Embedded Index Coding

Alexandra Porter
Department of Computer Science
Stanford University
Stanford, CA
amporter@stanford.edu

Mary Wootters
Departments of Computer Science
and Electrical Engineering
Stanford University
Stanford, CA
marykw@stanford.edu

***Abstract*—Motivated by applications in distributed storage and distributed computation, we introduce *embedded index coding* (EIC). EIC is a type of distributed index coding in which nodes in a distributed system act as both senders and receivers of information. We show how embedded index coding is related to index coding in general, and give characterizations and bounds on the communication costs of optimal embedded index codes. We also define *task-based* EIC, in which each sending node encodes and sends data blocks independently of the other nodes. Task-based EIC is more computationally tractable and has advantages in applications such as distributed storage, in which senders may complete their broadcasts at different times. Finally, we give heuristic algorithms for approximating optimal embedded index codes, and demonstrate empirically that these algorithms perform well.**

***Index Terms*—Index Coding, Distributed Storage, Coded Computation**

I. Introduction

In *index coding*, defined by [2], sender(s) encode data blocks into messages which are broadcast to receivers. The receivers already have some of the data blocks, and the goal is to take advantage of this "side information" in order to minimize the number of messages broadcast. For example, if node r_1 knows a data block b_1 and node r_2 knows block b_2, a sender S can broadcast $b_1 \oplus b_2$. Then r_1 can cancel out b_1 and r_2 can cancel b_2 such that both nodes learn a distinct new block from a single broadcast message.

Index coding is typically studied in the models depicted in Figures 1a and 1b, where the senders are distinct from the receivers. In this paper, we consider a setting—depicted in Figure 1c—where the senders *are* the receivers. This model is motivated by applications in distributed storage and distributed computation. For example, in coded computation, e.g. [8], the *shuffle* phase consists of nodes communicating computed values with each other.

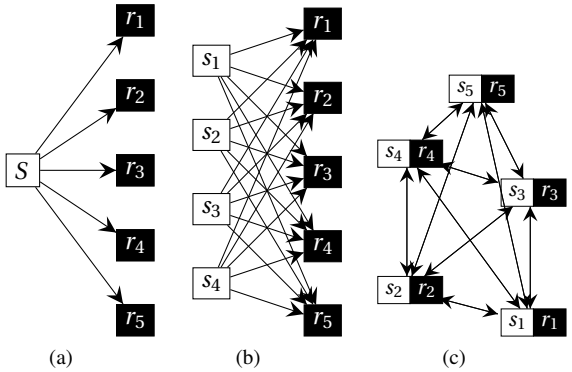


Fig. 1: Communication model for (a) centralized index coding with sender S, receivers $r_1, ..., r_5$; (b) general multi-sender index coding with senders $s_1, .., s_4$ and receivers $r_1, ..., r_5$; and (c) embedded index coding, a special case of (b) with joint sender and receiver nodes $r_1 = s_1, ..., r_5 = s_5$.

We call this model *embedding index coding* (EIC). EIC can be seen as a special case of the multi-sender index coding model in Figure 1b. In this paper, we will demonstrate that by considering EIC as a special case, we can prove new results and design faster algorithms than are available for the more general multi-sender index coding problem.

We also introduce a new notion of solution to an embedded index coding problem called a *task-based* solution. In a task-based solution, the communication can be partitioned into independent tasks, so that each receiver is only reliant on a single sender to get a par-

This work is partially supported by NSF grant CCF-1657049 and NSF CAREER grant CCF-1844628. AP is partially supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1656518.

978-1-5386-6900-6/19/$31.00 ©2019 IEEE

ticular block.[1] Task-based solutions are easier to reason about, and we will give efficient heuristics to find good ones. Moreover, task-based solutions can be more robust to failures or delays: if a sender's messages are corrupted or lost, the messages from other senders can still be used fully to decode data blocks.

We prove several results establishing relationships between centralized (single-sender) index coding, EIC, and task-based EIC. In particular, we show in Theorem 2 that the optimal length of a solution to a general EIC problem is only a factor of two worse than the optimal length in the centralized model. In Lemma 1 we give an upper bound on the length of the best task-based solution to an EIC problem.

