Joint Source-Network Coding for Large-Scale Sensor Networks

Susana B. Cruz, Gerhard Maierbacher, João Barros
Instituto de Telecomunicações
Faculdade de Engenharia da Universidade do Porto
4200-465 Porto, Portugal
{susana.bulas.cruz, gerhard, jjbarros}@fe.up.pt

Abstract—A modular system architecture based on separate compression and network coding is known to be theoretically suboptimal for relevant classes of sensor networks with correlated sources. Motivated by this observation, we present a feasible solution for joint source and network coding with distortion constraints. By choosing encoders that are simple scalar index assignments, we are able to move the complexity to the destination decoder. Given the network topology and the correlation structure of the data, our algorithms solve the problem of finding encoder and decoder instances that minimize the mean square error of every sample. A proof-of-concept and the complexity analysis of the proposed algorithms underline the effectiveness of our factor graph approach. The presented schemes are shown to yield low-distortion estimates of the collected data even in scenarios where a modular solution would fail.

I. INTRODUCTION

Distributed compression and cooperative forwarding are key towards ensuring the energy efficiency of sensor networks. In scenarios where the sensor nodes are monitoring a confined area, their measurements are likely to be correlated. Distributed source coding techniques [1][2] therefore allow us to reduce the amount of data to be transmitted. On the other hand, network coding [3] has proven to be a key enabler towards maximizing the throughput in networks with one or more data sources and multiple sinks. Although distributed source coding and network coding can be carried out separately, this approach is shown to be suboptimal in general networks with correlated sources [4]. The obvious solution from an information-theoretic point of view is to opt for joint source-network coding, which in practice is rather challenging mostly due to the incurred computational complexity [5].

First attempts to provide practical coding solutions for these problems (e.g. [6]) are mostly of sub-optimal nature and only work for a small number of encoders. Previous work on joint source-network decoding (see [7]) has shown that it is possible to exploit knowledge about the network topology and the correlation structure of the data to find a decoder capable of trading off complexity and end-to-end distortion. In this paper, we capitalize on this approach to provide a complete solution for joint source-network coding. It is assumed that the topology of the considered sensor network scenario is known and that the source statistics are represented by an appropriate factor graph [8]. The key challenge is to find a general graphical representation that includes not only the statistical dependencies that are intrinsic to the data but also takes into account the network coding operations that occur during transmission across the network. The combination of these two aspects lead to a global factor graph for source-network coding, which can be used for an adequate and efficient decoder implementation. The question is how to find such a factor graph in reasonable time.

Our main contribution is thus a class of joint source-network codes that are suitable for large-scale networks and can be implemented with reasonable complexity. To construct the corresponding factor graph, we need adequate flow rates for multicasting information. For that purpose, we build on the subgraph construction method of [9], which is applicable to two sources only, and devise an algorithm that is capable of dealing with an arbitrarily large set of correlated sources. The key is to combine the subgraph construction step with a source optimized-clustering method. Three different coding strategies are analyzed with particular emphasis on a systematic coding scheme that is also part of this paper’s contribution.

The rest of this paper is organized as follows. Section II describes the system model and the problem under consideration. Section III characterizes three strategies to use in the design of the network coding functions. The decoding model is presented in Section IV. Section V offers a performance analysis followed by Section VI containing the conclusions.

II. SYSTEM SETUP AND PROBLEM FORMULATION

A. Source Model

We consider a system setup, illustrated in Fig. 1, with \( S \) correlated sources \( U_1, U_2, \ldots, U_S \). Each of those sources, indexed by \( s \in S, S = \{1, 2, \ldots, S\} \), produces a continuous-valued source sample \( u_s(t) \) at time instant \( t \). To simplify the approach with respect to the source model, we consider one time instant only, dropping the time index \( t \) by considering only the spatial correlations between measurements and not their temporal dependence. Our code construction can however...
be extended to account also for correlation in time. The output symbols are collected in the vector $u = (u_1, u_2, \cdots, u_S)^T$, $u \in \mathbb{R}^S$. We assume that the sources are Gaussian distributed. The individual source samples $u_n$, $n \in S$ have zero mean $E\{u_n\} = 0$, unit variance $\text{Cov}\{u_n, u_m\} = 1$ and are correlated with $u_m$, $m \neq n$, $m \in S$, according to the correlation coefficient $\rho_{n,m} = \text{Cov}\{u_n, u_m\}$.

