and eq. (11,12) for IBEC. The results are shown in Figure 7. In this figure, the storage cost is on the left while transmission costs are depicted on the right. For each method, the empty bars correspond to the theoretical performances while the filled ones represent the experimental ones.

The IBEC scheme theoretically achieves the smallest storage (13) and transmission (14) costs. Said differently, our IBEC scheme achieves the smallest transmission rate as MP scheme while reaching also the smallest storage cost as the $C$ scheme, which validates its potential advantage. Let us now discuss the practical performance.

The difference between theoretical and practical costs is small for the AI and MP schemes since they use an arithmetic coder whose performance is not far from the Shannon bounds. On the contrary, the channel codes used in the C scheme and by the incremental coders the IBEC scheme have a more significant gap between theory and practice. Nevertheless, despite this disadvantage, the practical performances comparison in Fig. 7 still demonstrate the benefits of our scheme. This is indeed visible from the following observations.

- Compared to the AI scheme, both storage and transmission costs are always smaller with the IBEC scheme. For the dataset Torus, the gap is lower since most $h_{i,j}$ are around 0.5 bits per symbol as shown in Fig. 6 (e), which results in a practical bitrate closer to 1, that is the cost of an intra source coding. In all the other datasets, the correlation between the sources is much higher, and the results demonstrate the ability of the IBEC scheme to take into account this correlation at the storage and transmission stages.

- Given the fact that the MP scheme reaches the best transmission rate possible (thanks to an extensive storage cost), we observe that our IBEC scheme is really efficient. Indeed, the transmission rate achieved by the IBEC scheme is almost the same (or slightly higher) than the MP scheme, for a storage cost that is much lower. This is even more visible when the navigation graph $G$ is highly connected (as the dataset Community Network). To be able to reach the best transmission cost, the MP scheme has to store many residues, exploding the storage cost, while this storage cost remains small with our scheme.

- Instead of storing any possible navigation transitions, the C scheme stores for each source, the worst one, as the IBEC scheme does. This is the reason why the storage costs of the C and IBEC schemes are the same (and the minimum ones) in the theoretical and practical performances. However, instead of transmitting the whole codeword for every request as the C scheme, our IBEC scheme only transmits the necessary subpart. This leads to reduced transmission costs as can be seen in Fig. 7. The difference is not large for the datasets where the correlation between the sources is homogeneous as the Torus one (see Fig. 6 (e)). Indeed, the gap between
Fig. 6. Different graphs $G$ used for experiments. The blue nodes are the sources, and the green node corresponds to $X_0$. The adjacency matrix $A$ of each graph is available in [32].
As a conclusion, in both theoretical and practical aspects, one observes that the IBEC scheme reaches or is very close to the best expected transmission costs, with a minimal storage cost. However, at the same time, the transmission costs explode whatever the size of the tile. It confirms the interest of our IBEC approach, since it enables an ideal transmission cost, perfectly aligned with what the client requests, for a reasonable storage overhead.

VI. CONCLUSION

This paper proposes a formulation of the Massive Random Access problem in which the client’s constraints are modeled by a navigation graph. This enables an original analysis of the existing coding scheme performance. In a second time, a new coding scheme based on incremental coding is proposed. The experimental results demonstrate that our method outperforms those of the literature by minimizing storage and transmission costs at the same time.

REFERENCES