Finally, based on (the proofs of) the bounds above, we design heuristics for designing general EIC schemes and task-based EIC schemes, and we give evidence that these heuristics perform well.

A. Related Work

Index coding was first introduced by [2], based on the Informed-Source Coding on Demand (ISCOD) model proposed by [3], and many extensions and variations have been studied. We focus on linear index coding. As shown in [2], the rate of an optimal index code is given precisely by the *minrank* of a corresponding matrix and we build on this result in our work. Embedded index codes are a special case of the linear multi-sender index codes in [5] and [7], which both consist of multiple senders and multiple receivers, but as two distinct and non-overlapping sets of nodes; this is the setting depicted in Figure 1b. In [7] rank minimization is used in an approach similar to our method; we compare the two methods in Section IV. The model in Figure 1c is used in [4], but the setting is less general than EIC's because all nodes want all blocks they don't already have.

There are several other related notions, including *composite coding* [1] and *Instantly Decodable Network Codes (IDNC's)* [6]; we defer a more in-depth discussion of related work to the full version [9].

II. Framework

In this section we describe the model for embedded index coding. Let n be the number of nodes and m be the number of data blocks. Let $\mathcal{D} \in \mathbb{F}^m$ be the set of data blocks, for an arbitrary finite field $\mathbb{F}$. We assume that each node can do local computation and can broadcast information over an error-free channel to all other nodes.

[1]We note that this is a generalization of *Instantly Decodable Network Codes* [6].

We then define an embedded index coding problem in terms of what values of $\mathcal{D}$ nodes *have* and *need*.

Definition 1. *An* embedded index coding (EIC) problem *is specified by a pair of matrices $R, B \in \{0,1\}^{n\times m}$ s.t.* $\text{supp}(B) \cap \text{supp}(R) = \emptyset$.

Informally, given EIC problem (R, B), node i *needs* block j if $R_{ij} = 1$ and *has* block j if $B_{ij} = 1$. Each node i will broadcast a set of $b_i \in \mathbb{N}$ linear combinations of its blocks. $\boldsymbol{e}_j$ denotes the j^{th} standard basis vector.

Definition 2. *For an embedded index coding problem (R, B) a* linear broadcast solution *which solves (R, B) is a collection of matrices $\beta^{(1)}, ..., \beta^{(n)}$ and integers $b_1, ..., b_n$ with $\beta^{(i)} \in \mathbb{F}_2^{b_i \times m}$ so that:*

- *the j^{th} column of $\beta^{(i)}$ is zero if $B_{ij} = 0$*
- *for each $i \in [n]$ and each $j \in [m]$ s.t. $R_{ij} = 1$, there is some vector $\boldsymbol{\alpha}^{(i,j)} \in \mathbb{F}_2^{\sum_\ell b_\ell + m}$ so that*

$$\boldsymbol{e}_j = \boldsymbol{\alpha}^{(i,j)} \cdot \begin{bmatrix} -\beta^{(1)}- \\ \cdots \\ -\beta^{(n)}- \\ \overline{diag}(\overline{B_i}) \end{bmatrix}.$$

- *The* length *of an EIC solution is $\Sigma_\ell b_\ell$, the number of symbols broadcast. We also refer to this as the* communication cost *of the solution.*

To use a linear broadcast solution, each node i computes and broadcasts $\beta^{(i)} \cdot \mathcal{D}$. This can be computed locally because the only non-zero columns of $\beta^{(i)}$ correspond to non-zero entries of row B_i, i.e. blocks node i has. Then each node i can decode the blocks it wants using the set of decoding vectors $\{\boldsymbol{\alpha}^{(i,j)} : R_{ij} = 1\}$ and the received messages, $\beta^{(1)} \cdot \mathcal{D}, ..., \beta^{(n)} \cdot \mathcal{D}$.

A. Problem Graph and Problem Matrix

In this section we extend the work of [2] to represent EIC problems as graphs.