**B. Encoder Setup**

The observed source samples $u_S \in \mathbb{R}$ are quantized onto the quantization indices $i_s \in \mathcal{I}_s$, $\mathcal{I}_s = \{0, 1, \ldots, |\mathcal{I}_s| - 1\}$. A non-uniform Lloyd-Max quantizer is used to minimize the distortion since it determines the optimum decision and reconstruction levels when the optimization criterion used is the MSE (mean square error) [10]. The obtained quantization index $i_s$ is associated with the reconstruction level $\tilde{u}_{s,i_s} \in \mathcal{U}_s$, $\mathcal{U}_s = \{\tilde{u}_{s,0}, \tilde{u}_{s,1}, \ldots, \tilde{u}_{s,|\mathcal{I}_s|-1}\}$, which, due to the usage of a Lloyd-Max quantizer, corresponds to the centroid of the quantization region.

In each encoder the quantization index $i_s \in \mathcal{I}_s$ is mapped onto the codeword $w_n \in \mathcal{W}_s$, $\mathcal{W}_s = \{0, 1, \ldots, |\mathcal{W}_s| - 1\}$, by the mapping function $a_s : \mathcal{I}_s \rightarrow \mathcal{W}_s$ such that $w_s = a_s(i_s)$. By choosing codeword alphabets $\mathcal{W}_1$, $\mathcal{W}_2$, $\ldots$, $\mathcal{W}_S$ smaller than the number of quantization levels in $\mathcal{I}_1$, $\mathcal{I}_2$, $\ldots$, $\mathcal{I}_S$, i.e. $|\mathcal{W}_s| < |\mathcal{I}_s|$, data compression can be achieved.

**C. Network**

We consider a network with $N$ nodes, represented by the directed graph $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{v_n : n \in N\}$ is the set of nodes $v_n$ uniquely identified by the indices $n \in N = \{1, 2, \cdots, N\}$ and $\mathcal{E}$ is the set of directed edges $e_{kl} = (v_k, v_l)$, where $k \in N$ identifies the start node $v_k$ and $l \in N$, $l \neq k$, identifies the destination node $v_l$. Each edge has a capacity $c_{kl}$ that is the maximum amount of data that may be sent through that edge from the node $v_k$ to the node $v_l$ in each time unit. In the network there are source nodes, sink nodes and ordinary network nodes, possibly participating in the data transmission from the sources to the sinks. As defined before, the set $S \subseteq N$ identifies the source nodes within the network. The set of sink nodes is identified by the set $T \subseteq N$, $T = \{1, 2, \cdots, T\}$. In this work, we assume that every node in the network is able to perform coding operations and can be used to convey data from the sources to the sinks. Each sink contains a decoder that seeks to recover the original data sent by all the sources.

**D. Decoder**

We assume that the network nodes have coding capabilities, i.e. the packets that leave each source node, can be combined in the intermediate nodes to produce a new packet. Thus, the vectors of messages $m_1$, $m_2$, $\ldots$, $m_T$ that arrive at each of the $T$ decoders at the respective sinks can have different alphabet sizes, respectively $M_1$, $M_2$, $\ldots$, $M_T$, as shown in Fig. 1. The vector of messages $m_t$, $t \in T$, which consists in general of several messages, is a function of the codewords $w$.

The decoder has to perform decoding operations to reverse both the coding operations performed in the source encoders and the network coding operations. Since separation between compression and network coding is not always possible [4], we will use a joint source-network coding scheme. Naturally, the joint source-network decoding approach results in a high decoder complexity [5].

Regarding each of the $T$ sinks, we assume that the decoder at the considered sink node has: (i) access to the received packets $m_i(w)$, (ii) a-priori knowledge about the source statistics $p(u)$ and (iii) knowledge about which network nodes and edges were traversed by the received packets as well as the traversed nodes’ coding functions and the traversed edges’ transition probabilities. The latter can be disseminated a priori or sent along in the packet header.