Definition 3. *Let $P = \{(i, j) : R_{ij} = 1\}$ be the set of requirement pairs of a EIC problem defined by (R, B).*

Definition 4. *Given an EIC problem (R, B) we define the* problem graph *G as follows. Let the vertex set be $V(G) = \{v_{(i,j)} : R_{i,j} = 1\}$. Let the (directed) edge set be $E(G) = \{(v_{(i,j)}, v_{(x,y)}) : B_{xj} = 1$ or $j = y\}$.*

Figure 2 shows an example of a problem graph.

It was shown by [2] that in the case of each node requesting a single, unique block, the length of an optimal index code is given by the *minrank,* defined below. We will generalize the result of [2] to our more general notion of a problem graph in Theorem 1.

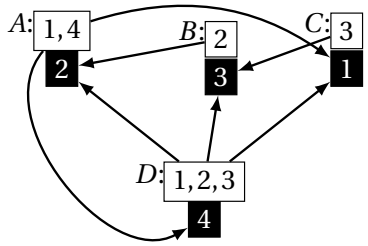


Fig. 2: Problem graph G. Each pair of boxes is a node, where the black box contains indices of requested data blocks and the white box contains indices of side information blocks. For example, node A is requesting block $\mathcal{D}_2$ and has blocks $\mathcal{D}_1, \mathcal{D}_4$ in its side information.

Definition 5. *Given a graph $G = (V, E)$, we say that a matrix $A \in \{0,1\}^{|V|\times|V|}$* fits *$G$ if $A_{kk} = 1$ for all $k \in [|V|]$ and for any $k, \ell \in |V|$,$(k, \ell) \notin E$ implies that $A_{k\ell} = 0$. The* minrank *of a graph G in field $\mathbb{F}_2$, denoted $\mathrm{minrk}_2(G)$, is the rank of the lowest rank matrix A which fits G with entries in $\mathbb{F}_2$:*

$$\mathrm{minrk}_2(G) := \min\{\mathrm{rk}_2(A) : A \textit{ fits } G\}$$

B. Task-Based Solutions

We now define a particular type of solution, motivated by the efficiency of finding a solution and which may be more useful in the presence of failures.

Definition 6. *A task $T = (k, M)$ is defined by a sender node k and a set of pairs $M \subseteq \{(i, j) : R_{ij} = 1$ and $B_{kj} = 1\} \subseteq P$.*

Definition 7. *A* task-based solution *is a linear broadcast solution $\beta^{(1)}, ..., \beta^{(n)}$ such that for each $(i, j) \in [n]\times[m]$ such that $R_{ij} = 1$, there exists a $\beta^{(\ell)}$ and coefficient vector $\boldsymbol{\alpha}_\ell^{(i,j)}$ such that $\boldsymbol{e}_j = \boldsymbol{\alpha}_\ell^{(i,j)} \cdot \begin{bmatrix} - & -\beta^{(\ell)}- \\ & \overline{diag}(\overline{B_i}) & - \end{bmatrix}$.*

Informally, a task-based solution is a linear solution in which each node i decodes each requested block j using only messages from one sender node, i.e. one vector $\beta^{(x)} \cdot \mathcal{D}$ for some node x. A task-based solution to (R, B) is also related to the corresponding problem graph G by specifying a partition of the vertices. Let $N^+(v_{(i,j)}) \subseteq V(G)$ denote the out-edge neighborhood of a vertex $v_{(i,j)} \in V(G)$.

Definition 8. *For each node $i \in [n]$ let $N_i := N^+(v_{(i,j)})$ for any $j \in [m]$ s.t. $R_{ij} = 1$ denote the* sender neighborhood *of node i.*

Remark 1. *Each node i and its sender neighborhood N_i (or any subset of N_i) together form an instance of an index coding problem with a single source as originally defined by [2].*

Definition 9. *The task-based solution* neighborhood partition *of the problem graph G is the set $\{\tilde{N}_1, ...\tilde{N}_n\}$ where $\tilde{N}_i \subseteq N_i$ for all $i \in [n]$ such that: $v_{(x,y)} \in \tilde{N}_i$ if and only if $\beta^{(i)}$ is the matrix used in the task-based solution such that x decodes block y using $\boldsymbol{e}_y = \boldsymbol{\alpha}_i^{(x,y)} \cdot \begin{bmatrix} - & -\beta^{(i)}- \\ & \overline{diag}(\overline{B_x}) & - \end{bmatrix}$.*

A task-based solution exists for the EIC problem shown in Figure 2, using sender neighborhoods $N_D = \{A, B, C\}$ and $N_A = \{D\}$. The messages for the task executed by node D are $\mathcal{D}_1 \oplus \mathcal{D}_2$ and $\mathcal{D}_2 \oplus \mathcal{D}_3$, and the message broadcast by node A for its task is $\mathcal{D}_4$. Then nodes A, B, and C each decode their requested block from the task executed by node D, and node D decodes its request from the task executed by node A.

While we only study task-based solutions on the EIC model, task-based solutions can also be used for multi-sender index coding in general.

C. Centralized Solutions

We also want to compare decentralized solutions to EIC problems to the optimal centralized index coding solution. Thus we define a solution to an EIC problem which assumes some oracle server exists with access to all of $\mathcal{D}$ (and has no requirements itself).

Definition 10. *For an EIC problem defined by (R, B), a* centralized linear broadcast solution *which solves (R, B) is a matrix β and integer b with $\beta \in \mathbb{F}_2^{b\times m}$ such that for each $i \in [n]$ and each $j \in [m]$ s.t. $R_{ij} = 1$, there is some vector $\boldsymbol{\alpha}^{(i,j)} \in \mathbb{F}_2^{b+m}$ so that*

$$\boldsymbol{e}_j = \boldsymbol{\alpha}^{(i,j)} \cdot \begin{bmatrix} - & \beta & - \\ & \overline{diag}(\overline{B_i}) & - \end{bmatrix}.$$

Finally, we use the following symbols to denote the optimal lengths for each type of solution:

Definition 11. *Let $(C)_{(R,B)}$ denote the minimum length of a centralized linear broadcast solution to the EIC problem (R, B) as defined in Definition 10.*

Let $(D)_{(R,B)}$ denote the minimum length of a decentralized linear broadcast solution to the EIC problem (R, B) as defined in Definition 2.

Let $(T)_{(R,B)}$ denote the minimum length of a decentralized and task-based solution to the EIC problem (R, B) as defined in Definition 7.

III. Minimum Code Lengths and Relationships

In this section, we analyze the values of $(C)_{(R,B)}$, $(D)_{(R,B)}$, and $(T)_{(R,B)}$ for a given (R, B). We drop

(R, B) from the notation when comparing two of these under the same (R, B) in general. In the full version, we show by example that the values of $(C)_{(R,B)}$, $(D)_{(R,B)}$, and $(T)_{(R,B)}$ may all be distinct, and we prove general separation results. In this extended abstract, we focus on upper bounds on $(D)_{(R,B)}$ and $(T)_{(R,B)}$. These will be useful in the next section when we design algorithms.

First, we discuss centralized solutions to an EIC problem, and introduce some useful machinery.

Our problem graph is a generalization of the *side information graph* of [2], and we prove the following generalizion of Theorem 5 of [2].

Theorem 1. *Given EIC (R, B) and the corresponding problem graph G, $(C)_{(R,B)} = \text{minrk}_2(G)$.*

The proof of Theorem 1 follows closely the proof of Theorem 5 in [2], see [9] for details.

Next, we prove an upper bound on (D). It can easily be seen that minimum length of a decentralized embedded index code is at least the minimum length of the corresponding centralized index code since EIC is a harder problem: that is, $(C) \leq (D)$. Thus we are interested in how much longer a decentralized embedded index code must be than an index code in the centralized model. Perhaps surprisingly, it turns out that not much is lost in the EIC model!

Theorem 2. *Given an EIC problem defined by (R, B), $(D)_{(R,B)} \leq 2 \cdot (C)_{(R,B)}$.*

The proof of Theorem 2 (available in [9]) uses Theorem 1, and is essentially a transformation which takes any centralized algorithm for a EIC problem and produces a decentralized algorithm with at most twice the length. We note that the proof of Theorem 2 crucially uses the EIC formulation; this shows why considering EIC separately as a special case of multi-sender index coding can be valuable.