We consider the MSE $E\{|\hat{U} - U|^2\}$ between the estimates $\hat{U} = (\hat{U}_1, \hat{U}_2, \ldots, \hat{U}_N)^T$ and source samples $U = (U_1, U_2, \ldots, U_N)^T$ as the fidelity criterion to be minimized by the decoder. The optimal estimate $\hat{u}_n(m_i(w))$ for a message vector $m_i(w)$ can be obtained by conditional mean estimation (CME) such that

$$\hat{u}_n(m_i(w)) = \sum_{i_n=0}^{\max(i_n)} \hat{u}_{n,i_n} \cdot p(i_n|m_i(w)), \quad (1)$$

which has a high complexity (for details please refer to [10]). The equation only holds because $\tilde{u}_{n,i_n}$ is the centroid of the quantization region, which holds for Lloyd-Max quantizers.

**E. Problem Statement**

Under the system model described above, our main goal is to provide algorithms that yield feasible encoders and decoders for joint source-network coding problems in large-scale sensor networks. As pointed out, the decoder for these scenarios is usually characterized by its high complexity [5]. Extending the work in [8] and [10], we intend to use graphical models to represent our source and network scenarios. By constructing factor graphs where we can run the sum-product algorithm [11], the complexity can be made, depending on the system parameters, to grow linearly with $N$. Based on previous work on practical joint source-network decoding [7], our aim is to devise a graphical source-network coding model that can be directly used for an efficient decoder implementation. The main focus of this work is to derive methods on how to...
construct those graphical models in an efficient way. We shall evaluate the decoder performance by implementing and testing a working prototype.

III. CODE DESIGN

The coding operations performed at the encoders and intermediate network nodes have to be known. Let \( h \) be the coding function, it has for inputs the values of the incoming edges of the node where the operation is performed, represented by \( k_1, k_2, ..., k_i \). Each of these values has alphabet sizes \( l_1, l_2, ..., l_i \), respectively. The result of performing \( h \) is the value \( k_0 \) that the node forwards to its outgoing edges and has a number of levels represented by \( l_0 \), such that

\[
R = h(k_1, k_2, \ldots, k_i), k_0 \in \{0, 1, \ldots, l_0 - 1\} \quad \text{and} \quad k_j \in \{1, 2, \ldots, l_j\}, j \in \{1, 2, \ldots, i\}.
\]

Although the design of those coding functions is not a decoder specific problem, we propose three different coding strategies.

If \( h \) is a random function, each combination of its input values is determined randomly between 0 and \( l_0 - 1 \). For small alphabet sizes it can lead to situations we want to avoid, such as having output values that are never used or having a high variation in the distribution of the output values.

If \( h \) is the sequential function, each combination of the input values is determined in a sequential way starting at 1 and adding one in each value. Afterwards it is calculated each value modulo the maximum output value. The function \( h \) can be represented by the operation \( p \mod l_0 \), being \( p \) the linear index of the mapping table, such that \( p \in \{1, 2, \ldots, l_1 \cdot l_2 \cdot \ldots \cdot l_i\} \). The advantage is that it guarantees a close to uniform distribution of the output values, preventing wasted bits.

The joint source-network coding mapping function, developed as a systematic code for joint source-network coding, has the same advantage of the sequential function, but contains extra benefits in the way the coding is done. The coding operation is performed in three steps: (i) replicate elementary matrix, (ii) adjust output alphabet size and (iii) reorder values. We start by calculating the minimum value of the input alphabet sizes: \( l_{\text{min}} = \min(l_1, l_2, ..., l_i) \). We then produce a elementary matrix with the results of the addition in finite fields of the form \( 2^j \), being \( j \) a positive integer, for \( i \) dimensions with \( l_{\text{min}} \) levels each. This operation is equivalent to bitwise exclusive OR operation of the input values and we choose it because the number of occurrences of the different values is uniform, and each appears only once in each array of each dimension. The resulting alphabet size of the output is the same of the inputs, i.e. \( l_{\text{min}} \). If the alphabet size required in the output does not match the one in the matrix, a sum or modulo operation is performed (ii), to increase or reduce the number of output levels, respectively. If the output alphabet size \( l_0 \) is higher than \( l_{\text{min}} \), we subsequently reorder the arrays in the larger dimension (iii), so that the repeated values have the larger possible distance between them in all directions. The aim is that the combinations that result in the same value are as distant and different from each other as possible, because with the extra information we get from the correlations we are likely to have an area in which the probability of that value is higher. If the distance between the potential values is sufficiently large compared to the area, we can discard the values that fall outside the expected area and be more accurate in our estimate. These are the reasons why we built this type of function and expect it to have a high performance when used for source-network coding operations in a scenario in which the correlation between the sources is to be explored.

IV. DECODING MODEL AND ALGORITHM

As explained in [8], using a factor graph offers the great advantage of allowing for a scalable decoding solution in large-scale networks. The standard implementation of the optimal MMSE (minimum mean-square error) decoder is unfeasible in these cases, since the complexity grows exponentially with the number of network nodes. Factor graphs enable us to use the sum-product algorithm [11], which can make the complexity grow linearly with the size of the network [8], depending on the system parameters. We show how to construct both source and network models and combine them in a single factor graph: the joint source-network model.

A. Subgraph Construction

To construct our decoder model, we need to known the flow rates to multicast the information through the network for each edge. However, the subgraph construction method described in [9], which provides us with those rates, is only defined for two correlated sources, whereas we aim to address scenarios with an arbitrary number of correlated sources. Thus, we propose an extension to an arbitrary number of correlated sources, by considering pairs of sources using the source-optimized clustering model described in [10]. We apply the cluster optimization algorithm with a maximum cluster size of two, using the model for two correlated sources [9] within each cluster. Between the clusters, we shall assume an uncorrelated model, considering only the most important component, namely the correlation inside the clusters. The procedure is illustrated in the example of Fig. 2.

![Fig. 2: Virtual nodes and edges for 6 correlated sources grouped in 4 uncorrelated clusters to apply the subgraph construction method](image)

B. Network Model

The network is defined by the set of nodes \( \mathcal{N} \), the set of source nodes \( \mathcal{S} \), the set of sink nodes \( \mathcal{T} \), the nodes’ positions,
and the transmission range \( r \). We distribute the nodes randomly in a unit square and set the network links as the connections between the nodes whose Euclidean distance does not exceed \( r \). To each edge \( e_{kl} \) we assign a capacity \( c_{kl} \) and a weight \( w_{kl} \) (set to any chosen value, such as a function of the distance, latency or energy), so that we can apply the subgraph construction method described above to the original network.

The network model and adding the functions that contain the information about the correlation among the sources within clusters with size two. We create a function node that represents the probability mass function between each two source nodes that belong to the same cluster. For each sink, we construct a factor graph representing both the network operations and the source correlations, in order to jointly decode the messages that reach that sink.

**D. Decoding based on the Sum-Product Algorithm**

Our decoding process is based on the sum-product algorithm [11], which involves passing messages on the factor graph. It can be used to compute the marginal functions of all nodes efficiently and exactly in a cycle-free factor graph, and can also be run in factor graphs with cycles albeit without guaranteeing convergence to the MMSE [11]. We use the messages that reach each decoder to initialize the nodes representing those messages in the corresponding factor graph. After a sufficient number of iterations, the messages will get to the source nodes. As the factor graphs constructed for the decoder have cycles, a stopping criterion is required. We set the total number of iterations to the total number of nodes in the factor graph, so that the messages reach all the factor graph nodes at least once. The messages that are obtained in the factor graph source nodes after the simulation provide the probabilities \( p(i_n|m_i(w)) \) in (1), allowing us to calculate the estimates.

**E. Complexity Analysis**

For the complexity analysis we will consider the factor graphs for each sink separately. Let \( D \) be the maximum degree of a variable or function node and \( L \) be the number of levels of a variable node. As we are looking for an upper bound to the complexity of calculating one message, we can consider the degree of the node as an estimate for the number of outgoing edges. In the network model, we assume \( M \) as the number of messages that arrive at the targeted sink and \( N \) as the total number of nodes in the network (not the original but the obtained subgraph). We introduce \( I \) as the total number of iterations performed, usually set to the total number of nodes in the factor graph (variable and function nodes), which can be upper bounded by \( 2 \times M \times N \). The overall complexity is then upper bounded by \( I \times (M \times N \times O(D^2 \times L) + M \times (N - 1) \times O(D \times L^2)) \approx O(I \times M \times N \times D^2 \times L^2) \).