Finally, we prove an upper bound on (T). In the full version [9], we give a characterization of $(T)_{(R,B)}$ in terms of the minrank of the $\tilde{N}_i$ in an optimal neighborhood partition as defined in Section II-B. Using this, we obtain the following upper bound on $(T)_{(R,B)}$ which we will use in Section IV-B to design a heuristic algorithm for finding task-based solutions.

Lemma 1. *Given an EIC problem defined by (R, B), let $\mathcal{N}$ be the set of all possible neighborhood partitions (Definition 9). Then*

$$(T)_{(R,B)} \leq \min_{\{\tilde{N}_1,\ldots,\tilde{N}_n\} \in \mathcal{N}} \sum_{i=1}^{n} \chi(\overline{\tilde{N}_i}).$$

IV. Algorithms

In this section, we use results from the previous section to design heuristics for finding good EIC solutions. We also demonstrate empirically that our algorithms perform well. More precisely, we describe two algorithms, one to approximate the best decentralized solution and one to approximate the best task-based solution.

A. Approximating (D)

The proof of Theorem 2 gives a method to design a decentralized solution to an EIC problem. More precisely, this method (pseudocode given in the full version [9]) first computes or approximates an optimal centralized solution with length $(C)_{(R,B)}$ and then uses the transformation used in the proof of Theorem 2 to arrive at a decentralized solution with length at most $2 \cdot (C)_{(R,B)}$.

We compare the cost of this method to the methods given by [5] and [7]. More precisely, the main computational task of both methods is computing the minrank of a graph, by searching over a set of possible fitting matrices. In practice, we may wish to use a heuristic to approximate the minrank; however, one way to compare the speed of these algorithms is to compute the size of the search space that would be required to compute the minrank exactly.[2] Figure 3 shows how the base-2 logarithm of the search space for our algorithm compares to that of the Combined LT-CMAR procedure of [7]. Except for the smallest values of n and p, S_{EIC} is smaller, meaning that our algorithm has a smaller search space than the combined LT-CMAR algorithm.

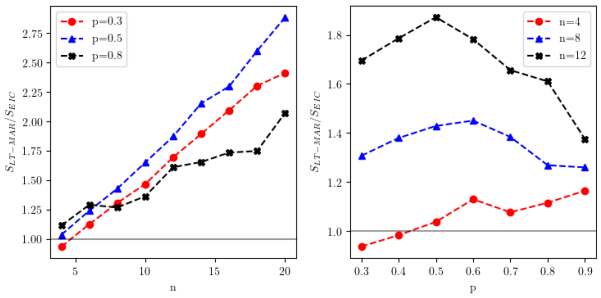
B. Approximating (T)

Computing a task-based solution consists of two main steps: finding a neighborhood partition (Definition 9) and finding an index coding solution to the task defined by each $\tilde{N}_i$ for sender node i. Using the upper bound of Lemma 1, we will show how to find a good choice for the neighborhood partition. We begin with a definition.

Definition 12. *Given a problem graph G for some EIC problem (R, B) with sender node neighborhoods $N_1, ..., N_n$, let the set of* neighborhood-cliques *be $\mathscr{C} := \{V(C) : C$ is a maximal clique in $G[N_i]$ for some $N_i\}$*

The following theorem (proven in [9]) shows that using neighborhood-cliques, finding a neighborhood decomposition to minimize the upper bound in Lemma 1 reduces to solving a minimum cover problem.

[2]We note that if the minrank is computed exactly, then Combined LT-CMAR becomes an exact algorithm, while our algorithm is a two-approximation.



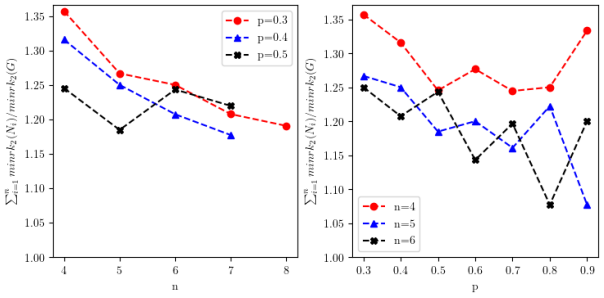
(a) (b)

Fig. 3: $S_{LT-CMAR}$ is the value $\log_2$ of the search space size for a given (R, B) using the method of [7]. S_{EIC} is the value $\log_2$ of the search space size using our method of approximating with the corresponding centralized solution. Ratios $S_{LT-CMAR}/S_{EIC}$ are plotted for the averages over sets of 20 Erdős-Renyi graphs with the given number of vertices n and probability p for each directed edge. When $S_{LT-CMAR}/S_{EIC} > 1$ our algorithm has a strictly smaller search space.

Theorem 3. *Given EIC problem (R, B) and corresponding problem graph G, solving for the neighborhood partition $\tilde{N}_1, ...\tilde{N}_n$ to minimize $\sum_{i=1}^{n} \chi(\tilde{N}_i)$ is exactly equivalent to the min cover problem over vertices of G with sets $\mathscr{C} = \{V(C_i) : C_i$ is a maximal clique in $G[N_i]\}$.*

This theorem gives us a constructive algorithm for $\tilde{N}_1, ..., \tilde{N}_n$. Since mincover is NP-hard solving for these will be as well, but we can use existing mincover approximation algorithms. Our algorithm to approximate an optimal task-based solution to an EIC problem computes a mincover of the vertices of the corresponding problem graph G, using the set of maximal cliques, $\mathscr{C}$. The cliques used then specify the neighborhood partition, since each is fully contained the neighborhood of some node. See the full version for algorithm details [9].

Figure 4 shows the ratio of the length of our approximately optimal task based solution compared to the length of the optimal centralized solution. This ratio upper bounds the ratio of a true optimal task based solution to the corresponding centralized solution. In all of our experiments this approximation ratio is upper-bounded by 1.4. As in the experiments in Figure 3, Erdős-Renyi graphs are randomly generated for a variety of values for n, the number of nodes, and p, the directed edge probability. As the size of the graph increases for a fixed edge probability, the ratio appears to converge.



(a) (b)

Fig. 4: Ratio of the length of our task-based solution returned by our algorithm to the length of the optimal centralized solution.[3]

For a fixed number of nodes, there also appears to be some upper bound on the ratio even as the probability of each edge goes to 1.

References

[1] Fatemeh Arbabjolfaei, Bernd Bandemer, Young-Han Kim, Eren Şaşoğlu, and Lele Wang. On the capacity region for index coding. In *2013 IEEE International Symposium on Information Theory*, pages 962–966. IEEE, 2013.

[2] Ziv Bar-Yossef, Yitzhak Birk, T Jayram, and Tomer Kol. Index coding with side information. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*.

[3] Yitzhak Birk and Tomer Kol. Coding on demand by an informed source (ISCOD) for efficient broadcast of different supplemental data to caching clients. *IEEE Transactions on Information Theory*, 52(6):2825–2830, 2006.

[4] Salim El Rouayheb, Alex Sprintson, and Parastoo Sadeghi. On coding for cooperative data exchange. In *2010 IEEE Information Theory Workshop on Information Theory (ITW 2010, Cairo)*, pages 1–5. IEEE, 2010.

[5] Jae-Won Kim and Jong-Seon No. Linear index coding with multiple senders and extension to a cellular network. *arXiv preprint arXiv:1901.07136*, 2019.

[6] Anh Le, Arash S Tehrani, Alexandros G Dimakis, and Athina Markopoulou. Instantly decodable network codes for real-time applications. In *2013 International Symposium on Network Coding (NetCod)*, pages 1–6. IEEE, 2013.

[7] Min Li, Lawrence Ong, and Sarah J Johnson. Multi-sender index coding for collaborative broadcasting: A rank-minimization approach. *IEEE Transactions on Communications*, 2018.

[8] Songze Li, Mohammad Ali Maddah-Ali, and A Salman Avestimehr. Fundamental tradeoff between computation and communication in distributed computing. In *Information Theory (ISIT), 2016 IEEE International Symposium on*, pages 1814–1818. IEEE, 2016.

[9] Alexandra Porter and Mary Wootters. Embedded index coding. *arXiv preprint arXiv:1904.02179*, 2019.

[3]Sample sizes in these experiments are 10 random graphs, except $p = 0.9, n = 6$ which only uses 5, since the search space for the brute-force minrank algorithm explodes, increasing exponentially in the number of graph edges.