The first important conclusion is that, in the worst case, the complexity grows quadratically with the size of the network \( N \). The only parameter that influences the complexity exponentially is \( L^D \). We can limit the value of \( L \) by the capacity constraints and the value of \( D \) can also be limited by protocols constructing the overlay network in the real network nodes.

**V. PERFORMANCE ANALYSIS**

An exemplary network with 30 nodes is considered, serving as a proof-of-concept, although the approach works for arbitrary scenarios. We define the following parameters: 6 sources, 3 sinks, quantizer resolution of 3 bits, edge’s capacity constraint of 6 bits, edge’s weight as a function of its length (10
TABLE I: System SNR in dB

<table>
<thead>
<tr>
<th></th>
<th>Rand</th>
<th>Seq</th>
<th>JSNC</th>
<th></th>
<th>Rand</th>
<th>Seq</th>
<th>JSNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta = 0.25$</td>
<td>8.21</td>
<td>13.71</td>
<td>13.68</td>
<td>$\beta = 1$</td>
<td>8.83</td>
<td>14.60</td>
<td>14.68</td>
</tr>
</tbody>
</table>

times the nodes’ distance), and 20000 samples per source. For the encoder mapping function, we use the Diophantine index assignment for distributed source coding, described in [12].

Fig. 4: (a) Original network with $N = 30$ randomly distributed nodes in a unit square with $r = 0.3$, and (b) the same network after the subgraph construction optimization for a capacity constraint of 6 bits per edge and using a strong correlation scenario with $\beta = 0.25$.

To compare the performance with different coding operations, we apply in the network illustrated in Fig. 4 the three mapping functions described in Section III, denoted in Table I by "Rand", "Seq", and "JSNC", respectively. In order to simulate different correlation scenarios, we vary the coefficient $\beta$ describing how correlation decays exponentially with Euclidean distance. To evaluate the overall performance of our decoding scheme, we measure the output signal-to-noise ratio (SNR) given by

$$\text{Output SNR} = 10 \cdot \log_{10} \left( \frac{||u||^2}{||u - u'||^2} \right) \text{ in dB.} \quad (2)$$

Fig. 4(b) shows the results of the subgraph construction method, which yields the transmission rates, when we set $\beta$ to 0.25 to represent strong correlation between the source data. Considering weakly correlated sources, with $\beta$ set to 1.0, the constructed subgraph is similar to the one in Fig. 4(b) but uses an extra edge, since it needs to transmit more data. The obtained SNR values for the three different mapping functions are presented on Table I. It is important to note that the SNR values in strong and weak correlation cases cannot be compared directly, since also the overall cost of the obtained subgraph (sum of the edges’ costs, linearly proportional to their rates) in each of the situations is different. In this particular example, the total cost of the subgraph is 15.6% higher in the weak correlation scenario. Anyway, the decoder works in both cases with good performance, in particular for the sequential and the joint source-network coding schemes.

VI. CONCLUSIONS

This work was motivated by the challenges of efficiently transmitting data in large wireless sensor network scenarios. Optimal solutions call for joint source-network coding, which come at the cost of high complexity decoders. Our main goal is to design and implement a coding and decoding strategy inspired in the approach proposed in [7]. The methodology we adopted uses a graphical model based on factor graphs where the sum-product algorithm [11] is run to represent complex scenarios, allowing a feasible decoder complexity, confirmed in our complexity analysis. The proposed decoding model has the following distinguishing characteristics: (i) we combine a source-optimized clustering method with a subgraph construction method for two correlated sources, extending the latter to an arbitrary number of correlated sources; (ii) we construct a feasible decoder by combining our source and network models, allowing for joint decoding; and (iii) we propose three different coding strategies, including a systematic coding scheme for joint source-network decoding. We implemented a working prototype using Matlab, leading to the following conclusions: (i) the use of our subgraph construction method extension to a higher number of correlated sources is viable, and (ii) the proposed systematic coding scheme for joint source-network decoding performs better than a random mapping function.

REFERENCES